Build a React + TypeScript App from Zero to Production Build

Created by Manuel Herrera Mesén · November 30, 2026

Part 4 of 4 | react | typescript | node | express | frontend | serie-react

IN THIS ARTICLE

- Step 00: What We Are Going to Build
- Step 01: Setting Up the Environment
- Step 02: Creating the Monorepo
- Step 03: The Contract
- Step 04: The API
- Step 05: The Frontend Foundation
- Step 06: Sign-In
- Step 07: The Pokédex
- Step 08: The Detail Page
- Step 09: The Team Builder
- Step 10: The Gyms
- Step 11: Quality
- Step 12: Continuous Integration
- Step 13: The Production Build
- Step 14: Wrap-Up
- Bibliography

> Part 4 of 4 of the series "Learn React by Building a Real App". It follows part 3: Advanced React.

> **Note:** Non-profit educational fan project. Not affiliated with, sponsored or endorsed by Nintendo, Game Freak or The Pokémon Company. Pokémon and their names are trademarks of their respective owners. Data provided by PokéAPI.

In the three previous parts you learned React through small examples, each with one idea. A real app does not arrive like that: it arrives as an empty folder and a list of screens, and the ideas have to live together in the same code, with an API behind them, tests that protect it, and a build you can ship.

In this part you build **Dex 506**, the trainer's Pokédex you saw on the cover of part 1, from an empty folder to the production build. Every step comes with the complete files, the exact commands, what you should see in the terminal, and how to check that it worked.

How to read this guide. It is a guide to follow with the terminal open, step by step and in order: each step builds on the previous one. When a step uses a React concept, it links to the section of the series where it was explained; what is not React (Node, Express, pnpm, continuous integration) is explained here, the first

time it appears. Files that already exist are not repeated in full: you see the **change**, in diff format, where lines with **+** are added and lines with **-** are removed. At the end of each step there is a zip with that step's complete code, so you can compare yours or pick up from there. As in the previous parts, the code is in English and the text on screen is in Spanish.

I built all of the code step by step, in the same order as this article, with Node 26.13.1 and pnpm 12.21.0, on macOS 27.2 and on Linux inside Docker: each step's code passes its lint, its type check, its tests, and its build with no network, and it is exactly the code in its zip. I did not test it on Windows; where a command differs in PowerShell, I give you the variant.

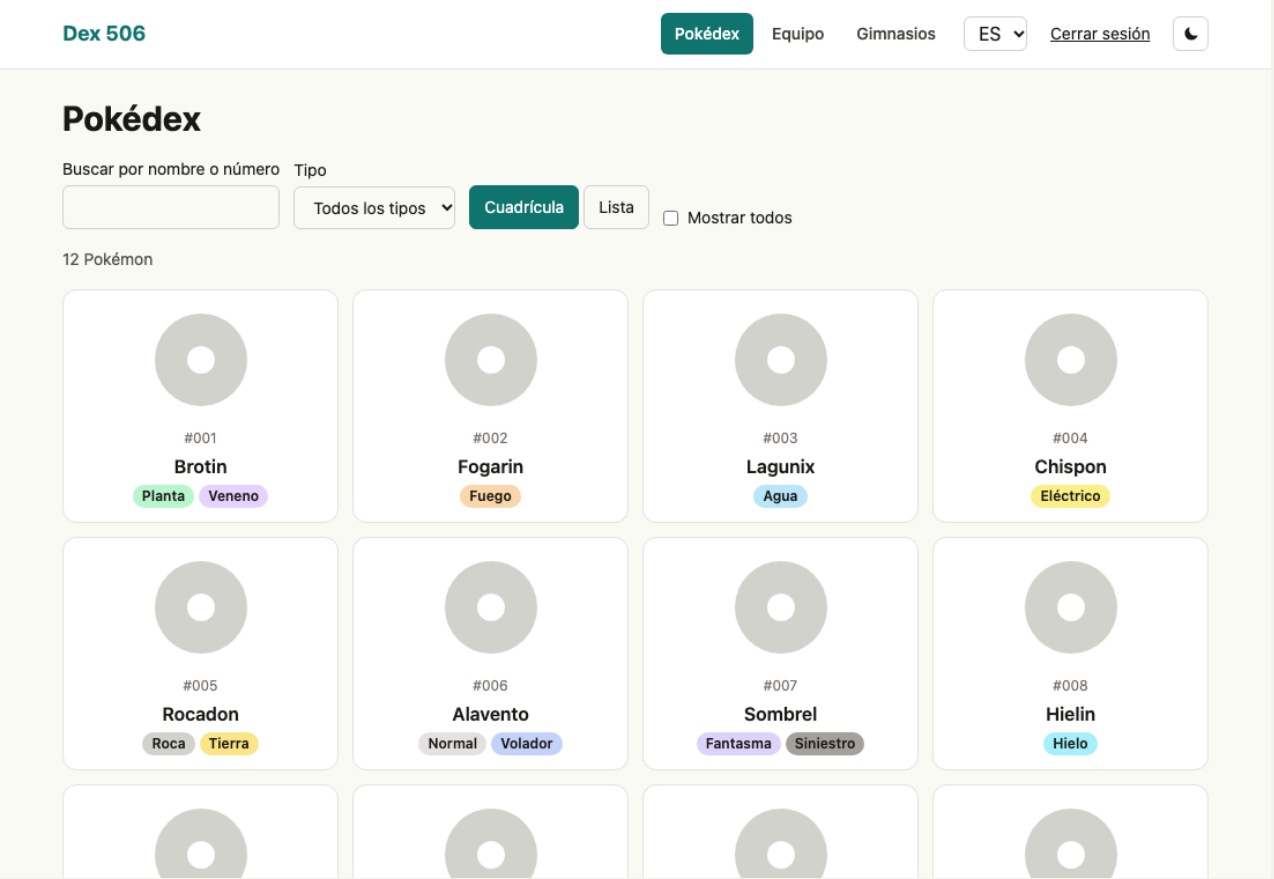
By the end of this part you will be able to:

1. organize a full stack project in a monorepo with pnpm;
2. define a contract shared by the API and the web app with Zod;
3. build an API with Node and Express 5, with sessions, validation, errors, and tests;
4. put together Dex 506's five screens with everything from parts 1 to 3;
5. test the app at three levels, measure its performance, and review its security;
6. run everything in continuous integration and generate the production build.

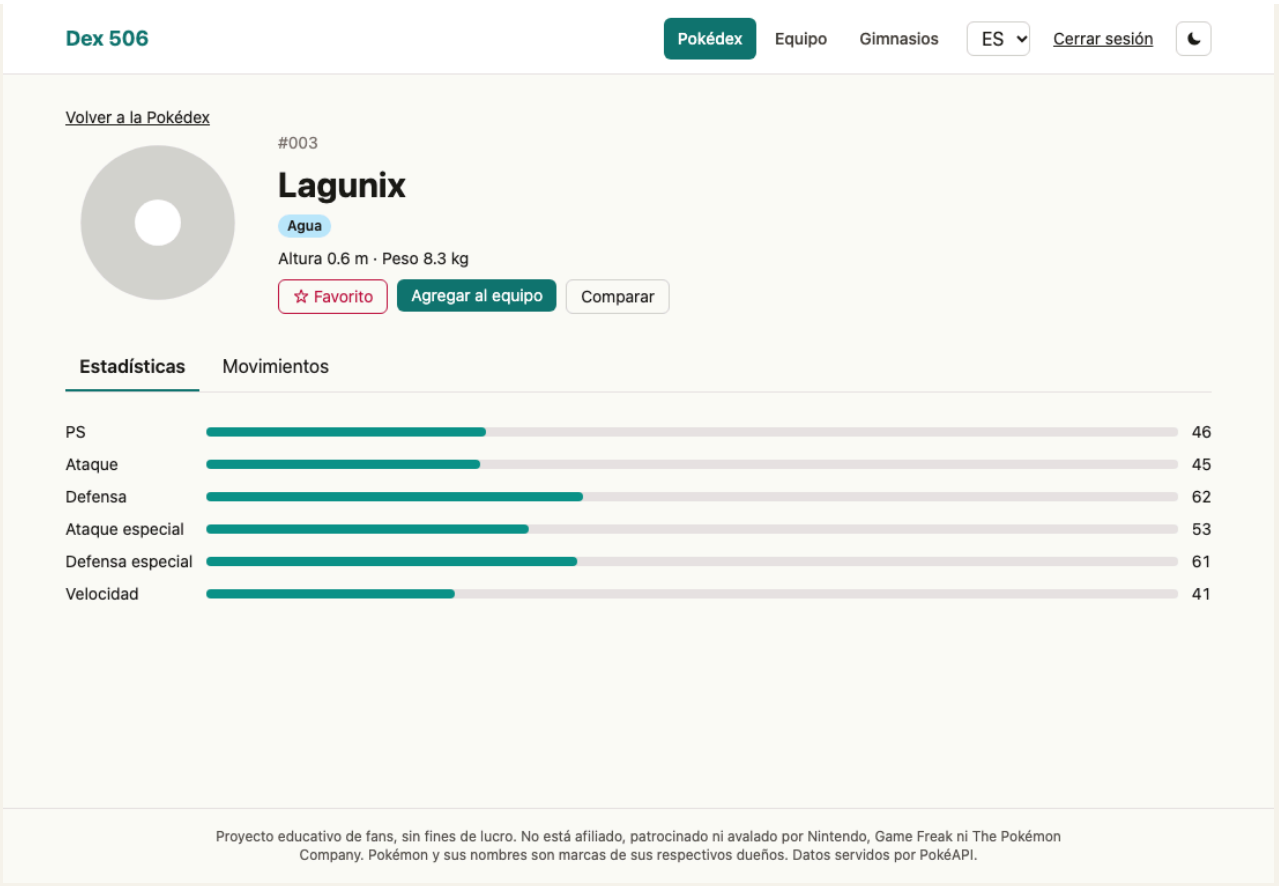
Step 00: What We Are Going to Build

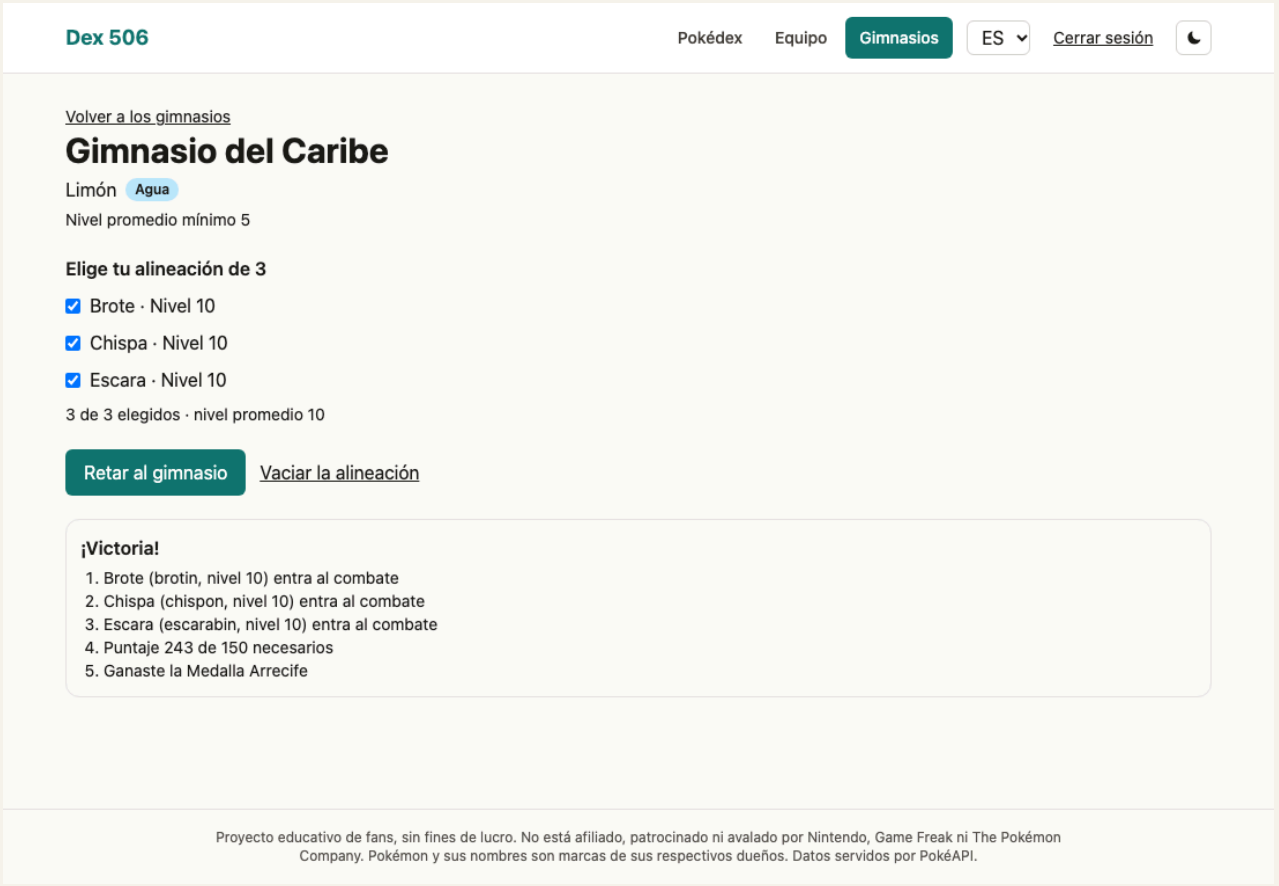
In this step you will see the finished app, what it has to do, and how the code is split up.

The five screens



1. **Sign-in.** You sign in with an email and a password. The team and gym screens require a session; if you open them without one, the app takes you to the login and, once you sign in, brings you back to where you were going.
2. **Pokédex.** The first 151 Pokémon, with a search box, a type filter, pagination, and a grid or list view. The filter lives in the URL, so it can be shared.
3. **Detail page.** A single Pokémon in detail: its image, types, size, stats, and moves, in tabs. From there you mark it as a favorite, add it to your team, or compare it with another one.
4. **Team builder.** Your team of up to six members. Each member has a nickname, a level, and one to four moves, and is added or edited in a two-step form.
5. **Gyms.** Eight made-up gyms, one per region of Costa Rica. You pick a lineup of three members from your team, challenge the gym and, if you win, you take its badge.





The screenshots use the made-up creatures the app is tested with. With the real data, each card shows its Pokémon.

What you do not see

Besides the screens, the app has to meet these requirements:

- **Tests:** every business rule, every API route, and every screen has tests, plus an end-to-end run in a real browser.
- **Accessibility:** it works with a keyboard and with a screen reader, and it passes an automated audit in both of its themes, light and dark.
- **Performance:** typing in the search box with all 151 Pokémon on screen responds within one frame. It is the measurement you saw in "Performance".
- **Security:** the session lives in an **`httpOnly`** cookie and no secret reaches the browser.
- **A production build** that the API serves together with the web app, from a single process.

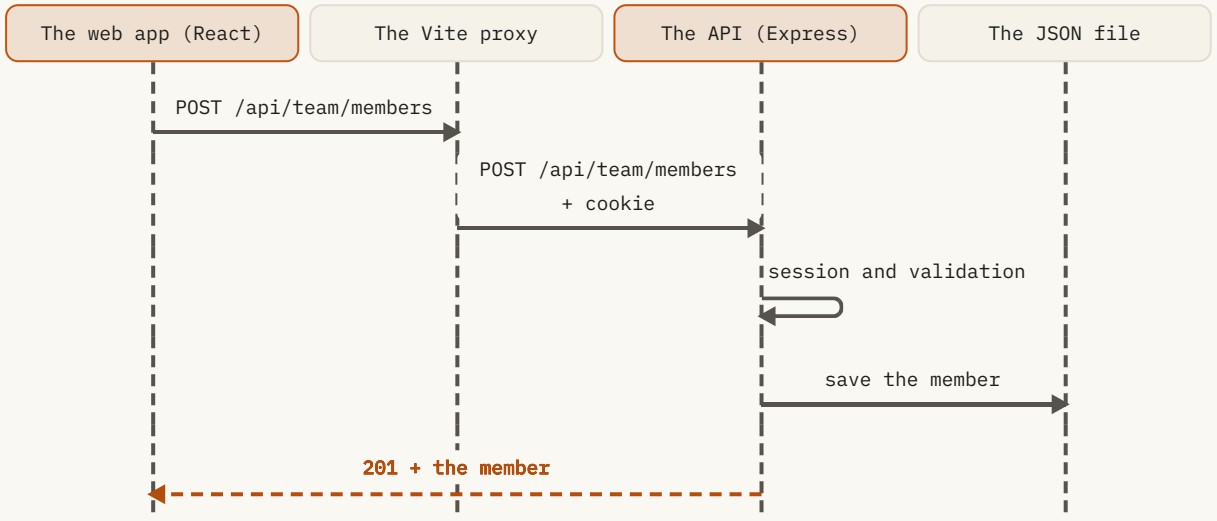
How the code is split up

Dex 506 is a **monorepo**: a single repository with several packages that are developed together.

Package	What it is
`packages/contracts`	The Zod schemas and the business rules shared by the API and the web app
`apps/api`	The API, with Node and Express, which stores the data in a JSON file
`apps/web`	The web app, with React, Vite, and Tailwind

This is how a request travels, from the moment you tap a button until the screen changes:

A Dex 506 request, from the screen to the data



The team builder form sends the new member. The web app and the API are served from the same origin, so the request goes to the same address as the page.

In development, Vite forwards anything that starts with /api to the API. The browser adds the session cookie on its own.

The API checks the cookie and validates the body with the same Zod schema the form used.

The change is written to the JSON file, atomically.

The API responds with the created member; if something fails, with an error of a known shape, which the web app shows on the matching field.

The data for the 151 Pokémon comes from **PokéAPI**, a free public API. This repository does not store any Pokémon data or images: a script downloads them to your machine, into folders that git ignores, and the tests use twelve made-up creatures.

Step 01: Setting Up the Environment

In this step you will install the tools and check their versions.

Node.js

Node.js is the program that runs JavaScript outside the browser: the Dex 506 API runs on it, and so do the web app's tools (Vite, Vitest). This guide uses Node 26, the LTS version (*Long Term Support*, the one that receives fixes for the longest time). Install it from nodejs.org or with a version manager such as nvm.

Node runs **`.ts`** files directly, stripping the types as it loads them, with no compilation step: it does this by default since versions 22.18 and 23.6, and it has been a stable feature since 25.2 (Node.js, *Modules: TypeScript*). This app takes advantage of it: the API has no build, and Node runs it exactly as you write it.

pnpm

pnpm is a package manager, like npm: it downloads the libraries the project uses. This guide uses it for two reasons: it handles monorepos well, with its **workspaces** (several packages in one repository, installed together and able to use each other), and it stores each version of each library only once on disk, even when several projects use it.

The pnpm documentation recommends installing it with its script on macOS and Linux, and with npm on Windows (pnpm, *Installation*):

```
curl -fsSL https://get.pnpm.io/install.sh | sh -
```

On Windows, in PowerShell:

```
npm install -g pnpm@latest
```

Close the terminal and open a new one, so it picks up the new command.

git and the editor

git keeps the history of your changes: if a step goes wrong, you go back to the previous one. Install it from git-scm.com and, if this is your first time using it, tell it your name and email, which go into every commit (otherwise your first **`git commit`** fails with "Author identity unknown"):

```
git config --global user.name "Your name"
git config --global user.email "you@example.com"
```

Any editor works; with Visual Studio Code, the Oxc extension (Oxc is the project behind oxlint, the tool from step 02 that checks the app's code) and the Prettier extension (the formatter) flag errors and format on save.

Checking the versions

```
node --version
pnpm --version
git --version
```

You should see **`v26`** on the first line (the guide uses **`v26.13.1`**) and **`12`** on the second (the guide uses **`12.21.0`**). Any recent version of git works. The project pins the exact pnpm version in its **`package.json`**: if yours is different, pnpm downloads and uses the project's version (pnpm, *CLI settings*).

Accounts

You do not need any account to follow the guide. Only step 12, continuous integration, uses a free GitHub account.

> The usual mistake: **`pnpm: command not found`**. It happens if you are still in the terminal where you installed it: open a new one.

Step 02: Creating the Monorepo

In this step you will create the project folder with the configuration all of its packages share.

Create an empty folder, go into it, and start a git repository:

```
mkdir dex-506
cd dex-506
git init
```

All the files in this step go in the root of **dex-506**.

The root package

package.json describes the project: its name, the Node and pnpm versions it requires, the **scripts** (named commands you run with **pnpm <name>**), and the tools all the packages share.

package.json

```
{
  "name": "dex-506",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "packageManager": "pnpm@12.21.0",
  "engines": {
    "node": ">=26.13.0"
  },
  "scripts": {
    "dev": "pnpm --parallel --filter \"./apps/*\" dev",
    "lint": "oxlint",
    "format": "prettier --write .",
    "format:check": "prettier --check .",
    "typecheck": "pnpm -r --if-present typecheck",
    "test": "vitest run",
    "build": "pnpm -r --if-present build"
  },
  "devDependencies": {
    "@types/node": "26.6.6",
    "oxlint": "1.94.0",
    "prettier": "3.10.1",
    "typescript": "7.0.2",
    "vitest": "5.1.0"
  }
}
```

- **"private": true** prevents it from being published to the npm registry by mistake.
- **"type": "module"** says that **.js** and **.ts** files use ES modules (**import** and **export**).
- **packageManager** pins pnpm 12.21.0, and **engines** requires Node 26.13 or newer.
- **pnpm -r** (*recursive*) runs the script in every package that has it, and **--if-present** keeps the packages that do not have it from failing. **--filter** picks which packages to run a command in, and **--parallel** runs them at the same time: **pnpm --parallel --filter "./apps/*" dev** starts the API and the web app together, which arrive in steps 04 and 05.
- **devDependencies** are the tools used only for development and testing (the linter, Vitest); **dependencies** are the libraries the code needs in order to run.

- Versions are exact, with no **^**: that way whoever follows this guide installs the same thing the guide tested.

pnpm-workspace.yaml says where the workspace packages are: inside **apps/** and **packages/**.

pnpm-workspace.yaml

```yaml
packages:
  - 'apps/*'
  - 'packages/*'
```

TypeScript

tsconfig.base.json is the TypeScript configuration all the packages inherit:

tsconfig.base.json

```json
{
  "compilerOptions": {
    "target": "es2023",
    "module": "esnext",
    "moduleResolution": "bundler",
    "moduleDetection": "force",
    "strict": true,
    "noUncheckedIndexedAccess": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true,
    "verbatimModuleSyntax": true,
    "erasableSyntaxOnly": true,
    "allowImportingTsExtensions": true,
    "noEmit": true,
    "skipLibCheck": true
  }
}
```

- **strict** turns on every strict check, and **noUncheckedIndexedAccess** makes reading **lista[i]** give **T | undefined**: an index might not exist.
- **noEmit**: TypeScript only checks types, it does not generate JavaScript. Node runs the API's **.ts** files by stripping the types, and Vite builds the web app.
- **erasableSyntaxOnly** forbids the TypeScript syntax that cannot just be erased, such as **enum**. It is what Node needs in order to strip the types when it loads a file.
- **allowImportingTsExtensions** lets you write **import './rules.ts'** with the extension, which is how Node finds the file.

- **`verbatimModuleSyntax`** requires you to mark whatever is only a type with **`import type`**, so it gets erased entirely.

Tests, lint, and formatting

`vitest.config.ts` makes a single **`pnpm test`** run the tests of every package: each package has its own configuration, and this file only lists them.

vitest.config.ts

```ts
import { defineConfig } from 'vitest/config'

// A single command (`pnpm test`) runs every package's tests: each one declares its own
// configuration and they are only listed here.
export default defineConfig({
  test: {
    projects: ['packages/*', 'apps/*'],
  },
})
```

A **linter** reads the code and flags likely errors without running it. Dex 506 uses **oxlint**, the same one as the Vite template from part 1:

.oxlintrc.json

```json
{
  "$schema": "./node_modules/oxlint/configuration_schema.json",
  "plugins": ["typescript", "oxc", "import"],
  "ignorePatterns": ["**/dist/**", "**/node_modules/**", "**/.cache/**", "**/.data/**"],
  "rules": {
    "no-console": ["warn", { "allow": ["warn", "error", "info"] }]
  }
}
```

`no-console` warns when a forgotten **`console.log`** is left behind, and lets **`warn`**, **`error`**, and **`info`** through, which the API uses on purpose.

> – Going deeper
>
> the standard for TypeScript projects is ESLint with typescript-eslint. Dex 506 uses oxlint because typescript-eslint does not support TypeScript 7 yet, the version this series uses. For a project on TypeScript 6, ESLint is still a good option.

Prettier formats the code: quotes, semicolons, line length. Its rules and the files it leaves alone:

.prettierrc.json

```
{
  "semi": false,
  "singleQuote": true,
  "trailingComma": "all",
  "printWidth": 100
}
```

.prettierignore

```
pnpm-lock.yaml
dist
.cache
.data
coverage
```

What git does not keep

`.gitignore` lists what git should not keep: what gets generated (**`node_modules`**, **`dist`**), secrets (**`.env`**), and the PokéAPI data, which is downloaded in step 04:

.gitignore

```
node_modules
dist
coverage
*.local
.env
.env.*
!.env.example

# PokéAPI data: `pnpm seed` downloads it and it is NEVER committed (see README,
"Propiedad intelectual").
apps/api/.cache
apps/api/.data

# Playwright
test-results
playwright-report

.DS_Store
.vscode/*
!.vscode/extensions.json
.idea
```

`.nvmrc` tells nvm, the Node version manager, which version to use in this folder:

.nvmrc

```
26
```

The README and the license

The **README.md** explains what the project is, the intellectual property notice, and the commands:

README.md

```
# Dex 506

La app de la serie «Aprende React construyendo una app real»: la Pokédex de un
entrenador, con
React 19, TypeScript, una API en Node con Express y pruebas.

> **Propiedad intelectual.** Proyecto educativo de fans, sin fines de lucro. No está
afiliado,
> patrocinado ni avalado por Nintendo, Game Freak ni The Pokémon Company. Pokémon y sus
nombres son
> marcas de sus respectivos dueños. Datos servidos por PokéAPI.
>
> Este repositorio no guarda datos ni imágenes de Pokémon: `pnpm seed` los descarga de
PokéAPI a
> carpetas que git ignora, y las pruebas usan criaturas inventadas (`fixtures/`).

## Requisitos

- Node.js 26 LTS
- pnpm 12

## Comandos

| Comando          | Qué hace                                          |
| ---------------- | ------------------------------------------------- |
| `pnpm install`   | Instala las dependencias de todos los paquetes |
| `pnpm lint`      | Revisa el código con oxlint                       |
| `pnpm typecheck` | Comprueba los tipos de cada paquete               |
| `pnpm test`      | Corre las pruebas de todos los paquetes           |
| `pnpm build`     | Genera el build de producción                     |
| `pnpm format`    | Formatea el código con Prettier                   |
```

And **LICENSE**, the MIT license, which lets anyone use and modify the code:

LICENSE

```
MIT License

Copyright (c) 2026 Manuel Herrera Mesén
```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

The MIT license covers the code only. It does not cover Pokémon names, data or
images, which belong to their respective owners (see README.md).
```

Install and check

```
pnpm install
pnpm lint
pnpm typecheck
```

`pnpm install` downloads the root's tools and creates **`pnpm-lock.yaml`**, the **lockfile**: the exact list of every installed package, with its version. Commit it to git: that way every later install is identical. **`pnpm lint`** finishes with no warnings, and **`pnpm typecheck`** does nothing yet, because there are no packages.

Save the step in git:

```
git add .
git commit -m "paso 02: el monorepo"
```

Download the code for step 02.

Step 03: The Contract

In this step you will write the package the API and the web app share: the shape of each piece of data and the rules of the game.

Why a shared package

The API and the web app talk about the same things: a Pokémon, a team member, a gym. If each one defines its own version, sooner or later they stop matching, and the bug shows up in production. In "Architecture, Quality, and Security" you saw the solution: a **contract**, a shared definition of what is sent and what is received. In Dex 506 it is a workspace package, **`@dex506/contracts`**.

The package

All the files in this step go in **`packages/contracts/`**.

packages/contracts/package.json

```
{
  "name": "@dex506/contracts",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "exports": {
    ".": "./src/index.ts"
  },
  "scripts": {
    "typecheck": "tsc -p tsconfig.json"
  },
  "dependencies": {
    "zod": "4.6.7"
  }
}
```

`exports` says what someone who writes **`import … from '@dex506/contracts'`** gets: the **`src/index.ts`** file, in TypeScript, uncompiled. It works because the two packages that use it understand TypeScript: Node strips the types when it loads it, and Vite transforms it when it builds the web app.

packages/contracts/tsconfig.json

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "lib": ["ES2023"],
    "types": []
  },
  "include": ["src"]
}
```

packages/contracts/vitest.config.ts

```
import { defineProject } from 'vitest/config'
```

```
export default defineProject({
  test: {
    name: 'contracts',
    environment: 'node',
  },
})
```

`defineProject` is the configuration of one project inside the workspace's tests, and **`name`** identifies its tests in the output. **`environment: 'node'`**: these tests do not need a DOM.

The schemas

Each piece of data is defined once, as a Zod schema, and its TypeScript type comes from the schema with **`z.infer`**, as in "Forms" and in "Advanced TypeScript for React".

packages/contracts/src/schemas.ts

```
import { z } from 'zod'

/** The 18 types, in PokéAPI's order. `as const` makes them literals, not `string`. */
export const POKEMON_TYPES = [
  'normal',
  'fighting',
  'flying',
  'poison',
  'ground',
  'rock',
  'bug',
  'ghost',
  'steel',
  'fire',
  'water',
  'grass',
  'electric',
  'psychic',
  'ice',
  'dragon',
  'dark',
  'fairy',
] as const

export const PokemonTypeSchema = z.enum(POKEMON_TYPES)
export type PokemonType = z.infer<typeof PokemonTypeSchema>

export const StatsSchema = z.object({
  hp: z.number().int().nonnegative(),
  attack: z.number().int().nonnegative(),
  defense: z.number().int().nonnegative(),
  specialAttack: z.number().int().nonnegative(),
```

```
  specialDefense: z.number().int().nonnegative(),
  speed: z.number().int().nonnegative(),
})
export type Stats = z.infer<typeof StatsSchema>

export const PokemonSchema = z.object({
  id: z.number().int().positive(),
  name: z.string().min(1),
  types: z.array(PokemonTypeSchema).min(1).max(2),
  stats: StatsSchema,
  /** In decimeters, as PokéAPI delivers it. */
  height: z.number().int().nonnegative(),
  /** In hectograms, as PokéAPI delivers it. */
  weight: z.number().int().nonnegative(),
  spriteUrl: z.url(),
  artworkUrl: z.url().nullable(),
  moves: z.array(z.string()),
})
export type Pokemon = z.infer<typeof PokemonSchema>

/** What the Pokédex shows in the list: no stats and no moves. */
export const PokemonSummarySchema = PokemonSchema.pick({
  id: true,
  name: true,
  types: true,
  spriteUrl: true,
})
export type PokemonSummary = z.infer<typeof PokemonSummarySchema>

export const MAX_TEAM = 6
export const MAX_MOVES = 4

export const TeamMemberInputSchema = z.object({
  pokemonId: z.number().int().positive(),
  nickname: z.string().trim().min(1).max(12),
  level: z.number().int().min(1).max(100),
  moves: z.array(z.string().min(1)).min(1).max(MAX_MOVES),
})
export type TeamMemberInput = z.infer<typeof TeamMemberInputSchema>

export const TeamMemberSchema = TeamMemberInputSchema.extend({
  id: z.string().min(1),
})
export type TeamMember = z.infer<typeof TeamMemberSchema>

export const GymSchema = z.object({
  id: z.string().min(1),
  name: z.string().min(1),
  city: z.string().min(1),
  type: PokemonTypeSchema,
```

```
  badge: z.string().min(1),
  lineupSize: z.literal(3),
  minAvgLevel: z.number().int().min(1).max(100),
  threshold: z.number().int().positive(),
  /** The types that deal the most damage to this gym: they add points in the challenge.
*/
  weakTo: z.array(PokemonTypeSchema),
})
export type Gym = z.infer<typeof GymSchema>

export const ChallengeInputSchema = z.object({
  memberIds: z.array(z.string().min(1)).length(3),
})
export type ChallengeInput = z.infer<typeof ChallengeInputSchema>

export const ChallengeResultSchema = z.object({
  won: z.boolean(),
  score: z.number().int(),
  log: z.array(z.string()),
})
export type ChallengeResult = z.infer<typeof ChallengeResultSchema>

export const UserSchema = z.object({
  id: z.string().min(1),
  email: z.email(),
  trainerName: z.string().min(1),
})
export type User = z.infer<typeof UserSchema>

export const LoginInputSchema = z.object({
  email: z.email(),
  password: z.string().min(1),
})
export type LoginInput = z.infer<typeof LoginInputSchema>

export const ApiErrorSchema = z.object({
  error: z.object({
    code: z.string(),
    message: z.string(),
    /** One message per form field, when it is a validation error. */
    fields: z.record(z.string(), z.string()).optional(),
  }),
})
export type ApiError = z.infer<typeof ApiErrorSchema>

export function pageSchema<T extends z.ZodType>(item: T) {
  return z.object({
    data: z.array(item),
    page: z.number().int().positive(),
    limit: z.number().int().positive(),
```

```
    total: z.number().int().nonnegative(),
  })
}
export type Page<T> = { data: T[]; page: number; limit: number; total: number }

export const PokemonQuerySchema = z.object({
  q: z.string().trim().max(30).optional(),
  type: PokemonTypeSchema.optional(),
  page: z.coerce.number().int().positive().default(1),
  limit: z.coerce.number().int().min(1).max(151).default(24),
})
export type PokemonQuery = z.infer<typeof PokemonQuerySchema>
```

- **POKEMON_TYPES** uses **as const**, so **z.enum** and the **PokemonType** type know the exact 18 types.
- **PokemonSummarySchema** is a **pick** of the full Pokémon: what the list shows, without stats or moves.
- **TeamMemberSchema** extends the member that gets sent with the **id** the API assigns to it.
- **ApiErrorSchema** is the shape of every API error: a code, a message and, if it is a validation error, a message per field.
- **pageSchema** is a generic function: it takes the schema of one item and returns the schema of a page of items. **PokemonQuerySchema** uses **z.coerce** because URL parameters always arrive as text.

The rules

The business rules are pure functions: they take data and return a result, without touching anything else. That is why they can be tested on their own and used on both sides.

packages/contracts/src/rules.ts

```
import type { Gym, Pokemon, TeamMember } from './schemas.ts'

/** The average level of a group of members, rounded down. 0 if there are none. */
export function averageLevel(members: Pick<TeamMember, 'level'>[]): number {
  if (members.length === 0) return 0
  const total = members.reduce((sum, member) => sum + member.level, 0)
  return Math.floor(total / members.length)
}

/** The sum of a Pokémon's six base stats. */
export function baseStatTotal(pokemon: Pick<Pokemon, 'stats'>): number {
  return Object.values(pokemon.stats).reduce((sum, value) => sum + value, 0)
}

export type LineupEntry = {
  member: Pick<TeamMember, 'nickname' | 'level'>
  pokemon: Pick<Pokemon, 'types' | 'stats'>
}
```

```
/**
 * A lineup's score against a gym: each member's level, plus a tenth of its base stats,
plus 40
 * points for each member of a type the gym is weak to.
 */
export function lineupScore(lineup: LineupEntry[], gym: Pick<Gym, 'weakTo'>): number {
  return lineup.reduce((score, { member, pokemon }) => {
    const advantage = pokemon.types.some((type) => gym.weakTo.includes(type)) ? 40 : 0
    return score + member.level + Math.floor(baseStatTotal(pokemon) / 10) + advantage
  }, 0)
}
```

`lineupScore` is the heart of the gyms: it adds up each member's level, a tenth of their stats, and 40 points if their type is strong against the gym.

packages/contracts/src/index.ts

```
export * from './schemas.ts'
export * from './rules.ts'
```

`index.ts` gathers what the package exports.

The tests

packages/contracts/src/rules.test.ts

```
import { describe, expect, it } from 'vitest'
import { averageLevel, baseStatTotal, lineupScore } from './rules.ts'
import { TeamMemberInputSchema } from './schemas.ts'

const stats = { hp: 50, attack: 50, defense: 50, specialAttack: 50, specialDefense: 50,
speed: 50 }

describe('averageLevel', () => {
  it('redondea hacia abajo', () => {
    expect(averageLevel([{ level: 10 }, { level: 11 }])).toBe(10)
  })

  it('devuelve 0 sin miembros', () => {
    expect(averageLevel([])).toBe(0)
  })
})

describe('lineupScore', () => {
  it('suma nivel, un décimo de las estadísticas y la ventaja de tipo', () => {
    const lineup = [
      { member: { nickname: 'Uno', level: 20 }, pokemon: { types: ['water' as const],
stats } },
```

```
      { member: { nickname: 'Dos', level: 30 }, pokemon: { types: ['fire' as const],
stats } },
    ]
    // 20 + 30 of level, 30 + 30 of stats (300 / 10), 40 for the water member
    expect(lineupScore(lineup, { weakTo: ['water'] })).toBe(150)
    expect(baseStatTotal({ stats })).toBe(300)
  })
})

describe('TeamMemberInputSchema', () => {
  it('rechaza un apodo de más de 12 caracteres y más de 4 movimientos', () => {
    const result = TeamMemberInputSchema.safeParse({
      pokemonId: 1,
      nickname: 'UnApodoDemasiadoLargo',
      level: 5,
      moves: ['a', 'b', 'c', 'd', 'e'],
    })
    expect(result.success).toBe(false)
    expect(result.error?.issues.map((issue) => issue.path[0])).toEqual(['nickname',
'moves'])
  })
})
```

These are unit tests, like the ones in "Testing": **`describe`** groups the tests for one function, and the last one checks that the schema rejects a nickname and a list of moves that are too long, and on which fields.

Install and check

```
pnpm install
pnpm typecheck
pnpm test
```

`pnpm install` installs Zod for the new package. **`pnpm typecheck`** checks its types, and **`pnpm test`** runs its four tests: Vitest should finish with **`Tests 4 passed (4)`**.

```
git add .
git commit -m "paso 03: el contrato"
```

Download the code for step 03.

Step 04: The API

In this step you will build the Dex 506 API with Node and Express, with its session, its validation, its errors, and its tests.

Just enough Express

In part 2 you saw just enough HTTP from the side of whoever makes the request. Now you are on the other side: an **HTTP server** is a program that listens for requests on a port and responds to each one. **Express** is the most widely used Node framework for writing them. Its three ideas:

- a **route** connects a method and a URL with the function that responds: **`router.get('/pokemon', (req, res) => …)`**. **`router`** is a group of routes, created with Express's **`Router()`**; **`req`** is the request, and **`res`** is the response you build;
- a **middleware** is a function every request passes through before the route: it can read it, change it, respond, or call **`next()`** to move on;
- an **error handler** is a middleware with four parameters, **`(err, req, res, next)`**, which receives any error from a route.

Express 5, the version in this guide, also sends the errors from an **`async`** route to that handler, without a **`try`**/**`catch`** in each one.

The package

All the API files go in **`apps/api/`**.

apps/api/package.json

```
{
  "name": "@dex506/api",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "node --watch --env-file-if-exists=.env src/server.ts",
    "start": "node --env-file-if-exists=.env src/server.ts",
    "seed": "node scripts/seed.ts",
    "reset": "node scripts/reset.ts",
    "typecheck": "tsc -p tsconfig.json"
  },
  "dependencies": {
    "@dex506/contracts": "workspace:*",
    "cookie-parser": "1.4.7",
    "express": "5.3.0",
    "zod": "4.6.7"
  },
  "devDependencies": {
    "@types/cookie-parser": "1.4.10",
    "@types/express": "5.0.6",
    "@types/supertest": "7.2.1",
    "supertest": "7.3.2"
  }
}
```

- **`"@dex506/contracts": "workspace:*"`** uses the package from step 03, from the same repository.
- **`dev`** runs the server with **`node --watch`**, which restarts it when you change a file, and **`--env-file-if-exists=.env`** loads the **environment variables** from the **`.env`** file if it exists. An environment variable is a value the program reads from the system when it starts, like the port: that way the same code works on your machine and on a server, with different values.
- **`cookie-parser`** reads the cookies of each request, and **`supertest`** makes requests to the app in the tests without starting a server.

apps/api/tsconfig.json

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "lib": ["ES2023"],
    "types": ["node"],
    "module": "nodenext",
    "moduleResolution": "nodenext",
    "resolveJsonModule": true
  },
  "include": ["src", "scripts", "test"]
}
```

module and **moduleResolution** set to **nodenext** make TypeScript resolve imports the way Node does, and **types: ["node"]** adds Node's types (**process**, **Buffer**).

apps/api/vitest.config.ts

```
import { defineProject } from 'vitest/config'

export default defineProject({
  test: {
    name: 'api',
    environment: 'node',
  },
})
```

.env.example documents the API's environment variables. The real **.env** is not committed to git; you create it by copying this one:

apps/api/.env.example

```
# Copy this file to .env and adjust it. Node reads it with --env-file-if-exists (see
package.json).
PORT=3001
# pokeapi: the data `pnpm seed` downloaded · fixtures: the made-up creatures in
fixtures/
DEX_DATA=pokeapi
```

The API's errors

Every error response has the shape of the contract's **ApiError**. **HttpError** is an error that already knows which HTTP status and which **code** to respond with:

apps/api/src/http-error.ts

```
/** An error that already knows which HTTP status and which ApiError `code` to answer
with. */
export class HttpError extends Error {
  readonly status: number
  readonly code: string
  readonly fields: Record<string, string> | undefined

  constructor(status: number, code: string, message: string, fields?: Record<string,
string>) {
    super(message)
    this.status = status
    this.code = code
    this.fields = fields
  }
}
```

```
/**
 * A route parameter as a string. Express types `req.params` as `string | string[] |
undefined`
 * when the route goes through another middleware first; this checks it once.
 */
export function routeParam(params: Record<string, unknown>, name: string): string {
  const value = params[name]
  if (typeof value !== 'string')
    throw new HttpError(400, 'VALIDATION', `Falta el parámetro ${name}`)
  return value
}
```

routeParam reads a URL parameter, like the **:id** in **/team/members/:id**, and fails with a 400 if it is not a string.

apps/api/src/middleware/errors.ts

```
import type { ErrorRequestHandler, RequestHandler } from 'express'
import { ZodError } from 'zod'
import type { ApiError } from '@dex506/contracts'
import { HttpError } from '../http-error.ts'

/** Zod's errors as one message per field: `{ nickname: 'Too big…' }`. */
export function fieldsFrom(error: ZodError): Record<string, string> {
  const fields: Record<string, string> = {}
  for (const issue of error.issues) {
    const key = issue.path.join('.') || '_'
    fields[key] ??= issue.message
  }
  return fields
}

export const notFound: RequestHandler = (req, _res, next) => {
  next(new HttpError(404, 'NOT_FOUND', `No existe la ruta ${req.method} ${req.path}`))
}

/** An error that already carries a 4xx status, like the ones express.json throws. */
function isClientError(err: unknown): err is { status: number } {
  return (
    typeof err === 'object' &&
    err !== null &&
    'status' in err &&
    typeof err.status === 'number' &&
    err.status >= 400 &&
    err.status < 500
  )
}
```

```
/**
 * The last middleware: every error ends up here and leaves with the ApiError shape.
Express 5
 * also brings an `async` handler's errors here, without a try/catch in every route.
 */
export const errorHandler: ErrorRequestHandler = (err, _req, res, next) => {
  // If the response already started (a file halfway sent), Express has to close it: no
JSON here.
  if (res.headersSent) return next(err)
  let status = 500
  let body: ApiError = { error: { code: 'INTERNAL', message: 'Error inesperado del
servidor' } }

  if (err instanceof HttpError) {
    status = err.status
    body = { error: { code: err.code, message: err.message, fields: err.fields } }
  } else if (err instanceof ZodError) {
    status = 400
    body = {
      error: { code: 'VALIDATION', message: 'Hay datos inválidos', fields:
fieldsFrom(err) },
    }
  } else if (isClientError(err)) {
    // Errors from express.json itself: a malformed body (400) or one over the limit
(413).
    status = err.status
    body = {
      error: {
        code:
          status === 413 ? 'PAYLOAD_TOO_LARGE' : status === 400 ? 'BAD_REQUEST' :
'REQUEST_ERROR',
        message:
          status === 413 ? 'El cuerpo del pedido es demasiado grande' : 'El pedido no es
válido',
      },
    }
  } else {
    console.error(err)
  }
  res.status(status).json(body)
}
```

- **notFound** responds 404 to any URL no route handled.
- **errorHandler** goes last. If the response has already started going out (a file sent halfway), it leaves it to Express, which closes it. Otherwise: an **HttpError** goes out with its status; a **ZodError**, the error from a failed validation, goes out as a 400 with a message per field; an error that already carries a 4xx status, like the one from **express.json** with malformed JSON (400) or a body that is too large

(413), goes out with its status; anything else is logged to the console and goes out as a 500, without showing internal details.

Practicing network states

To practice the loading and error states from part 2, this middleware lets you ask the API for a delay or a failure by adding a parameter to the URL of a request to the API (not to the URL of a page in the web app: later on, step 07 shows how to trigger the failure from the browser, and step 11 does it in a test):

apps/api/src/middleware/network.ts

```
import type { RequestHandler } from 'express'
import { HttpError } from '../http-error.ts'

/**
 * Two helpers for practicing the frontend's network states:
 * - `?delay=1500` delays the response (up to 3 seconds);
 * - `?fail=1` answers 503, as if the server were down.
 *
 * Practice helpers only: in production they are off, so nobody can slow down or break
the API.
 */
export const latencyAndChaos: RequestHandler = async (req, _res, next) => {
  if (process.env.NODE_ENV === 'production') return next()
  const delay = Math.min(Number(req.query.delay) || 0, 3000)
  if (delay > 0) await new Promise((resolve) => setTimeout(resolve, delay))
  if (req.query.fail === '1') {
    throw new HttpError(503, 'UNAVAILABLE', 'El servidor no está disponible (simulado
con ?fail=1)')
  }
  next()
}
```

`/api/pokemon?delay=1500` takes a second and a half, and **`/api/pokemon?fail=1`** responds 503, as if the server were down. In production they do nothing: nobody can slow down or bring down the API from the URL. **`NODE_ENV`** is the environment variable that, by convention, tells Node and its libraries whether they are running in production (**`production`**) or not.

The data

The API stores the trainers, sessions, teams, favorites, and badges in a JSON file, **`apps/api/.data/db.json`**, which is created the first time it starts. For a learning app it is enough, and you can read it at a glance; an app with many users would use a database.

apps/api/src/store.ts

```ts
import { randomBytes, randomUUID, scrypt, scryptSync, timingSafeEqual } from
'node:crypto'
import { mkdir, readFile, rename, writeFile } from 'node:fs/promises'
import path from 'node:path'
import { promisify } from 'node:util'
import type { TeamMember, User } from '@dex506/contracts'

type StoredUser = User & { passwordHash: string }
type StoredMember = TeamMember & { ownerId: string }

export type Data = {
  users: StoredUser[]
  /** token → the session's user and when it expires (milliseconds since 1970). */
  sessions: Record<string, { userId: string; expiresAt: number }>
  members: StoredMember[]
  favorites: Record<string, number[]>
  badges: Record<string, string[]>
}

export function hashPassword(password: string, salt = randomBytes(16).toString('hex')):
string {
  return `${salt}:${scryptSync(password, salt, 32).toString('hex')}`
}

const scryptAsync = promisify(scrypt) as (
  password: string,
  salt: string,
  keylen: number,
) => Promise<Buffer>

/** Async: computing the hash does not block the server while other requests wait. */
export async function verifyPassword(password: string, stored: string): Promise<boolean>
{
  const [salt, hash] = stored.split(':')
  if (!salt || !hash) return false
  const candidate = await scryptAsync(password, salt, 32)
  return timingSafeEqual(candidate, Buffer.from(hash, 'hex'))
}

/** The two test trainers. Both passwords are "dex506-demo". */
export function demoUsers(): StoredUser[] {
  return [
    {
      id: 'u1',
      email: 'ana@dex506.test',
      trainerName: 'Ana',
      passwordHash: hashPassword('dex506-demo'),
    },
    {
      id: 'u2',
```

```
      email: 'leo@dex506.test',
      trainerName: 'Leo',
      passwordHash: hashPassword('dex506-demo'),
    },
  ]
}

export function emptyData(): Data {
  return { users: demoUsers(), sessions: {}, members: [], favorites: {}, badges: {} }
}

/**
 * The app's data in memory, saved to a JSON file after every change. Without a file
(`file` set
 * to null), it lives only in memory: that is how the tests use it.
 */
export class Store {
  private data: Data
  private readonly file: string | null
  private saving: Promise<void> = Promise.resolve()

  private constructor(data: Data, file: string | null) {
    this.data = data
    this.file = file
  }

  static inMemory(data: Data = emptyData()): Store {
    return new Store(data, null)
  }

  static async fromFile(file: string): Promise<Store> {
    let raw: string
    try {
      raw = await readFile(file, 'utf8')
    } catch (error) {
      // Only a missing file means "first run". Any other error, or a corrupt JSON
below, stops the
      // server instead of replacing the trainers' data with an empty file.
      if ((error as NodeJS.ErrnoException).code !== 'ENOENT') throw error
      const store = new Store(emptyData(), file)
      await store.save()
      return store
    }
    return new Store(JSON.parse(raw) as Data, file)
  }

  read(): Readonly<Data> {
    return this.data
  }
```

```
  /**
   * Applies a change and saves it. The saves are chained one after another: two
requests at the
   * same time never write the file at once.
   */
  async update(change: (data: Data) => void): Promise<void> {
    change(this.data)
    const saving = this.saving.then(() => this.save())
    this.saving = saving.catch(() => {})
    await saving
  }

  /**
   * Atomic write: first to a temporary file and then `rename`, which replaces the file
in one
   * step. If the process dies halfway through writing, the previous JSON stays intact.
   */
  private async save(): Promise<void> {
    if (!this.file) return
    await mkdir(path.dirname(this.file), { recursive: true })
    const tmp = `${this.file}.${randomUUID()}.tmp`
    await writeFile(tmp, JSON.stringify(this.data, null, 2))
    await rename(tmp, this.file)
  }
}
```

- Passwords are not stored: what is stored is a **hash** made with **`scrypt`**, a function that turns the password into a string you cannot reverse, with a **salt** (a random value that is different for each password). **`timingSafeEqual`** compares without revealing, through timing, how many characters match.
- **`verifyPassword`** uses the asynchronous version of **`scrypt`**: computing the hash takes time, and meanwhile the API keeps handling other requests. **`hashPassword`** is synchronous because it only runs at startup.
- Each session stores its user and when it expires: that way the session also expires on the server, not only the cookie in the browser.
- **`demoUsers`** creates the two test accounts, Ana and Leo, both with the password **`dex506-demo`**.
- **`save`** writes **atomically**: first to a temporary file, and then it renames it. Renaming replaces the file in a single step, so if the process dies in the middle of a write, the previous JSON is left intact.
- **`update`** chains the writes, one after another: two simultaneous requests never write the file at the same time. Each temporary file also has a random name.
- **`fromFile`** only starts with empty data if the file does not exist. If it is corrupt or cannot be read, the API does not start: a clear error is better than replacing the trainers' data.
- **`Store.inMemory()`** writes nothing: it is what the tests use. To start over with the trainers' data, **`pnpm --filter @dex506/api reset`** deletes the file.

The Pokémon are loaded once, at startup, and validated against the contract:

apps/api/src/pokedex.ts

```
import { readFile } from 'node:fs/promises'
import { fileURLToPath } from 'node:url'
import { PokemonSchema, type Pokemon } from '@dex506/contracts'

export const SEEDED_FILE = fileURLToPath(new URL('../.data/pokedex.json',
import.meta.url))
export const FIXTURES_FILE = fileURLToPath(
  new URL('../../../fixtures/pokemon.json', import.meta.url),
)

export type DataMode = 'pokeapi' | 'fixtures'

/**
 * The Pokémon the API serves. With `pokeapi`, the ones `pnpm seed` downloaded; with
`fixtures`,
 * the made-up creatures. Each one is validated against the contract on load: a corrupt
file fails
 * at startup, not in the middle of a request.
 */
export async function loadPokedex(mode: DataMode): Promise<Pokemon[]> {
  const file = mode === 'fixtures' ? FIXTURES_FILE : SEEDED_FILE
  let raw: string
  try {
    raw = await readFile(file, 'utf8')
  } catch {
    throw new Error(
      mode === 'pokeapi'
        ? 'No hay datos de PokéAPI. Corre `pnpm seed` una vez, o usa DEX_DATA=fixtures
en apps/api/.env.'
        : `No se encontró ${file}.`,
    )
  }
  return PokemonSchema.array().parse(JSON.parse(raw))
}
```

With **`DEX_DATA=pokeapi`**, the API uses the ones the script in the next section downloads; with **`DEX_DATA=fixtures`**, the made-up creatures. If a file is corrupt, the API fails at startup, not in the middle of a request.

Downloading the Pokémon

The **`seed`** script downloads Pokémon 1 through 151 from PokéAPI:

apps/api/scripts/seed.ts

```ts
// Downloads Pokémon 1 to 151 from PokéAPI ONCE, stores them as-is in .cache/ (what its fair use
// policy asks: cache locally) and writes to .data/pokedex.json the shape the app uses. Both
// folders are in .gitignore: this data is never committed.
//
// Usage: pnpm seed (from apps/api, or `pnpm --filter @dex506/api seed` from the root)

import { mkdir, readFile, writeFile } from 'node:fs/promises'
import { fileURLToPath } from 'node:url'
import { POKEMON_TYPES, PokemonSchema, type Pokemon, type PokemonType } from '@dex506/contracts'
import { SEEDED_FILE } from '../src/pokedex.ts'

const CACHE_DIR = fileURLToPath(new URL('../.cache/pokeapi/', import.meta.url))
const FIRST = 1
const LAST = 151
const PAUSE_MS = 250

type RawPokemon = {
  id: number
  name: string
  height: number
  weight: number
  types: { slot: number; type: { name: string } }[]
  stats: { base_stat: number; stat: { name: string } }[]
  sprites: {
    front_default: string | null
    other?: { 'official-artwork'?: { front_default: string | null } }
  }
  moves: { move: { name: string } }[]
}

const sleep = (ms: number) => new Promise((resolve) => setTimeout(resolve, ms))

/** A Pokémon's raw JSON: from the cache if it is there, from PokéAPI (and into the cache) if not. */
async function fetchRaw(id: number): Promise<{ raw: RawPokemon; fromCache: boolean }> {
  const file = `${CACHE_DIR}pokemon-${id}.json`
  try {
    return { raw: JSON.parse(await readFile(file, 'utf8')) as RawPokemon, fromCache: true }
  } catch {
    const response = await fetch(`https://pokeapi.co/api/v2/pokemon/${id}`, {
      headers: { 'User-Agent': 'dex-506-tutorial' },
    })
    if (!response.ok) throw new Error(`PokéAPI respondió ${response.status} para el Pokémon ${id}`)
    const raw = (await response.json()) as RawPokemon
    await writeFile(file, JSON.stringify(raw))
```

```
    return { raw, fromCache: false }
  }
}

function stat(raw: RawPokemon, name: string): number {
  return raw.stats.find((entry) => entry.stat.name === name)?.base_stat ?? 0
}

function normalize(raw: RawPokemon): Pokemon {
  const types = raw.types
    .sort((a, b) => a.slot - b.slot)
    .map((entry) => entry.type.name)
    .filter((name): name is PokemonType => (POKEMON_TYPES as readonly
string[]).includes(name))
  return PokemonSchema.parse({
    id: raw.id,
    name: raw.name,
    types,
    stats: {
      hp: stat(raw, 'hp'),
      attack: stat(raw, 'attack'),
      defense: stat(raw, 'defense'),
      specialAttack: stat(raw, 'special-attack'),
      specialDefense: stat(raw, 'special-defense'),
      speed: stat(raw, 'speed'),
    },
    height: raw.height,
    weight: raw.weight,
    spriteUrl: raw.sprites.front_default,
    artworkUrl: raw.sprites.other?.['official-artwork']?.front_default ?? null,
    moves: raw.moves.map((entry) => entry.move.name).sort(),
  })
}

await mkdir(CACHE_DIR, { recursive: true })
const pokedex: Pokemon[] = []
for (let id = FIRST; id <= LAST; id++) {
  const { raw, fromCache } = await fetchRaw(id)
  pokedex.push(normalize(raw))
  if (!fromCache) {
    console.info(`${id}/${LAST} ${raw.name}`)
    await sleep(PAUSE_MS) // pause between downloads: PokéAPI is free and its
maintainers pay for it
  }
}
await mkdir(new URL('../.data/', import.meta.url), { recursive: true })
await writeFile(SEEDED_FILE, JSON.stringify(pokedex))
console.info(`Listo: ${pokedex.length} Pokémon en ${SEEDED_FILE}`)
```

- PokéAPI's fair use policy asks you to cache locally every resource you request (PokéAPI, *documentación v2*). That is why the script saves each response in **`apps/api/.cache/pokeapi/`** and, the next time, reads it from there.
- It waits 250 milliseconds between one download and the next: PokéAPI is free, and its maintainers keep it running.
- **`normalize`** reduces the PokéAPI response to the contract's shape, and **`PokemonSchema.parse`** validates it.

`reset` deletes the trainers' data, without touching the Pokémon:

apps/api/scripts/reset.ts

```
// Deletes the trainers' data (teams, favorites, badges, sessions). It does not touch
the
// PokéAPI data: that is what `pnpm seed` is for.
import { rm } from 'node:fs/promises'
import { fileURLToPath } from 'node:url'

const dbFile = fileURLToPath(new URL('../.data/db.json', import.meta.url))
await rm(dbFile, { force: true })
console.info(`Datos de los entrenadores borrados (${dbFile})`)
```

The test creatures

The tests do not use the PokéAPI data, which is not stored in the repository: they use twelve made-up creatures, in **`fixtures/`**, at the root of the monorepo.

fixtures/README.md

```
# fixtures

Criaturas inventadas para las pruebas y para usar la app sin red (`DEX_DATA=fixtures`).
Tienen la
misma forma que los datos de PokéAPI que normaliza `pnpm seed`, pero los nombres,
estadísticas y
movimientos son inventados: así este repositorio no guarda datos de Pokémon.

Las URLs de sus imágenes apuntan a un dominio que no existe (`dex506.invalid`) a
propósito: la app
muestra su propia silueta cuando una imagen no carga.
```

fixtures/pokemon.json

```
[
  {
    "id": 1,
    "name": "brotin",
```

```
    "types": ["grass", "poison"],
    "stats": {
      "hp": 47,
      "attack": 46,
      "defense": 52,
      "specialAttack": 61,
      "specialDefense": 68,
      "speed": 42
    },
    "height": 8,
    "weight": 64,
    "spriteUrl": "https://dex506.invalid/sprites/1.png",
    "artworkUrl": null,
    "moves": ["hoja-afilada", "polvo-somnifero", "embestida", "latigazo-verde",
"semilla-drenadora"]
  },
  {
    "id": 2,
    "name": "fogarin",
    "types": ["fire"],
    "stats": {
      "hp": 41,
      "attack": 55,
      "defense": 40,
      "specialAttack": 63,
      "specialDefense": 47,
      "speed": 68
    },
    "height": 7,
    "weight": 78,
    "spriteUrl": "https://dex506.invalid/sprites/2.png",
    "artworkUrl": null,
    "moves": ["brasa", "arañazo", "chispa-de-fuego", "humareda"]
  },
  {
    "id": 3,
    "name": "lagunix",
    "types": ["water"],
    "stats": {
      "hp": 46,
      "attack": 45,
      "defense": 62,
      "specialAttack": 53,
      "specialDefense": 61,
      "speed": 41
    },
    "height": 6,
    "weight": 83,
    "spriteUrl": "https://dex506.invalid/sprites/3.png",
    "artworkUrl": null,
```

```
    "moves": ["chorro", "burbuja", "embestida", "ola-pequena"]
  },
  {
    "id": 4,
    "name": "chispon",
    "types": ["electric"],
    "stats": {
      "hp": 33,
      "attack": 57,
      "defense": 38,
      "specialAttack": 52,
      "specialDefense": 47,
      "speed": 88
    },
    "height": 5,
    "weight": 54,
    "spriteUrl": "https://dex506.invalid/sprites/4.png",
    "artworkUrl": null,
    "moves": ["chispazo", "rayo-corto", "agilidad", "embestida"]
  },
  {
    "id": 5,
    "name": "rocadon",
    "types": ["rock", "ground"],
    "stats": {
      "hp": 43,
      "attack": 77,
      "defense": 96,
      "specialAttack": 28,
      "specialDefense": 34,
      "speed": 22
    },
    "height": 5,
    "weight": 188,
    "spriteUrl": "https://dex506.invalid/sprites/5.png",
    "artworkUrl": null,
    "moves": ["derrumbe", "lanzarrocas", "excavar", "defensa-ferrea"]
  },
  {
    "id": 6,
    "name": "alavento",
    "types": ["normal", "flying"],
    "stats": {
      "hp": 38,
      "attack": 47,
      "defense": 37,
      "specialAttack": 39,
      "specialDefense": 33,
      "speed": 59
    },
```

```
    "height": 4,
    "weight": 21,
    "spriteUrl": "https://dex506.invalid/sprites/6.png",
    "artworkUrl": null,
    "moves": ["rafaga", "picotazo", "ataque-rapido", "remolino"]
  },
  {
    "id": 7,
    "name": "sombrel",
    "types": ["ghost", "dark"],
    "stats": {
      "hp": 52,
      "attack": 67,
      "defense": 58,
      "specialAttack": 73,
      "specialDefense": 63,
      "speed": 66
    },
    "height": 9,
    "weight": 38,
    "spriteUrl": "https://dex506.invalid/sprites/7.png",
    "artworkUrl": null,
    "moves": ["sombra-fria", "mordisco", "mirada-fija", "niebla-oscura"]
  },
  {
    "id": 8,
    "name": "hielin",
    "types": ["ice"],
    "stats": {
      "hp": 57,
      "attack": 48,
      "defense": 53,
      "specialAttack": 62,
      "specialDefense": 67,
      "speed": 47
    },
    "height": 7,
    "weight": 66,
    "spriteUrl": "https://dex506.invalid/sprites/8.png",
    "artworkUrl": null,
    "moves": ["escarcha", "viento-helado", "nevada", "embestida"]
  },
  {
    "id": 9,
    "name": "mentalis",
    "types": ["psychic", "fairy"],
    "stats": {
      "hp": 58,
      "attack": 43,
      "defense": 52,
```

```
      "specialAttack": 88,
      "specialDefense": 82,
      "speed": 63
    },
    "height": 8,
    "weight": 45,
    "spriteUrl": "https://dex506.invalid/sprites/9.png",
    "artworkUrl": null,
    "moves": ["mente-clara", "destello", "calma", "brillo-lunar"]
  },
  {
    "id": 10,
    "name": "ferrok",
    "types": ["steel"],
    "stats": {
      "hp": 67,
      "attack": 82,
      "defense": 112,
      "specialAttack": 43,
      "specialDefense": 57,
      "speed": 37
    },
    "height": 13,
    "weight": 372,
    "spriteUrl": "https://dex506.invalid/sprites/10.png",
    "artworkUrl": null,
    "moves": ["golpe-metalico", "defensa-ferrea", "cabezazo", "chirrido"]
  },
  {
    "id": 11,
    "name": "punazo",
    "types": ["fighting"],
    "stats": {
      "hp": 72,
      "attack": 87,
      "defense": 53,
      "specialAttack": 33,
      "specialDefense": 42,
      "speed": 58
    },
    "height": 11,
    "weight": 181,
    "spriteUrl": "https://dex506.invalid/sprites/11.png",
    "artworkUrl": null,
    "moves": ["puno-rapido", "patada-baja", "concentracion", "embestida"]
  },
  {
    "id": 12,
    "name": "escarabin",
    "types": ["bug", "grass"],
```

```
    "stats": {
      "hp": 48,
      "attack": 57,
      "defense": 67,
      "specialAttack": 43,
      "specialDefense": 53,
      "speed": 47
    },
    "height": 6,
    "weight": 49,
    "spriteUrl": "https://dex506.invalid/sprites/12.png",
    "artworkUrl": null,
    "moves": ["picadura", "hoja-afilada", "seda", "embestida"]
  }
]
```

The gyms

The eight gyms are fixed data in the API:

apps/api/src/data/gyms.ts

```ts
import type { Gym } from '@dex506/contracts'

/** Eight made-up gyms, one for each region of Costa Rica the app travels through. */
export const GYMS: Gym[] = [
  {
    id: 'caribe',
    name: 'Gimnasio del Caribe',
    city: 'Limón',
    type: 'water',
    badge: 'Medalla Arrecife',
    lineupSize: 3,
    minAvgLevel: 5,
    threshold: 150,
    weakTo: ['grass', 'electric'],
  },
  {
    id: 'cafetal',
    name: 'Gimnasio del Cafetal',
    city: 'Heredia',
    type: 'grass',
    badge: 'Medalla Grano',
    lineupSize: 3,
    minAvgLevel: 10,
    threshold: 170,
    weakTo: ['fire', 'flying', 'ice', 'bug', 'poison'],
  },
  {
```

```
    id: 'tormenta',
    name: 'Gimnasio de la Tormenta',
    city: 'San José',
    type: 'electric',
    badge: 'Medalla Rayo',
    lineupSize: 3,
    minAvgLevel: 15,
    threshold: 190,
    weakTo: ['ground'],
  },
  {
    id: 'golfo',
    name: 'Gimnasio del Golfo',
    city: 'Puntarenas',
    type: 'flying',
    badge: 'Medalla Brisa',
    lineupSize: 3,
    minAvgLevel: 20,
    threshold: 210,
    weakTo: ['electric', 'ice', 'rock'],
  },
  {
    id: 'niebla',
    name: 'Gimnasio de la Niebla',
    city: 'Cartago',
    type: 'ghost',
    badge: 'Medalla Bruma',
    lineupSize: 3,
    minAvgLevel: 25,
    threshold: 230,
    weakTo: ['ghost', 'dark'],
  },
  {
    id: 'sabana',
    name: 'Gimnasio de la Sabana',
    city: 'Liberia',
    type: 'ground',
    badge: 'Medalla Llanura',
    lineupSize: 3,
    minAvgLevel: 30,
    threshold: 250,
    weakTo: ['water', 'grass', 'ice'],
  },
  {
    id: 'volcan',
    name: 'Gimnasio del Volcán',
    city: 'Alajuela',
    type: 'fire',
    badge: 'Medalla Lava',
    lineupSize: 3,
```

```
    minAvgLevel: 35,
    threshold: 270,
    weakTo: ['water', 'ground', 'rock'],
  },
  {
    id: 'cerro',
    name: 'Gimnasio del Cerro',
    city: 'Pérez Zeledón',
    type: 'rock',
    badge: 'Medalla Cumbre',
    lineupSize: 3,
    minAvgLevel: 40,
    threshold: 290,
    weakTo: ['water', 'grass', 'fighting', 'ground', 'steel'],
  },
]
```

`minAvgLevel` is the average level each gym requires, **`threshold`** is the score you have to reach, and **`weakTo`** lists the types that add the **`lineupScore`** advantage.

The session

When you sign in, the API creates a random **`token`**, stores it in the **`Store`** together with the user, and sends it in an **`httpOnly`** cookie, as you saw in "Architecture, Quality, and Security".

apps/api/src/middleware/auth.ts

```
import type { RequestHandler, Response } from 'express'
import type { User } from '@dex506/contracts'
import { HttpError } from '../http-error.ts'
import type { Store } from '../store.ts'

export const SESSION_COOKIE = 'dex506_session'
/** How long a session lasts: eight hours, in milliseconds. */
export const SESSION_MS = 1000 * 60 * 60 * 8

/** The session's user, which `requireAuth` leaves for the routes that come after it. */
export function currentUser(res: Response): User {
  return res.locals.user as User
}

/** Lets the request through only with a valid session cookie; otherwise, 401. */
export function requireAuth(store: Store): RequestHandler {
  return (req, res, next) => {
    const token: unknown = req.cookies?.[SESSION_COOKIE]
    const session = typeof token === 'string' ? store.read().sessions[token] : undefined
    // The cookie expires in the browser, and the session expires here too: an old token
is not enough.
```

```
    const userId = session && session.expiresAt > Date.now() ? session.userId :
undefined
    const user = store.read().users.find((candidate) => candidate.id === userId)
    if (!user) throw new HttpError(401, 'UNAUTHENTICATED', 'Inicia sesión para
continuar')
    res.locals.user = {
      id: user.id,
      email: user.email,
      trainerName: user.trainerName,
    } satisfies User
    next()
  }
}
```

`requireAuth` lets the request through only if its cookie matches a stored session, and leaves the user in **`res.locals`**, Express's place for passing data from a middleware to the route; **`currentUser`** reads it.

apps/api/src/routes/auth.ts

```
import { randomBytes } from 'node:crypto'
import { Router } from 'express'
import { LoginInputSchema, type User } from '@dex506/contracts'
import { HttpError } from '../http-error.ts'
import { SESSION_COOKIE, SESSION_MS, currentUser, requireAuth } from
'../middleware/auth.ts'
import { hashPassword, verifyPassword, type Store } from '../store.ts'

/** A hash to compare against when the email does not exist: the login takes the same
time. */
const FAKE_HASH = hashPassword('not-a-real-password')

export function authRoutes(store: Store, options: { secureCookies: boolean }): Router {
  const router = Router()

  router.post('/login', async (req, res) => {
    const { email, password } = LoginInputSchema.parse(req.body)
    const user = store.read().users.find((candidate) => candidate.email ===
email.toLowerCase())
    // The hash is checked even for an unknown email: the response time does not reveal
which
    // accounts exist.
    const valid = await verifyPassword(password, user?.passwordHash ?? FAKE_HASH)
    if (!user || !valid) {
      throw new HttpError(401, 'INVALID_CREDENTIALS', 'Correo o contraseña incorrectos')
    }
    const token = randomBytes(32).toString('hex')
    await store.update((data) => {
      // Expired sessions are cleaned up on each login, so the file does not grow
forever.
```

```
      for (const [old, session] of Object.entries(data.sessions)) {
        if (session.expiresAt <= Date.now()) delete data.sessions[old]
      }
      data.sessions[token] = { userId: user.id, expiresAt: Date.now() + SESSION_MS }
    })
    // httpOnly: the page's JavaScript cannot read it, so an injected script cannot
steal it.
    res.cookie(SESSION_COOKIE, token, {
      httpOnly: true,
      sameSite: 'lax',
      secure: options.secureCookies,
      path: '/',
      maxAge: SESSION_MS,
    })
    res.json({ id: user.id, email: user.email, trainerName: user.trainerName } satisfies
User)
  })

  router.post('/logout', async (req, res) => {
    const token: unknown = req.cookies?.[SESSION_COOKIE]
    if (typeof token === 'string') {
      await store.update((data) => {
        delete data.sessions[token]
      })
    }
    res.clearCookie(SESSION_COOKIE, { path: '/' })
    res.status(204).end()
  })

  router.get('/me', requireAuth(store), (_req, res) => {
    res.json(currentUser(res))
  })

  return router
}
```

- **`POST /api/auth/login`** validates the body with the contract's schema, compares the password, and creates the session. The session and the cookie last eight hours (**`SESSION_MS`**, in milliseconds). With an email that does not exist, it still compares the password against a fake hash (**`FAKE_HASH`**): if it answered faster, the response time would reveal which accounts exist. And every login deletes the expired sessions, so the file does not grow forever.
- **`POST /api/auth/logout`** deletes the session and the cookie.
- **`GET /api/auth/me`** returns the session's user, or 401: it is what the web app asks when it opens, to know whether someone is signed in.

The routes

The Pokémon and the types, with no session:

apps/api/src/routes/pokemon.ts

```
import { Router } from 'express'
import {
  POKEMON_TYPES,
  PokemonQuerySchema,
  type Page,
  type Pokemon,
  type PokemonSummary,
} from '@dex506/contracts'
import { HttpError } from '../http-error.ts'

export function findPokemon(pokedex: Pokemon[], id: number): Pokemon {
  const pokemon = pokedex.find((candidate) => candidate.id === id)
  if (!pokemon) throw new HttpError(404, 'POKEMON_NOT_FOUND', `No existe el Pokémon
${id}`)
  return pokemon
}

export function pokemonRoutes(pokedex: Pokemon[]): Router {
  const router = Router()

  router.get('/pokemon', (req, res) => {
    const { q, type, page, limit } = PokemonQuerySchema.parse(req.query)
    const text = q?.toLowerCase()
    const matches = pokedex.filter(
      (pokemon) =>
        (!text || pokemon.name.includes(text) || String(pokemon.id) === text) &&
        (!type || pokemon.types.includes(type)),
    )
    const start = (page - 1) * limit
    const body: Page<PokemonSummary> = {
      data: matches
        .slice(start, start + limit)
        .map(({ id, name, types, spriteUrl }) => ({ id, name, types, spriteUrl })),
      page,
      limit,
      total: matches.length,
    }
    res.json(body)
  })

  router.get('/pokemon/:id', (req, res) => {
    res.json(findPokemon(pokedex, Number(req.params.id)))
  })

  router.get('/types', (_req, res) => {
    res.json(POKEMON_TYPES)
```

```
  })

  return router
}
```

`GET /api/pokemon` filters by text (in the name or the number) and by type, and returns a page. **`findPokemon`** responds 404 if the number does not exist; the other routes use it too.

The team, with a session:

apps/api/src/routes/team.ts

```
import { randomUUID } from 'node:crypto'
import { Router } from 'express'
import {
  MAX_TEAM,
  TeamMemberInputSchema,
  type Pokemon,
  type TeamMember,
  type TeamMemberInput,
} from '@dex506/contracts'
import { HttpError, routeParam } from '../http-error.ts'
import { currentUser, requireAuth } from '../middleware/auth.ts'
import type { Store } from '../store.ts'
import { findPokemon } from './pokemon.ts'

function withoutOwner({
  ownerId: _ownerId,
  ...member
}: TeamMember & { ownerId: string }): TeamMember {
  return member
}

/** The moves must belong to the chosen Pokémon: if not, an error on the `moves` field.
*/
function checkMoves(pokedex: Pokemon[], input: TeamMemberInput): void {
  const pokemon = findPokemon(pokedex, input.pokemonId)
  const unknown = input.moves.filter((move) => !pokemon.moves.includes(move))
  if (unknown.length > 0) {
    throw new HttpError(400, 'VALIDATION', 'Hay datos inválidos', {
      moves: `${pokemon.name} no conoce: ${unknown.join(', ')}`,
    })
  }
}

export function teamRoutes(store: Store, pokedex: Pokemon[]): Router {
  const router = Router()
  // The session is required route by route: a router.use(requireAuth) on a router
mounted at
```

```
  // /api would require it for routes that are not its own too, even ones that do not
exist.
  const auth = requireAuth(store)

  /** The member, if it belongs to the user; 404 if it does not exist, 403 if it is
another trainer's. */
  function ownMember(memberId: string, userId: string) {
    const member = store.read().members.find((candidate) => candidate.id === memberId)
    if (!member) throw new HttpError(404, 'MEMBER_NOT_FOUND', 'Ese miembro no existe')
    if (member.ownerId !== userId)
      throw new HttpError(403, 'FORBIDDEN', 'Ese miembro es de otro entrenador')
    return member
  }

  router.get('/team', auth, (_req, res) => {
    const userId = currentUser(res).id
    res.json(
      store
        .read()
        .members.filter((member) => member.ownerId === userId)
        .map(withoutOwner),
    )
  })

  router.post('/team/members', auth, async (req, res) => {
    const userId = currentUser(res).id
    const input = TeamMemberInputSchema.parse(req.body)
    checkMoves(pokedex, input)
    const size = store.read().members.filter((member) => member.ownerId ===
userId).length
    if (size >= MAX_TEAM) {
      throw new HttpError(422, 'TEAM_FULL', `Tu equipo ya tiene ${MAX_TEAM} miembros`)
    }
    const member = { ...input, id: randomUUID(), ownerId: userId }
    await store.update((data) => {
      data.members.push(member)
    })
    res.status(201).json(withoutOwner(member))
  })

  router.put('/team/members/:id', auth, async (req, res) => {
    const member = ownMember(routeParam(req.params, 'id'), currentUser(res).id)
    const input = TeamMemberInputSchema.parse(req.body)
    checkMoves(pokedex, input)
    const updated = { ...member, ...input }
    await store.update((data) => {
      data.members = data.members.map((candidate) =>
        candidate.id === member.id ? updated : candidate,
      )
    })
```

```
    res.json(withoutOwner(updated))
  })

  router.delete('/team/members/:id', auth, async (req, res) => {
    const member = ownMember(routeParam(req.params, 'id'), currentUser(res).id)
    await store.update((data) => {
      data.members = data.members.filter((candidate) => candidate.id !== member.id)
    })
    res.status(204).end()
  })

  return router
}
```

- Each route gets **`auth`** separately, for the reason its comment gives: with **`router.use`**, a URL that does not exist would respond 401 instead of 404.
- **`checkMoves`** checks that the moves belong to the chosen Pokémon, with an error on the **`moves`** field.
- A seventh member is a 422: the data is valid, but it cannot be applied. **`ownMember`** gives 404 if the member does not exist and 403 if it belongs to another trainer.
- **`withoutOwner`** removes **`ownerId`** from the response: it is internal data.

The favorites:

apps/api/src/routes/favorites.ts

```
import { Router } from 'express'
import type { Pokemon } from '@dex506/contracts'
import { routeParam } from '../http-error.ts'
import { currentUser, requireAuth } from '../middleware/auth.ts'
import type { Store } from '../store.ts'
import { findPokemon } from './pokemon.ts'

export function favoriteRoutes(store: Store, pokedex: Pokemon[]): Router {
  const router = Router()
  // The session is required route by route: a router.use(requireAuth) on a router
mounted at
  // /api would require it for routes that are not its own too, even ones that do not
exist.
  const auth = requireAuth(store)

  router.get('/favorites', auth, (_req, res) => {
    res.json(store.read().favorites[currentUser(res).id] ?? [])
  })

  router.put('/favorites/:pokemonId', auth, async (req, res) => {
    const { id } = findPokemon(pokedex, Number(routeParam(req.params, 'pokemonId')))
    const userId = currentUser(res).id
    await store.update((data) => {
```

```
      const current = data.favorites[userId] ?? []
      if (!current.includes(id)) data.favorites[userId] = [...current, id]
    })
    res.status(204).end()
  })

  router.delete('/favorites/:pokemonId', auth, async (req, res) => {
    const id = Number(routeParam(req.params, 'pokemonId'))
    const userId = currentUser(res).id
    await store.update((data) => {
      data.favorites[userId] = (data.favorites[userId] ?? []).filter((favorite) =>
favorite !== id)
    })
    res.status(204).end()
  })

  return router
}
```

PUT and **DELETE** are **idempotent**: repeating them leaves the same result. Marking the same favorite twice does not duplicate it.

The gyms and the challenge:

apps/api/src/routes/gyms.ts

```
import { Router } from 'express'
import {
  ChallengeInputSchema,
  averageLevel,
  lineupScore,
  type ChallengeResult,
  type Gym,
  type Pokemon,
} from '@dex506/contracts'
import { HttpError, routeParam } from '../http-error.ts'
import { currentUser, requireAuth } from '../middleware/auth.ts'
import type { Store } from '../store.ts'
import { findPokemon } from './pokemon.ts'

export function gymRoutes(store: Store, pokedex: Pokemon[], gyms: Gym[]): Router {
  const router = Router()
  // The session is required route by route: a router.use(requireAuth) on a router
mounted at
  // /api would require it for routes that are not its own too, even ones that do not
exist.
  const auth = requireAuth(store)

  function findGym(id: string): Gym {
```

```
    const gym = gyms.find((candidate) => candidate.id === id)
    if (!gym) throw new HttpError(404, 'GYM_NOT_FOUND', `No existe el gimnasio ${id}`)
    return gym
  }

  router.get('/gyms', auth, (_req, res) => {
    res.json(gyms)
  })

  router.get('/gyms/:id', auth, (req, res) => {
    res.json(findGym(routeParam(req.params, 'id')))
  })

  router.get('/badges', auth, (_req, res) => {
    res.json(store.read().badges[currentUser(res).id] ?? [])
  })

  router.post('/gyms/:id/challenge', auth, async (req, res) => {
    const gym = findGym(routeParam(req.params, 'id'))
    const userId = currentUser(res).id
    const { memberIds } = ChallengeInputSchema.parse(req.body)
    const team = store.read().members.filter((member) => member.ownerId === userId)
    const lineup = memberIds.map((id) => team.find((member) => member.id === id))
    if (
      lineup.some((member) => member === undefined) ||
      new Set(memberIds).size !== memberIds.length
    ) {
      throw new HttpError(
        422,
        'LINEUP_INVALID',
        'La alineación tiene que ser de 3 miembros distintos de tu equipo',
      )
    }
    const members = lineup.filter((member) => member !== undefined)
    if (averageLevel(members) < gym.minAvgLevel) {
      throw new HttpError(
        422,
        'LEVEL_TOO_LOW',
        `${gym.name} exige un nivel promedio de ${gym.minAvgLevel}; tu alineación tiene
${averageLevel(members)}`,
      )
    }
    const entries = members.map((member) => ({
      member,
      pokemon: findPokemon(pokedex, member.pokemonId),
    }))
    const score = lineupScore(entries, gym)
    const won = score >= gym.threshold
    const log = [
      ...entries.map(
```

```
        ({ member, pokemon }) =>
          `${member.nickname} (${pokemon.name}, nivel ${member.level}) entra al
combate`,
      ),
      `Puntaje ${score} de ${gym.threshold} necesarios`,
      won ? `Ganaste la ${gym.badge}` : `${gym.name} te ganó esta vez`,
    ]
    if (won) {
      await store.update((data) => {
        const badges = data.badges[userId] ?? []
        if (!badges.includes(gym.id)) data.badges[userId] = [...badges, gym.id]
      })
    }
    res.json({ won, score, log } satisfies ChallengeResult)
  })

  return router
}
```

The challenge validates the lineup (three different members of your team), requires the gym's average level, computes the score with the contract's **`lineupScore`**, builds a battle **log** (the list of what happened) and, if you win, saves the badge.

The app and the server

apps/api/src/app.ts

```
import cookieParser from 'cookie-parser'
import express, { type Express } from 'express'
import type { Gym, Pokemon } from '@dex506/contracts'
import { GYMS } from './data/gyms.ts'
import { errorHandler, notFound } from './middleware/errors.ts'
import { latencyAndChaos } from './middleware/network.ts'
import { authRoutes } from './routes/auth.ts'
import { favoriteRoutes } from './routes/favorites.ts'
import { gymRoutes } from './routes/gyms.ts'
import { pokemonRoutes } from './routes/pokemon.ts'
import { teamRoutes } from './routes/team.ts'
import type { Store } from './store.ts'

export type AppOptions = {
  pokedex: Pokemon[]
  store: Store
  gyms?: Gym[]
  secureCookies?: boolean
}

/**
```

```
 * The Express app, not listening on any port: `server.ts` starts it and the tests use
it directly
 * with supertest. What it needs (data, storage) comes in as parameters.
 */
export function createApp({
  pokedex,
  store,
  gyms = GYMS,
  secureCookies = false,
}: AppOptions): Express {
  const app = express()
  app.disable('x-powered-by')
  app.use(express.json({ limit: '10kb' }))
  app.use(cookieParser())
  app.use('/api', latencyAndChaos)

  app.get('/api/health', (_req, res) => {
    res.json({ ok: true })
  })
  app.use('/api/auth', authRoutes(store, { secureCookies }))
  app.use('/api', pokemonRoutes(pokedex))
  app.use('/api', teamRoutes(store, pokedex))
  app.use('/api', favoriteRoutes(store, pokedex))
  app.use('/api', gymRoutes(store, pokedex, gyms))

  app.use(notFound)
  app.use(errorHandler)
  return app
}
```

`createApp` puts the app together with what it receives and returns it **without listening** on any port. That way the tests use it with their own data, and the real server starts it:

- **`app.disable('x-powered-by')`** stops announcing that the server is Express.
- **`express.json({ limit: '10kb' })`** reads the JSON body of each request and rejects larger ones.
- Order matters: the middlewares go before the routes, and **`notFound`** and **`errorHandler`** go last.

apps/api/src/server.ts

```
import { fileURLToPath } from 'node:url'
import { createApp } from './app.ts'
import { loadPokedex, type DataMode } from './pokedex.ts'
import { Store } from './store.ts'

const port = Number(process.env.PORT ?? 3001)
const mode: DataMode = process.env.DEX_DATA === 'fixtures' ? 'fixtures' : 'pokeapi'
const dbFile = fileURLToPath(new URL('../.data/db.json', import.meta.url))
```

```
const pokedex = await loadPokedex(mode)
const store = await Store.fromFile(dbFile)
const app = createApp({ pokedex, store, secureCookies: process.env.NODE_ENV ===
'production' })

app.listen(port, (error) => {
  // In Express 5 a failure to listen (a port already in use) arrives here instead of
being thrown.
  if (error) {
    console.error(`No se pudo usar el puerto ${port}: ${error.message}`)
    process.exit(1)
  }
  console.info(
    `API de Dex 506 en http://localhost:${port} (${pokedex.length} Pokémon, datos:
${mode})`,
  )
})
```

`server.ts` reads the configuration from the environment, loads the data, and listens on port 3001. With **`NODE_ENV=production`**, the session cookie requires HTTPS.

The tests

`helpers.ts` builds an app with the made-up creatures and in-memory data, a fresh one for each test, and a supertest **agent** that keeps the cookie between requests, like a browser:

apps/api/test/helpers.ts

```ts
import request from 'supertest'
import { createApp } from '../src/app.ts'
import { loadPokedex } from '../src/pokedex.ts'
import { Store } from '../src/store.ts'

/** An app with the creatures from fixtures/ and in-memory storage, new for every test.
*/
export async function testApp() {
  const pokedex = await loadPokedex('fixtures')
  const store = Store.inMemory()
  return { app: createApp({ pokedex, store }), store, pokedex }
}

/** A supertest agent that keeps the session cookie between requests, like a browser. */
export async function loggedIn(email = 'ana@dex506.test') {
  const ctx = await testApp()
  const agent = request.agent(ctx.app)
  await agent.post('/api/auth/login').send({ email, password: 'dex506-demo'
}).expect(200)
  return { ...ctx, agent }
}
```

apps/api/test/auth.test.ts

```ts
import request from 'supertest'
import { describe, expect, it } from 'vitest'
import { loggedIn, testApp } from './helpers.ts'

describe('autenticación', () => {
  it('inicia sesión con una cookie httpOnly y /me devuelve el entrenador', async () => {
    const { app } = await testApp()
    const agent = request.agent(app)
    const login = await agent
      .post('/api/auth/login')
      .send({ email: 'ana@dex506.test', password: 'dex506-demo' })
    expect(login.status).toBe(200)
    expect(login.headers['set-cookie']?.[0]).toMatch(
      /dex506_session=.+; Max-Age=28800; Path=\/; Expires=.+; HttpOnly; SameSite=Lax/,
    )
    const me = await agent.get('/api/auth/me')
    expect(me.body).toEqual({ id: 'u1', email: 'ana@dex506.test', trainerName: 'Ana' })
  })

  it('rechaza una contraseña incorrecta con 401', async () => {
    const { app } = await testApp()
    const res = await request(app)
      .post('/api/auth/login')
      .send({ email: 'ana@dex506.test', password: 'otra' })
    expect(res.status).toBe(401)
```

```
    expect(res.body.error.code).toBe('INVALID_CREDENTIALS')
  })

  it('sin sesión, /me y el equipo responden 401', async () => {
    const { app } = await testApp()
    expect((await request(app).get('/api/auth/me')).status).toBe(401)
    expect((await request(app).get('/api/team')).status).toBe(401)
  })

  it('una sesión vencida ya no sirve, aunque el navegador mande la cookie', async () =>
{
    const { agent, store } = await loggedIn()
    await store.update((data) => {
      for (const session of Object.values(data.sessions)) session.expiresAt = Date.now()
- 1
    })
    expect((await agent.get('/api/auth/me')).status).toBe(401)
  })

  it('logout cierra la sesión', async () => {
    const { agent } = await loggedIn()
    await agent.post('/api/auth/logout').expect(204)
    expect((await agent.get('/api/auth/me')).status).toBe(401)
  })
})
```

apps/api/test/pokemon.test.ts

```
import request from 'supertest'
import { describe, expect, it } from 'vitest'
import { testApp } from './helpers.ts'

describe('pokémon', () => {
  it('pagina la lista y devuelve el total', async () => {
    const { app } = await testApp()
    const res = await request(app).get('/api/pokemon?page=2&limit=5')
    expect(res.body.total).toBe(12)
    expect(res.body.data.map((p: { id: number }) => p.id)).toEqual([6, 7, 8, 9, 10])
  })

  it('filtra por texto y por tipo', async () => {
    const { app } = await testApp()
    const byName = await request(app).get('/api/pokemon?q=lag')
    expect(byName.body.data.map((p: { name: string }) => p.name)).toEqual(['lagunix'])
    const byType = await request(app).get('/api/pokemon?type=grass')
    expect(byType.body.data.map((p: { name: string }) => p.name)).toEqual(['brotin',
'escarabin'])
  })

  it('un tipo que no existe es un 400 con el campo', async () => {
```

```
    const { app } = await testApp()
    const res = await request(app).get('/api/pokemon?type=cosmico')
    expect(res.status).toBe(400)
    expect(Object.keys(res.body.error.fields)).toEqual(['type'])
  })

  it('el detalle de un Pokémon que no existe es un 404', async () => {
    const { app } = await testApp()
    expect((await request(app).get('/api/pokemon/999')).status).toBe(404)
    expect((await request(app).get('/api/pokemon/3')).body.name).toBe('lagunix')
  })

  it('?delay demora y ?fail=1 responde 503', async () => {
    const { app } = await testApp()
    const start = Date.now()
    await request(app).get('/api/types?delay=200').expect(200)
    expect(Date.now() - start).toBeGreaterThanOrEqual(190)
    const down = await request(app).get('/api/types?fail=1')
    expect(down.status).toBe(503)
    expect(down.body.error.code).toBe('UNAVAILABLE')
  })

  it('una ruta que no existe responde 404 con forma de ApiError', async () => {
    const { app } = await testApp()
    const res = await request(app).get('/api/nada')
    expect(res.status).toBe(404)
    expect(res.body.error.code).toBe('NOT_FOUND')
  })
})
```

apps/api/test/team.test.ts

```
import { describe, expect, it } from 'vitest'
import { loggedIn } from './helpers.ts'

const member = { pokemonId: 3, nickname: 'Lagu', level: 12, moves: ['chorro', 'burbuja']
}

describe('equipo', () => {
  it('agrega, edita y quita un miembro', async () => {
    const { agent } = await loggedIn()
    const created = await agent.post('/api/team/members').send(member).expect(201)
    await agent
      .put(`/api/team/members/${created.body.id}`)
      .send({ ...member, level: 20 })
      .expect(200)
    expect((await agent.get('/api/team')).body).toEqual([
      { ...member, level: 20, id: created.body.id },
    ])
    await agent.delete(`/api/team/members/${created.body.id}`).expect(204)
```

```
    expect((await agent.get('/api/team')).body).toEqual([])
  })

  it('un dato inválido es un 400 con un mensaje por campo', async () => {
    const { agent } = await loggedIn()
    const res = await agent.post('/api/team/members').send({ ...member, nickname: '',
level: 101 })
    expect(res.status).toBe(400)
    expect(Object.keys(res.body.error.fields).sort()).toEqual(['level', 'nickname'])
  })

  it('un movimiento que el Pokémon no conoce es un error en el campo moves', async () =>
{
    const { agent } = await loggedIn()
    const res = await agent.post('/api/team/members').send({ ...member, moves: ['brasa']
})
    expect(res.status).toBe(400)
    expect(res.body.error.fields).toEqual({ moves: 'lagunix no conoce: brasa' })
  })

  it('el séptimo miembro es un 422', async () => {
    const { agent } = await loggedIn()
    for (let i = 0; i < 6; i++) await
agent.post('/api/team/members').send(member).expect(201)
    const res = await agent.post('/api/team/members').send(member)
    expect(res.status).toBe(422)
    expect(res.body.error.code).toBe('TEAM_FULL')
  })

  it('el miembro de otro entrenador es un 403; uno que no existe, un 404', async () => {
    const { agent, store } = await loggedIn()
    await store.update((data) => {
      data.members.push({ ...member, id: 'ajeno', ownerId: 'u2' })
    })
    expect((await agent.delete('/api/team/members/ajeno')).status).toBe(403)
    expect((await agent.delete('/api/team/members/no-existe')).status).toBe(404)
  })

  it('favoritos: marca, lista y desmarca; un Pokémon inexistente es 404', async () => {
    const { agent } = await loggedIn()
    await agent.put('/api/favorites/4').expect(204)
    expect((await agent.get('/api/favorites')).body).toEqual([4])
    await agent.delete('/api/favorites/4').expect(204)
    expect((await agent.get('/api/favorites')).body).toEqual([])
    expect((await agent.put('/api/favorites/999')).status).toBe(404)
  })
})
```

apps/api/test/gyms.test.ts

```ts
import { describe, expect, it } from 'vitest'
import { loggedIn } from './helpers.ts'

async function teamOf(agent: Awaited<ReturnType<typeof loggedIn>>['agent'], level:
number) {
  const ids: string[] = []
  for (const [pokemonId, moves] of [
    [1, ['hoja-afilada']],
    [4, ['chispazo']],
    [12, ['picadura']],
  ] as const) {
    const res = await agent
      .post('/api/team/members')
      .send({ pokemonId, nickname: `M${pokemonId}`, level, moves })
    ids.push(res.body.id)
  }
  return ids
}

describe('gimnasios', () => {
  it('con ventaja de tipo y nivel suficiente, gana la medalla', async () => {
    const { agent } = await loggedIn()
    const memberIds = await teamOf(agent, 10)
    const res = await agent.post('/api/gyms/caribe/challenge').send({ memberIds
}).expect(200)
    expect(res.body.won).toBe(true)
    expect(res.body.log.at(-1)).toBe('Ganaste la Medalla Arrecife')
    expect((await agent.get('/api/badges')).body).toEqual(['caribe'])
  })

  it('un nivel promedio bajo es un 422 que explica cuánto falta', async () => {
    const { agent } = await loggedIn()
    const memberIds = await teamOf(agent, 10)
    const res = await agent.post('/api/gyms/cerro/challenge').send({ memberIds })
    expect(res.status).toBe(422)
    expect(res.body.error).toEqual({
      code: 'LEVEL_TOO_LOW',
      message: 'Gimnasio del Cerro exige un nivel promedio de 40; tu alineación tiene
10',
    })
  })

  it('una alineación con un miembro ajeno o repetido es un 422', async () => {
    const { agent } = await loggedIn()
    const [a, b] = await teamOf(agent, 10)
    const res = await agent.post('/api/gyms/caribe/challenge').send({ memberIds: [a, b,
b] })
    expect(res.status).toBe(422)
    expect(res.body.error.code).toBe('LINEUP_INVALID')
  })
```

```
})
```

These are integration tests: each one goes through the whole route, from request to response, with the validation, the session, the data, and the error handler. Two more test individual pieces: the **`Store`**, with twenty changes at once and a corrupt file, and the errors for a malformed or oversized body.

apps/api/test/store.test.ts

```
import { mkdtemp, readFile, writeFile } from 'node:fs/promises'
import { tmpdir } from 'node:os'
import path from 'node:path'
import { describe, expect, it } from 'vitest'
import { Store } from '../src/store.ts'

describe('Store', () => {
  it('guarda cambios simultáneos sin perder ninguno', async () => {
    const file = path.join(await mkdtemp(path.join(tmpdir(), 'store-')), 'db.json')
    const store = await Store.fromFile(file)
    await Promise.all(
      Array.from({ length: 20 }, (_, i) =>
        store.update((data) => {
          data.badges[`u${i}`] = ['caribe']
        }),
      ),
    )
    const saved = JSON.parse(await readFile(file, 'utf8')) as { badges: Record<string,
string[]> }
    expect(Object.keys(saved.badges)).toHaveLength(20)
  })

  it('un archivo corrupto detiene el arranque en vez de reemplazarse', async () => {
    const file = path.join(await mkdtemp(path.join(tmpdir(), 'store-')), 'db.json')
    await writeFile(file, '{ esto no es JSON')
    await expect(Store.fromFile(file)).rejects.toThrow(SyntaxError)
    expect(await readFile(file, 'utf8')).toBe('{ esto no es JSON')
  })
})
```

apps/api/test/errors.test.ts

```
import request from 'supertest'
import { describe, expect, it } from 'vitest'
import { testApp } from './helpers.ts'

describe('errores del pedido', () => {
  it('un JSON mal formado es un 400, no un 500', async () => {
    const { app } = await testApp()
    const res = await request(app)
      .post('/api/auth/login')
```

```
      .set('Content-Type', 'application/json')
      .send('{"email":')
    expect(res.status).toBe(400)
    expect(res.body.error.code).toBe('BAD_REQUEST')
  })

  it('un cuerpo de más de 10 kB es un 413', async () => {
    const { app } = await testApp()
    const res = await request(app)
      .post('/api/auth/login')
      .send({ email: 'a'.repeat(11_000), password: 'x' })
    expect(res.status).toBe(413)
    expect(res.body.error.code).toBe('PAYLOAD_TOO_LARGE')
  })
})
```

Install, download, and test

```
pnpm install
pnpm test
```

Vitest should finish with **`Tests 28 passed (28)`**: the four from the contract and the API's.

To start the API with the real Pokémon, download them once (it takes about a minute) and create the **`.env`** from the example:

```
pnpm --filter @dex506/api seed
cp apps/api/.env.example apps/api/.env
pnpm --filter @dex506/api dev
```

On Windows, in PowerShell, the second command is **`Copy-Item apps/api/.env.example apps/api/.env`**. **`--filter`** runs the script only in the **`@dex506/api`** package. The API announces **`API de Dex 506 en http://localhost:3001 (151 Pokémon, datos: pokeapi)`** and keeps the terminal busy. Open another terminal, in the same folder, and request two URLs with **curl**, a terminal program that makes HTTP requests (on Windows, type **`curl.exe`**: in PowerShell, **`curl`** is a different command):

```
curl http://localhost:3001/api/health
curl "http://localhost:3001/api/pokemon?q=pika"
```

The first responds **`{"ok":true}`**, and the second, a page with a single Pokémon. With the made-up creatures (**`DEX_DATA=fixtures`**, below), search for **`q=lag`** instead of **`q=pika`**. To stop the API, go back to its terminal and press **`Ctrl+C`**: the same goes for every server in this guide. To work offline, change **`DEX_DATA=pokeapi`** to **`DEX_DATA=fixtures`** in **`apps/api/.env`**: the API uses the made-up creatures.

> **The usual mistake:** the API does not start and says "No hay datos de PokéAPI" (there is no PokéAPI data). You have not run the **seed** yet, or the **.env** says **DEX_DATA=pokeapi** without the downloaded data. Run the **seed** or use **DEX_DATA=fixtures**.

> **The usual mistake:** the API says "No se pudo usar el puerto 3001" (port 3001 could not be used) and exits. Another program is using that port, almost always an API left running in another terminal: stop it with **Ctrl+C**. The **listen** in **server.ts** checks for that error so it can say so instead of staying silent.

```
git add .
git commit -m "paso 04: la API"
```

Download the code for step 04. The zip does not include the PokéAPI data: after unzipping it, run the **seed** or use **DEX_DATA=fixtures**.

Step 05: The Frontend Foundation

In this step you will put together the web app: Vite, Tailwind, the router, the data client, the languages, the theme, and the layout all the screens share.

All the web app files go in **apps/web/**. This time the Vite template from part 1 is not used: the web app is one more package in the monorepo, and each file is written by hand so you understand what it does.

The package and its configuration

apps/web/package.json

```
{
  "name": "@dex506/web",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "typecheck": "tsc -b"
  },
  "dependencies": {
    "@dex506/contracts": "workspace:*",
    "@tanstack/react-query": "5.108.0",
    "axios": "1.21.1",
    "react": "19.3.1",
```

```
    "react-dom": "19.3.1",
    "react-router": "8.5.1"
  },
  "devDependencies": {
    "@tailwindcss/vite": "4.3.5",
    "@testing-library/jest-dom": "7.0.1",
    "@testing-library/react": "16.3.3",
    "@testing-library/user-event": "14.6.12",
    "@types/react": "19.3.0",
    "@types/react-dom": "19.3.0",
    "@vitejs/plugin-react": "6.3.0",
    "jsdom": "31.0.1",
    "msw": "3.0.7",
    "tailwindcss": "4.3.5",
    "vite": "8.3.8"
  }
}
```

These are the libraries from parts 1 to 3: React, React Router, TanStack Query, axios, and Tailwind, and for the tests, Vitest, Testing Library, jsdom, and MSW. **`typecheck`** is **`tsc -b`**, which checks the web app's two TypeScript projects (see below).

`index.html` is the page the browser loads, as in "Your First Project with Vite":

apps/web/index.html

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/dex-506.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Dex 506</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

And its icon, a custom SVG:

apps/web/public/dex-506.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 32 32"><rect width="32" height="32" rx="8" fill="#0f766e"/><circle cx="16" cy="16" r="9" fill="none" stroke="#fff" stroke-width="3"/><circle cx="16" cy="16" r="3" fill="#fb7185"/></svg>
```

`vite.config.ts` adds the Tailwind plugin from part 1 and a **`proxy`**: in development, Vite forwards everything that starts with **`/api`** to the API. To the browser, the web app and the API come from the same origin, and the session cookie travels without configuring CORS, as you saw in "Architecture, Quality, and Security".

apps/web/vite.config.ts

```
import tailwindcss from '@tailwindcss/vite'
import react from '@vitejs/plugin-react'
import { defineConfig } from 'vite'

export default defineConfig({
  plugins: [react(), tailwindcss()],
  server: {
    // In development, /api goes to the Express API. To the browser everything comes
from the same
    // origin (localhost:5173), so the session cookie travels without configuring CORS.
    proxy: {
      '/api': 'http://localhost:3001',
    },
  },
})
```

apps/web/vitest.config.ts

```
import react from '@vitejs/plugin-react'
import { defineProject } from 'vitest/config'

export default defineProject({
  plugins: [react()],
  test: {
    name: 'web',
    environment: 'jsdom',
    setupFiles: ['./src/test/setup.ts'],
  },
})
```

The web app has two TypeScript projects, like the Vite template: one for the code that runs in the browser and another for the configuration files, which run in Node.

apps/web/tsconfig.json

```
{
  "files": [],
  "references": [{ "path": "./tsconfig.app.json" }, { "path": "./tsconfig.node.json" }]
}
```

apps/web/tsconfig.app.json

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
    "lib": ["ES2023", "DOM", "DOM.Iterable"],
    "types": ["vite/client"],
    "jsx": "react-jsx"
  },
  "include": ["src"]
}
```

apps/web/tsconfig.node.json

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.node.tsbuildinfo",
    "lib": ["ES2023"],
    "types": ["node"],
    "module": "nodenext",
    "moduleResolution": "nodenext"
  },
  "include": ["vite.config.ts", "vitest.config.ts"]
}
```

`references` ties the two together, and **`tsc -b`** (*build mode*) checks them together. **`jsx: "react-jsx"`** is what lets you write JSX, and **`lib: ["DOM"]`** lets you use **`document`** and **`window`**.

The styles

apps/web/src/index.css

```
@import 'tailwindcss';

/* Dark mode is decided by ThemeContext with data-theme on <html>, not by the system
preference. */
@custom-variant dark (&:where([data-theme='dark'], [data-theme='dark'] *));

@theme {
  --font-sans: system-ui, -apple-system, 'Segoe UI', Roboto, 'Helvetica Neue', Arial,
sans-serif;

  /* The brand: a cloud-forest teal and a Pacific-sunset coral. */
  --color-brand-50: #f0fdfa;
  --color-brand-100: #ccfbf1;
  --color-brand-600: #0d9488;
```

```
  --color-brand-700: #0f766e;
  --color-brand-800: #115e59;
  --color-brand-900: #134e4a;
  --color-accent-400: #fb7185;
  --color-accent-600: #e11d48;

  /* One color per type, Dex 506's own. They are light backgrounds: the text on top is
always dark. */
  --color-type-normal: #e7e5e4;
  --color-type-fighting: #fecaca;
  --color-type-flying: #c7d2fe;
  --color-type-poison: #e9d5ff;
  --color-type-ground: #fde68a;
  --color-type-rock: #d6d3d1;
  --color-type-bug: #d9f99d;
  --color-type-ghost: #ddd6fe;
  --color-type-steel: #cbd5e1;
  --color-type-fire: #fed7aa;
  --color-type-water: #bae6fd;
  --color-type-grass: #bbf7d0;
  --color-type-electric: #fef08a;
  --color-type-psychic: #fbcfe8;
  --color-type-ice: #a5f3fc;
  --color-type-dragon: #c4b5fd;
  --color-type-dark: #a8a29e;
  --color-type-fairy: #fce7f3;
}

body {
  @apply bg-stone-50 font-sans text-stone-900 dark:bg-stone-950 dark:text-stone-100;
}
```

- **`@theme`** defines the custom colors, as in "Styles": the brand, a turquoise and a coral, and a light color for each Pokémon type.
- **`@custom-variant dark`** changes how **`dark:`** works: the dark theme is not decided by the system preference, but by a **`data-theme`** attribute on **`<html>`**, which the **`ThemeProvider`** below sets. That way the person chooses the theme in the app.
- **`@apply`** applies Tailwind classes inside the CSS, for the background and text colors.

The entry point

apps/web/src/main.tsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { createBrowserRouter } from 'react-router'
import { RouterProvider } from 'react-router/dom'
```

```
import { Providers } from './app/Providers.tsx'
import { createQueryClient } from './app/queryClient.ts'
import { routes } from './app/routes.tsx'
import './index.css'

const router = createBrowserRouter(routes)
const queryClient = createQueryClient()

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <Providers queryClient={queryClient}>
      <RouterProvider router={router} />
    </Providers>
  </StrictMode>,
)
```

The router and the **`QueryClient`** are created once, outside of rendering, as in "Global State".

apps/web/src/app/queryClient.ts

```
import { QueryClient } from '@tanstack/react-query'

export function createQueryClient(): QueryClient {
  return new QueryClient({
    defaultOptions: {
      queries: { staleTime: 30_000, retry: 1 },
    },
  })
}
```

Data is considered fresh for 30 seconds, and a query that fails is retried once, not the three times from "TanStack Query".

apps/web/src/app/Providers.tsx

```
import { QueryClientProvider, type QueryClient } from '@tanstack/react-query'
import type { ReactNode } from 'react'
import { I18nProvider } from '../i18n/I18nProvider.tsx'
import { ThemeProvider } from '../shared/theme/ThemeProvider.tsx'

/** Everything the app needs around it: server data, language, and theme. */
export function Providers({
  queryClient,
  children,
}: {
  queryClient: QueryClient
  children: ReactNode
}) {
```

```
  return (
    <QueryClientProvider client={queryClient}>
      <I18nProvider>
        <ThemeProvider>{children}</ThemeProvider>
      </I18nProvider>
    </QueryClientProvider>
  )
}
```

`Providers` brings all the providers together in one component. The tests use it the same way the app does, and every new screen that needs a provider adds it here.

The languages

The web app is in Spanish and English, with the dictionary from "Internationalization". All the files in this block go in **`apps/web/src/i18n/`**.

apps/web/src/i18n/es.ts

```
export const es = {
  appName: 'Dex 506',
  nav: { pokedex: 'Pokédex', team: 'Equipo', gyms: 'Gimnasios', main: 'Navegación
principal' },
  theme: { toLight: 'Cambiar a tema claro', toDark: 'Cambiar a tema oscuro' },
  language: { label: 'Idioma' },
  home: {
    title: 'Tu Pokédex de entrenador',
    body: 'Explora los Pokémon, arma tu equipo de seis y reta a los gimnasios de Costa
Rica.',
  },
  notFound: { title: 'Esta página no existe', back: 'Volver al inicio' },
  routeError: { title: 'Algo salió mal', retry: 'Recargar la página' },
  footer: {
    notice:
      'Proyecto educativo de fans, sin fines de lucro. No está afiliado, patrocinado ni
avalado por Nintendo, Game Freak ni The Pokémon Company. Pokémon y sus nombres son
marcas de sus respectivos dueños. Datos servidos por PokéAPI.',
  },
}

export type Messages = typeof es
```

`Messages` is the type of the Spanish dictionary: **`typeof es`**. The English one has to have the same shape:

apps/web/src/i18n/en.ts

```
import type { Messages } from './es.ts'
```

```ts
// `satisfies Messages`: if the translation is missing a key or has an extra one, it does not compile.
export const en = {
  appName: 'Dex 506',
  nav: { pokedex: 'Pokédex', team: 'Team', gyms: 'Gyms', main: 'Main navigation' },
  theme: { toLight: 'Switch to light theme', toDark: 'Switch to dark theme' },
  language: { label: 'Language' },
  home: {
    title: "Your trainer's Pokédex",
    body: 'Browse the Pokémon, build your team of six, and challenge the gyms of Costa Rica.',
  },
  notFound: { title: 'This page does not exist', back: 'Back to the start' },
  routeError: { title: 'Something went wrong', retry: 'Reload the page' },
  footer: {
    notice:
      'Non-profit educational fan project. Not affiliated with, sponsored or endorsed by Nintendo, Game Freak or The Pokémon Company. Pokémon and their names are trademarks of their respective owners. Data provided by PokéAPI.',
  },
} satisfies Messages
```

`satisfies Messages` makes a missing or extra key fail to compile: every new screen adds its texts to both languages.

apps/web/src/i18n/i18nContext.ts

```ts
import { createContext, use } from 'react'
import type { Messages } from './es.ts'

export type Locale = 'es' | 'en'
export type I18nValue = { locale: Locale; setLocale: (locale: Locale) => void; t: Messages }

export const I18nContext = createContext<I18nValue | null>(null)

/** The texts in the current language. Outside I18nProvider it is a programming error. */
export function useI18n(): I18nValue {
  const value = use(I18nContext)
  if (!value) throw new Error('useI18n tiene que usarse dentro de I18nProvider')
  return value
}
```

apps/web/src/i18n/I18nProvider.tsx

```ts
import { useEffect, useState, type ReactNode } from 'react'
import { en } from './en.ts'
import { es, type Messages } from './es.ts'
```

```
import { I18nContext, type Locale } from './i18nContext.ts'

const MESSAGES: Record<Locale, Messages> = { es, en }
const STORAGE_KEY = 'dex506.locale'

function initialLocale(): Locale {
  const saved = localStorage.getItem(STORAGE_KEY)
  return saved === 'en' || saved === 'es' ? saved : 'es'
}

export function I18nProvider({ children }: { children: ReactNode }) {
  const [locale, setLocale] = useState<Locale>(initialLocale)

  useEffect(() => {
    localStorage.setItem(STORAGE_KEY, locale)
    document.documentElement.lang = locale
  }, [locale])

  return <I18nContext value={{ locale, setLocale, t: MESSAGES[locale] }}>{children}
</I18nContext>
}
```

- **`useState<Locale>(initialLocale)`** passes the function without calling it: React uses its result only on the first render, to read the saved language (in development, with **`StrictMode`**, it calls it twice). This is **lazy initialization**.
- The effect synchronizes with something outside React: **`localStorage`** and the **`lang`** attribute on **`<html>`**, which screen readers use to pronounce in the right language.
- The Context and its hook go in a separate file from the provider: that way each file exports either components or functions, which is what Vite's hot reload requires.

The theme

With the same shape, the light or dark theme. Both files go in **`apps/web/src/shared/theme/`**:

apps/web/src/shared/theme/themeContext.ts

```
import { createContext, use } from 'react'

export type Theme = 'light' | 'dark'
export type ThemeValue = { theme: Theme; toggleTheme: () => void }

export const ThemeContext = createContext<ThemeValue | null>(null)

export function useTheme(): ThemeValue {
  const value = use(ThemeContext)
  if (!value) throw new Error('useTheme tiene que usarse dentro de ThemeProvider')
  return value
```

```
}
```

apps/web/src/shared/theme/ThemeProvider.tsx

```
import { useEffect, useState, type ReactNode } from 'react'
import { ThemeContext, type Theme } from './themeContext.ts'

const STORAGE_KEY = 'dex506.theme'

function initialTheme(): Theme {
  const saved = localStorage.getItem(STORAGE_KEY)
  if (saved === 'light' || saved === 'dark') return saved
  return window.matchMedia?.('(prefers-color-scheme: dark)').matches ? 'dark' : 'light'
}

export function ThemeProvider({ children }: { children: ReactNode }) {
  const [theme, setTheme] = useState<Theme>(initialTheme)

  // Syncing with something outside React (the <html> attribute and localStorage): an
effect.
  useEffect(() => {
    document.documentElement.dataset.theme = theme
    localStorage.setItem(STORAGE_KEY, theme)
  }, [theme])

  const toggleTheme = () => setTheme((current) => (current === 'dark' ? 'light' :
'dark'))
  return <ThemeContext value={{ theme, toggleTheme }}>{children}</ThemeContext>
}
```

If there is no saved theme, the system preference is used, with **`matchMedia`**.

The API client

apps/web/src/shared/api/client.ts

```
import axios, { AxiosError } from 'axios'
import { ApiErrorSchema, type ApiError } from '@dex506/contracts'

/** The app-wide axios instance: the base for every route and the session cookie on
every request. */
export const api = axios.create({
  baseURL: '/api',
  withCredentials: true,
  timeout: 10_000,
})

let onUnauthorized: (() => void) | null = null
```

```ts
/** AuthProvider registers it: what to do when the API answers 401 (the session expired). */
export function setUnauthorizedHandler(handler: (() => void) | null): void {
  onUnauthorized = handler
}

api.interceptors.response.use(
  (response) => response,
  (error: unknown) => {
    const url = error instanceof AxiosError ? (error.config?.url ?? '') : ''
    // A 401 from /auth/me or /auth/login is not an expired session: it is "no session" or "wrong password".
    if (
      error instanceof AxiosError &&
      error.response?.status === 401 &&
      !url.startsWith('/auth/')
    ) {
      onUnauthorized?.()
    }
    return Promise.reject(error)
  },
)

/** Any request error, in the ApiError shape: that way the interface always knows what to show. */
export function toApiError(error: unknown): ApiError {
  if (error instanceof AxiosError) {
    const parsed = ApiErrorSchema.safeParse(error.response?.data)
    if (parsed.success) return parsed.data
    if (!error.response)
      return { error: { code: 'NETWORK', message: 'No hay conexión con el servidor' } }
  }
  return { error: { code: 'UNKNOWN', message: 'Error inesperado' } }
}
```

- **`baseURL: '/api'`**: every request goes to the same origin, and the Vite proxy takes it to the API.
- **`withCredentials: true`** is what the web app would need, on its side, if one day the API lived on another origin; the API would also have to allow it through CORS, and the cookie would need **`SameSite=None`**. With the proxy, requests are same-origin and the browser already sends the cookie.
- The interceptor gives notice when the API responds 401, meaning the session expired, except under **`/auth/`**, where a 401 means "there is no session" or "wrong password". Who reacts to that notice is decided in step 06, with **`setUnauthorizedHandler`**.
- **`toApiError`** turns any error into an **`ApiError`**: the API's own if it has that shape, a network error if there was no response, or a generic one. That way the interface always knows what to show.

The router and the layout

apps/web/src/app/routes.tsx

```
import type { RouteObject } from 'react-router'
import { HomePage } from './HomePage.tsx'
import { Layout } from './Layout.tsx'
import { NotFoundPage } from './NotFoundPage.tsx'
import { RouteError } from './RouteError.tsx'

/** Routes as data: the app's router and the tests' router use the same list. */
export const routes: RouteObject[] = [
  {
    path: '/',
    element: <Layout />,
    errorElement: <RouteError />,
    children: [
      { index: true, element: <HomePage /> },
      { path: '*', element: <NotFoundPage /> },
    ],
  },
]
```

The routes are data, in the data mode from "Routing with React Router": the app and the tests use the same list. **`index: true`** is the **index route**, the one shown at the parent's exact URL, **`/`**.

apps/web/src/app/Layout.tsx

```
import { NavLink, Outlet } from 'react-router'
import { useI18n, type Locale } from '../i18n/i18nContext.ts'
import { useTheme } from '../shared/theme/themeContext.ts'

const linkClass = ({ isActive }: { isActive: boolean }) =>
  `rounded-md px-3 py-2 text-sm font-medium ${
    isActive
      ? 'bg-brand-700 text-white'
      : 'text-stone-700 hover:bg-brand-100 dark:text-stone-200 dark:hover:bg-brand-900'
  }`

export function Layout() {
  const { t, locale, setLocale } = useI18n()
  const { theme, toggleTheme } = useTheme()

  return (
    <div className="flex min-h-screen flex-col">
      <header className="border-b border-stone-200 bg-white dark:border-stone-800
dark:bg-stone-900">
        <div className="mx-auto flex max-w-5xl flex-wrap items-center gap-3 px-4 py-3">
```

```
          <NavLink to="/" className="mr-auto text-lg font-bold text-brand-700 dark:text-
brand-100">
            {t.appName}
          </NavLink>
          <nav aria-label={t.nav.main} className="flex gap-1">
            <NavLink to="/pokedex" className={linkClass}>
              {t.nav.pokedex}
            </NavLink>
          </nav>
          {/* Explicit label (htmlFor + id): wrapped in the <label>, the <select>'s name
would
              include the selected option. */}
          <label htmlFor="language" className="sr-only">
            {t.language.label}
          </label>
          <select
            id="language"
            value={locale}
            onChange={(event) => setLocale(event.target.value as Locale)}
            className="rounded-md border border-stone-300 bg-white px-2 py-1
dark:border-stone-700 dark:bg-stone-800"
          >
            <option value="es">ES</option>
            <option value="en">EN</option>
          </select>
          <button
            type="button"
            onClick={toggleTheme}
            aria-label={theme === 'dark' ? t.theme.toLight : t.theme.toDark}
            className="rounded-md border border-stone-300 px-2 py-1 text-sm dark:border-
stone-700"
          >
            {theme === 'dark' ? '☀' : '☾'}
          </button>
        </div>
      </header>
      <main className="mx-auto w-full max-w-5xl flex-1 px-4 py-6">
        <Outlet />
      </main>
      <footer className="border-t border-stone-200 px-4 py-4 text-center text-xs text-
stone-600 dark:border-stone-800 dark:text-stone-400">
        <p className="mx-auto max-w-3xl">{t.footer.notice}</p>
      </footer>
    </div>
  )
}
```

The layout has the app's name, the navigation, the language selector, the theme button, and the intellectual property notice in the footer. **`sr-only`** is a Tailwind class that hides text on screen but not from

screen readers: the **<select>** has its label "Idioma" (language) even though you do not see it. The label is separate, connected with **htmlFor** and **id**: if it wrapped the **<select>**, its name for a screen reader would include the selected option. The theme button shows a symbol, and its **aria-label** says what it does.

apps/web/src/app/HomePage.tsx

```
import { useI18n } from '../i18n/i18nContext.ts'

export function HomePage() {
  const { t } = useI18n()
  return (
    <section className="space-y-2">
      <h1 className="text-3xl font-bold">{t.home.title}</h1>
      <p className="text-stone-700 dark:text-stone-300">{t.home.body}</p>
    </section>
  )
}
```

apps/web/src/app/NotFoundPage.tsx

```
import { Link } from 'react-router'
import { useI18n } from '../i18n/i18nContext.ts'

export function NotFoundPage() {
  const { t } = useI18n()
  return (
    <section className="space-y-3">
      <h1 className="text-2xl font-bold">{t.notFound.title}</h1>
      <Link to="/" className="text-brand-700 underline dark:text-brand-100">
        {t.notFound.back}
      </Link>
    </section>
  )
}
```

apps/web/src/app/RouteError.tsx

```
import { useI18n } from '../i18n/i18nContext.ts'

/** What you see if a route throws while rendering: instead of a blank screen. */
export function RouteError() {
  const { t } = useI18n()
  return (
    <section role="alert" className="space-y-3 p-6">
      <h1 className="text-2xl font-bold">{t.routeError.title}</h1>
      <button type="button" onClick={() => window.location.reload()}
className="underline">
        {t.routeError.retry}
      </button>
```

```
    </section>
  )
}
```

`RouteError` is the root route's **`errorElement`**: if something fails while rendering, you see this instead of a blank screen.

The web app's tests

The test support files go in **`apps/web/src/test/`**.

apps/web/src/test/setup.ts

```ts
import '@testing-library/jest-dom/vitest'
import { cleanup } from '@testing-library/react'
import { afterAll, afterEach, beforeAll } from 'vitest'
import { server } from './server.ts'

// A request without a handler is an error (in MSW 3 the option is called
onUnhandledFrame): that way
// no test depends on a real network by accident.
beforeAll(() => server.listen({ onUnhandledFrame: 'error' }))
afterEach(() => {
  cleanup()
  server.resetHandlers()
  localStorage.clear()
})
afterAll(() => server.close())
```

apps/web/src/test/server.ts

```ts
import { setupServer } from 'msw/node'

/** The tests' MSW server: each test adds the handlers it needs with server.use. */
export const server = setupServer()
```

The MSW server starts with no handlers: each test adds the ones it needs with **`server.use`**. Also, each test leaves **`localStorage`** empty, so the language or theme from one does not affect the next.

apps/web/src/test/renderApp.tsx

```tsx
import { QueryClient } from '@tanstack/react-query'
import { render } from '@testing-library/react'
import { createMemoryRouter, RouterProvider } from 'react-router'
import { Providers } from '../app/Providers.tsx'
import { routes } from '../app/routes.tsx'
```

```
/** The whole app at one route, with an in-memory router and a fresh data cache for
every test. */
export function renderApp(path = '/') {
  const router = createMemoryRouter(routes, { initialEntries: [path] })
  const queryClient = new QueryClient({ defaultOptions: { queries: { retry: false } } })
  const result = render(
    <Providers queryClient={queryClient}>
      <RouterProvider router={router} />
    </Providers>,
  )
  return { ...result, router, queryClient }
}
```

renderApp mounts the whole app at a URL, with **createMemoryRouter**: a router that keeps the URL in memory instead of using the browser's address bar, which does not exist in jsdom. Each test gets a new **QueryClient**, with no retries.

apps/web/src/app/Layout.test.tsx

```
import { screen } from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import { describe, expect, it } from 'vitest'
import { renderApp } from '../test/renderApp.tsx'

describe('Layout', () => {
  it('muestra la navegación y el aviso de propiedad intelectual', () => {
    renderApp('/')
    expect(screen.getByRole('navigation', { name: 'Navegación principal'
})).toBeInTheDocument()
    expect(
      screen.getByText(/No está afiliado, patrocinado ni avalado por Nintendo/),
    ).toBeInTheDocument()
  })

  it('cambia de idioma y lo recuerda', async () => {
    renderApp('/')
    await userEvent.selectOptions(screen.getByRole('combobox', { name: 'Idioma' }),
'en')
    expect(screen.getByRole('heading', { name: "Your trainer's Pokédex"
})).toBeInTheDocument()
    expect(localStorage.getItem('dex506.locale')).toBe('en')
  })

  it('cambia el tema en el atributo de <html>', async () => {
    renderApp('/')
    const before = document.documentElement.dataset.theme
    await userEvent.click(screen.getByRole('button', { name: /Cambiar a tema/ }))
    expect(document.documentElement.dataset.theme).not.toBe(before)
  })
```

```
  it('una ruta que no existe muestra la página 404 dentro del layout', () => {
    renderApp('/no-existe')
    expect(screen.getByRole('heading', { name: 'Esta página no existe'
})).toBeInTheDocument()
    expect(screen.getByRole('navigation', { name: 'Navegación principal'
})).toBeInTheDocument()
  })
})
```

The changes at the root

The linter adds the React and accessibility (**`jsx-a11y`**) rules, and now checks the web app too. From here on, every step ends with **`pnpm test`**; if you want to get ahead of the CI in step 12, also run **`pnpm lint`** and **`pnpm typecheck`**, which step 11 bundles with the rest. **`exhaustive-deps`** is set as an error: a missing dependency in an effect does not get through.

.oxlintrc.json

```
--- a/.oxlintrc.json
+++ b/.oxlintrc.json
@@ -1,8 +1,11 @@
 {
   "$schema": "./node_modules/oxlint/configuration_schema.json",
-  "plugins": ["typescript", "oxc", "import"],
+  "plugins": ["typescript", "oxc", "import", "react", "jsx-a11y"],
   "ignorePatterns": ["**/dist/**", "**/node_modules/**", "**/.cache/**",
"**/.data/**"],
   "rules": {
-    "no-console": ["warn", { "allow": ["warn", "error", "info"] }]
+    "no-console": ["warn", { "allow": ["warn", "error", "info"] }],
+    "react/rules-of-hooks": "error",
+    "react/exhaustive-deps": "error",
+    "react/only-export-components": ["warn", { "allowConstantExport": true }]
   }
 }
```

pnpm 12 does not install a version published less than a day ago (1440 minutes, its default **`minimumReleaseAge`**), to leave time to detect a malicious package before it reaches your machine (pnpm, *Dependency resolution settings*). The versions listed under **`minimumReleaseAgeExclude`** in **`pnpm-workspace.yaml`** came out the same day I built the app, and the exception allowed installing them. Those packages are now more than a day old: the exception is no longer needed, but it does no harm either.

pnpm-workspace.yaml

```
--- a/pnpm-workspace.yaml
+++ b/pnpm-workspace.yaml
```

```
@@ -1,3 +1,7 @@
 packages:
   - 'apps/*'
   - 'packages/*'
+minimumReleaseAgeExclude:
+  - '@tanstack/query-core@5.108.0'
+  - '@tanstack/react-query@5.108.0'
+  - msw@3.0.7
```

Install, test, and open

```
pnpm install
pnpm test
```

Vitest should finish with **`Tests 32 passed (32)`**: the four new tests are the layout's.

```
pnpm dev
```

`pnpm dev` starts the API and the web app at the same time, in the same terminal (**`Ctrl+C`** stops both). Vite announces the web app's address, **`http://localhost:5173`**: open it and you will see the home page, with the navigation, the language selector, the theme button, and the notice in the footer. Change the language and the theme, reload the page, and they are still there.

```
git add .
git commit -m "paso 05: la base del frontend"
```

Download the code for step 05.

Step 06: Sign-In

In this step you will build the first screen: the login, the session that remembers it, and the routes that require it.

The new files in this step go in **`apps/web/src/features/auth/`**, the folder for the sign-in feature, as in "Organizing by feature".

Talking to the API

apps/web/src/features/auth/api.ts

```
import { AxiosError } from 'axios'
```

```ts
import { UserSchema, type LoginInput, type User } from '@dex506/contracts'
import { api } from '../../shared/api/client.ts'

/** The trainer from the session cookie, or null if there is no session. */
export async function fetchMe(): Promise<User | null> {
  try {
    const { data } = await api.get('/auth/me')
    return UserSchema.parse(data)
  } catch (error) {
    if (error instanceof AxiosError && error.response?.status === 401) return null
    throw error
  }
}

export async function login(input: LoginInput): Promise<User> {
  const { data } = await api.post('/auth/login', input)
  return UserSchema.parse(data)
}

export async function logout(): Promise<void> {
  await api.post('/auth/logout')
}
```

Each response is validated with the contract's schema before it is used: if the API and the web app ended up on different versions, the error shows up here, with a clear message. **`fetchMe`** turns the 401 into **`null`**: not having a session is not an error.

The session

The cookie is **`httpOnly`**, so the page's JavaScript cannot read it. To know whether someone is signed in, the web app asks the API when it opens: this is session **rehydration**.

apps/web/src/features/auth/authContext.ts

```ts
import { createContext, use } from 'react'
import type { User } from '@dex506/contracts'

export type AuthValue = {
  /** undefined while asking the API whether there is a session; null if there is none.
*/
  user: User | null | undefined
  /** true if the API could not say whether there is a session (not a 401: a 500 or no
network). */
  unavailable: boolean
  retry: () => void
  signIn: (user: User) => void
  signOut: () => Promise<void>
}
```

```
/** The session's key in the TanStack Query cache. */
export const ME_KEY = ['auth', 'me'] as const

export const AuthContext = createContext<AuthValue | null>(null)

export function useAuth(): AuthValue {
  const value = use(AuthContext)
  if (!value) throw new Error('useAuth tiene que usarse dentro de AuthProvider')
  return value
}
```

`user` has three possible values, and each one means something: **`undefined`** while it is asking, **`null`** if there is no session, and the user if there is one. **`unavailable`** is **`true`** if the API could not answer (a 500, or no network): that is not the same as not having a session.

apps/web/src/features/auth/AuthProvider.tsx

```
import { useQuery, useQueryClient } from '@tanstack/react-query'
import { useEffect, type ReactNode } from 'react'
import type { User } from '@dex506/contracts'
import { setUnauthorizedHandler } from '../../shared/api/client.ts'
import { fetchMe, logout } from './api.ts'
import { AuthContext, ME_KEY } from './authContext.ts'

/**
 * The app's session. When the page loads it asks the API who the cookie's user is
(rehydrating the
 * session): the cookie is httpOnly, so JavaScript cannot read it.
 */
export function AuthProvider({ children }: { children: ReactNode }) {
  const queryClient = useQueryClient()
  const me = useQuery({ queryKey: ME_KEY, queryFn: fetchMe, staleTime: Infinity })
  const user = me.data

  // If any request answers 401, the session expired: the user and their data are
forgotten.
  useEffect(() => {
    setUnauthorizedHandler(() => {
      queryClient.setQueryData(ME_KEY, null)
      queryClient.removeQueries({ predicate: (query) => query.queryKey[0] !== 'auth' })
    })
    return () => setUnauthorizedHandler(null)
  }, [queryClient])

  const signIn = (signedIn: User) => queryClient.setQueryData(ME_KEY, signedIn)
  const signOut = async () => {
    await logout()
    queryClient.setQueryData(ME_KEY, null)
```

```
    // The previous trainer's data must not stay in the cache. clear() is not used: it
would also
    // delete the session, and the active query would fetch it again.
    queryClient.removeQueries({ predicate: (query) => query.queryKey[0] !== 'auth' })
  }

  const retry = () => void me.refetch()

  return (
    <AuthContext value={{ user, unavailable: me.isError, retry, signIn, signOut }}>
      {children}
    </AuthContext>
  )
}
```

- The session is server data, so it lives in TanStack Query, with the key **`ME_KEY`**. **`staleTime: Infinity`** keeps it from being fetched again on its own: it changes only when someone signs in or out.
- **`signIn`** and **`signOut`** write directly to the cache with **`setQueryData`**, with no extra request.
- The effect registers what to do when the API responds 401 in the middle of the session: forget the user and remove their data from the cache, as when signing out, so whoever signs in next does not see the previous trainer's data. Its cleanup unregisters it.
- **`retry`** asks again, with **`refetch`**.
- When signing out, the previous trainer's data is removed from the cache. **`clear()`** is not used: it would also remove the session, and the active query would fetch it again right away.

The routes that require a session

apps/web/src/features/auth/RequireAuth.tsx

```
import { Navigate, Outlet, useLocation } from 'react-router'
import { useI18n } from '../../i18n/i18nContext.ts'
import { useAuth } from './authContext.ts'

/** A layout route: its child routes are shown only with a session; without one, to the
login. */
export function RequireAuth() {
  const { user, unavailable, retry } = useAuth()
  const location = useLocation()
  const { t } = useI18n()

  // Still no answer about the session: an alert with a retry if the API could not
answer, and an
  // <output> (role status) while asking.
  if (user === undefined && unavailable)
    return (
      <div role="alert" className="space-y-2">
```

```
        <p>{t.auth.unavailable}</p>
        <button type="button" onClick={retry} className="underline">
          {t.auth.retry}
        </button>
      </div>
    )
  if (user === undefined) return <output className="block">{t.auth.checking}</output>
  if (user === null)
    return <Navigate to="/login" replace state={{ from: location.pathname +
location.search }} />
  return <Outlet />
}
```

RequireAuth is a layout route with no URL of its own: it wraps its child routes. While it is asking, it shows an **<output>**, the HTML element for a result or a status (its role is **status**; since it appears together with its text, a screen reader may not announce it). If the API could not answer, it shows the problem with a button to retry, instead of waiting forever. With no session, **<Navigate>** takes you to the login with **replace** (the login replaces the history entry, and "back" does not return to the protected page) and saves in **state** the URL you wanted to go to, with its filters (**pathname** plus **search**).

The login

apps/web/src/features/auth/LoginPage.tsx

```
import { useActionState } from 'react'
import { useFormStatus } from 'react-dom'
import { Navigate, useLocation } from 'react-router'
import { LoginInputSchema } from '@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { toApiError } from '../../shared/api/client.ts'
import { login } from './api.ts'
import { useAuth } from './authContext.ts'

function SubmitButton({ label, pendingLabel }: { label: string; pendingLabel: string })
{
  // useFormStatus reads the state of the <form> that contains it: no props needed.
  const { pending } = useFormStatus()
  return (
    <button
      type="submit"
      disabled={pending}
      className="w-full rounded-md bg-brand-700 px-4 py-2 font-medium text-white
disabled:opacity-60"
    >
      {pending ? pendingLabel : label}
    </button>
  )
}
```

```tsx
type LoginResult = { error: string | null; email: string }

export function LoginPage() {
  const { user, signIn } = useAuth()
  const { t } = useI18n()
  const from = (useLocation().state as { from?: string } | null)?.from ?? '/'

  const [result, submitAction] = useActionState(
    async (_previous: LoginResult, formData: FormData): Promise<LoginResult> => {
      // React resets the form's uncontrolled fields when the Action ends, even when it fails:
      // the email goes back in the result and comes back as the field's defaultValue.
      const email = String(formData.get('email') ?? '')
      const input = LoginInputSchema.safeParse({
        email,
        password: formData.get('password'),
      })
      if (!input.success) return { error: t.auth.invalidInput, email }
      try {
        // Only the user is stored. The <Navigate> below does the redirect, rendered as soon as
        // there is a user: one declared navigation instead of two competing ones.
        signIn(await login(input.data))
        return { error: null, email }
      } catch (err) {
        const error =
          toApiError(err).error.code === 'INVALID_CREDENTIALS'
            ? t.auth.wrongCredentials
            : t.auth.failed
        return { error, email }
      }
    },
    { error: null, email: '' },
  )

  if (user) return <Navigate to={from} replace />

  return (
    <section className="mx-auto max-w-sm space-y-4">
      <h1 className="text-2xl font-bold">{t.auth.title}</h1>
      <form action={submitAction} className="space-y-3" noValidate>
        <label className="block space-y-1">
          <span className="text-sm font-medium">{t.auth.email}</span>
          <input
            name="email"
            type="email"
            defaultValue={result.email}
            autoComplete="username"
            required
```

```
            className="w-full rounded-md border border-stone-300 px-3 py-2 dark:border-
stone-700 dark:bg-stone-900"
          />
        </label>
        <label className="block space-y-1">
          <span className="text-sm font-medium">{t.auth.password}</span>
          <input
            name="password"
            type="password"
            autoComplete="current-password"
            required
            className="w-full rounded-md border border-stone-300 px-3 py-2 dark:border-
stone-700 dark:bg-stone-900"
          />
        </label>
        {result.error && (
          <p role="alert" className="text-sm text-accent-600">
            {result.error}
          </p>
        )}
        <SubmitButton label={t.auth.submit} pendingLabel={t.auth.submitting} />
      </form>
      <p className="text-sm text-stone-600 dark:text-stone-400">{t.auth.demoHint}</p>
    </section>
  )
}
```

It is the login with an Action from "Forms", with what a real app adds to it:

- the Action validates with the contract's schema before calling the API, and translates the error code into a message in the active language;
- it returns the email, and the field uses it as its **`defaultValue`**: React clears uncontrolled fields when the Action finishes, and this way the email is not lost when the password is wrong;
- the Action only saves the user. The redirect is done by the **`<Navigate>`** above, which renders as soon as there is a user: a declared navigation, instead of calling **`navigate()`** inside the Action and competing with it;
- **`autoComplete`** tells the browser what each field is, so the password manager fills them in;
- **`required`** marks the fields as mandatory for screen readers, and **`noValidate`** leaves validation to the schema.

The routes and the layout

The routes add the login, and the index route moves inside **`RequireAuth`**:

```
apps/web/src/app/routes.tsx
```

```
--- a/apps/web/src/app/routes.tsx
+++ b/apps/web/src/app/routes.tsx
@@ -1,4 +1,6 @@
 import type { RouteObject } from 'react-router'
+import { LoginPage } from '../features/auth/LoginPage.tsx'
+import { RequireAuth } from '../features/auth/RequireAuth.tsx'
 import { HomePage } from './HomePage.tsx'
 import { Layout } from './Layout.tsx'
 import { NotFoundPage } from './NotFoundPage.tsx'
@@ -11,7 +13,12 @@
     element: <Layout />,
     errorElement: <RouteError />,
     children: [
-      { index: true, element: <HomePage /> },
+      { path: 'login', element: <LoginPage /> },
+      {
+        // A route without a path: it only wraps its children with the session check.
+        element: <RequireAuth />,
+        children: [{ index: true, element: <HomePage /> }],
+      },
       { path: '*', element: <NotFoundPage /> },
     ],
   },
```

The layout offers to sign in or out, depending on whether there is a session:

apps/web/src/app/Layout.tsx

```
--- a/apps/web/src/app/Layout.tsx
+++ b/apps/web/src/app/Layout.tsx
@@ -1,4 +1,5 @@
 import { NavLink, Outlet } from 'react-router'
+import { useAuth } from '../features/auth/authContext.ts'
 import { useI18n, type Locale } from '../i18n/i18nContext.ts'
 import { useTheme } from '../shared/theme/themeContext.ts'

@@ -12,6 +13,7 @@
 export function Layout() {
   const { t, locale, setLocale } = useI18n()
   const { theme, toggleTheme } = useTheme()
+  const { user, signOut } = useAuth()

   return (
     <div className="flex min-h-screen flex-col">
@@ -39,6 +41,19 @@
             <option value="es">ES</option>
             <option value="en">EN</option>
           </select>
+          {user ? (
```

```
+            <button
+              type="button"
+              onClick={() => void signOut()}
+              className="rounded-md px-2 py-1 text-sm underline"
+            >
+              {t.auth.signOut}
+            </button>
+          ) : (
+            <NavLink to="/login" className={linkClass}>
+              {t.auth.signIn}
+            </NavLink>
+          )}
            <button
              type="button"
              onClick={toggleTheme}
```

`void signOut()` marks that the promise from **`signOut`** is deliberately not awaited.

The home page greets the trainer:

apps/web/src/app/HomePage.tsx

```
--- a/apps/web/src/app/HomePage.tsx
+++ b/apps/web/src/app/HomePage.tsx
@@ -1,10 +1,13 @@
+import { useAuth } from '../features/auth/authContext.ts'
 import { useI18n } from '../i18n/i18nContext.ts'

 export function HomePage() {
   const { t } = useI18n()
+  const { user } = useAuth()
   return (
     <section className="space-y-2">
       <h1 className="text-3xl font-bold">{t.home.title}</h1>
+      {user && <p className="text-lg">{t.home.greeting(user.trainerName)}</p>}
       <p className="text-stone-700 dark:text-stone-300">{t.home.body}</p>
     </section>
   )
```

And **`Providers`** adds the **`AuthProvider`**. It goes inside the **`QueryClientProvider`**, because it uses TanStack Query:

apps/web/src/app/Providers.tsx

```
--- a/apps/web/src/app/Providers.tsx
+++ b/apps/web/src/app/Providers.tsx
@@ -1,9 +1,10 @@
 import { QueryClientProvider, type QueryClient } from '@tanstack/react-query'
```

```
 import type { ReactNode } from 'react'
+import { AuthProvider } from '../features/auth/AuthProvider.tsx'
 import { I18nProvider } from '../i18n/I18nProvider.tsx'
 import { ThemeProvider } from '../shared/theme/ThemeProvider.tsx'

-/** Everything the app needs around it: server data, language, and theme. */
+/** Everything the app needs around it: server data, language, theme, and session. */
 export function Providers({
   queryClient,
   children,
@@ -14,7 +15,9 @@
   return (
     <QueryClientProvider client={queryClient}>
       <I18nProvider>
-        <ThemeProvider>{children}</ThemeProvider>
+        <ThemeProvider>
+          <AuthProvider>{children}</AuthProvider>
+        </ThemeProvider>
       </I18nProvider>
     </QueryClientProvider>
   )
```

The new texts, in both languages:

apps/web/src/i18n/es.ts

```
--- a/apps/web/src/i18n/es.ts
+++ b/apps/web/src/i18n/es.ts
@@ -6,7 +6,24 @@
   home: {
     title: 'Tu Pokédex de entrenador',
     body: 'Explora los Pokémon, arma tu equipo de seis y reta a los gimnasios de Costa Rica.',
+    greeting: (name: string) => `Hola, ${name}`,
   },
+  auth: {
+    title: 'Inicia sesión',
+    email: 'Correo',
+    password: 'Contraseña',
+    submit: 'Entrar',
+    submitting: 'Entrando…',
+    signIn: 'Iniciar sesión',
+    signOut: 'Cerrar sesión',
+    checking: 'Comprobando la sesión…',
+    invalidInput: 'Escribe un correo válido y tu contraseña.',
+    wrongCredentials: 'Correo o contraseña incorrectos.',
+    failed: 'No se pudo iniciar sesión. Inténtalo de nuevo.',
+    demoHint: 'Cuenta de prueba: ana@dex506.test · dex506-demo',
+    unavailable: 'No se pudo comprobar la sesión. Revisa tu conexión.',
```

```
+    retry: 'Reintentar',
+  },
   notFound: { title: 'Esta página no existe', back: 'Volver al inicio' },
   routeError: { title: 'Algo salió mal', retry: 'Recargar la página' },
   footer: {
```

apps/web/src/i18n/en.ts

```
--- a/apps/web/src/i18n/en.ts
+++ b/apps/web/src/i18n/en.ts
@@ -9,7 +9,24 @@
   home: {
     title: "Your trainer's Pokédex",
     body: 'Browse the Pokémon, build your team of six, and challenge the gyms of Costa
Rica.',
+    greeting: (name: string) => `Hi, ${name}`,
   },
+  auth: {
+    title: 'Sign in',
+    email: 'Email',
+    password: 'Password',
+    submit: 'Sign in',
+    submitting: 'Signing in…',
+    signIn: 'Sign in',
+    signOut: 'Sign out',
+    checking: 'Checking your session…',
+    invalidInput: 'Enter a valid email and your password.',
+    wrongCredentials: 'Wrong email or password.',
+    failed: 'Could not sign in. Please try again.',
+    demoHint: 'Demo account: ana@dex506.test · dex506-demo',
+    unavailable: 'Could not check your session. Check your connection.',
+    retry: 'Try again',
+  },
   notFound: { title: 'This page does not exist', back: 'Back to the start' },
   routeError: { title: 'Something went wrong', retry: 'Reload the page' },
   footer: {
```

The tests

The MSW handlers the tests share: a signed-in session, with Ana, and one with no session.

apps/web/src/test/handlers.ts

```ts
import { http, HttpResponse } from 'msw/http'
import type { User } from '@dex506/contracts'

export const ANA: User = { id: 'u1', email: 'ana@dex506.test', trainerName: 'Ana' }

/** A signed-in session: /auth/me returns Ana. */
```

```ts
export const signedIn = http.get('/api/auth/me', () => HttpResponse.json(ANA))

/** No session: /auth/me answers 401. */
export const signedOut = http.get('/api/auth/me', () =>
  HttpResponse.json(
    { error: { code: 'UNAUTHENTICATED', message: 'Inicia sesión' } },
    { status: 401 },
  ),
)
```

apps/web/src/features/auth/LoginPage.test.tsx

```tsx
import { screen } from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import { http, HttpResponse } from 'msw/http'
import { describe, expect, it } from 'vitest'
import { ANA, signedIn, signedOut } from '../../test/handlers.ts'
import { renderApp } from '../../test/renderApp.tsx'
import { server } from '../../test/server.ts'

describe('acceso', () => {
  it('sin sesión, una ruta protegida lleva al login', async () => {
    server.use(signedOut)
    renderApp('/')
    expect(await screen.findByRole('heading', { name: 'Inicia sesión'
})).toBeInTheDocument()
  })

  it('con credenciales correctas entra y vuelve a la ruta protegida', async () => {
    server.use(
      signedOut,
      http.post('/api/auth/login', async ({ request }) => {
        const body = (await request.json()) as { email: string; password: string }
        return body.password === 'dex506-demo'
          ? HttpResponse.json(ANA)
          : HttpResponse.json(
              { error: { code: 'INVALID_CREDENTIALS', message: 'x' } },
              { status: 401 },
            )
      }),
    )
    renderApp('/')
    await userEvent.type(await screen.findByLabelText('Correo'), 'ana@dex506.test')
    await userEvent.type(screen.getByLabelText('Contraseña'), 'dex506-demo')
    await userEvent.click(screen.getByRole('button', { name: 'Entrar' }))
    expect(await screen.findByText('Hola, Ana')).toBeInTheDocument()
  })

  it('con una contraseña incorrecta muestra el error sin salir del formulario', async ()
=> {
```

```
    server.use(
      signedOut,
      http.post('/api/auth/login', () =>
        HttpResponse.json(
          { error: { code: 'INVALID_CREDENTIALS', message: 'x' } },
          { status: 401 },
        ),
      ),
    )
    renderApp('/login')
    await userEvent.type(await screen.findByLabelText('Correo'), 'ana@dex506.test')
    await userEvent.type(screen.getByLabelText('Contraseña'), 'otra')
    await userEvent.click(screen.getByRole('button', { name: 'Entrar' }))
    expect(await screen.findByRole('alert')).toHaveTextContent('Correo o contraseña
incorrectos.')
    expect(screen.getByLabelText('Correo')).toHaveValue('ana@dex506.test')
  })

  it('si la API no puede comprobar la sesión, lo dice y permite reintentar', async () =>
{
    server.use(http.get('/api/auth/me', () => new HttpResponse(null, { status: 500 })))
    renderApp('/')
    expect(await screen.findByRole('alert')).toHaveTextContent('No se pudo comprobar la
sesión.')
    server.use(signedIn)
    await userEvent.click(screen.getByRole('button', { name: 'Reintentar' }))
    expect(await screen.findByText('Hola, Ana')).toBeInTheDocument()
  })

  it('con sesión, el header ofrece cerrarla y al cerrarla vuelve al login', async () =>
{
    server.use(
      signedIn,
      http.post('/api/auth/logout', () => new HttpResponse(null, { status: 204 })),
    )
    renderApp('/')
    await userEvent.click(await screen.findByRole('button', { name: 'Cerrar sesión' }))
    expect(await screen.findByRole('heading', { name: 'Inicia sesión'
})).toBeInTheDocument()
  })
})
```

The layout tests now need a session, because the home page requires one:

apps/web/src/app/Layout.test.tsx

```
--- a/apps/web/src/app/Layout.test.tsx
+++ b/apps/web/src/app/Layout.test.tsx
@@ -1,9 +1,13 @@
```

```
 import { screen } from '@testing-library/react'
 import userEvent from '@testing-library/user-event'
-import { describe, expect, it } from 'vitest'
+import { beforeEach, describe, expect, it } from 'vitest'
+import { signedIn } from '../test/handlers.ts'
 import { renderApp } from '../test/renderApp.tsx'
+import { server } from '../test/server.ts'

 describe('Layout', () => {
+  beforeEach(() => server.use(signedIn))
+
   it('muestra la navegación y el aviso de propiedad intelectual', () => {
     renderApp('/')
     expect(screen.getByRole('navigation', { name: 'Navegación principal'
})).toBeInTheDocument()
@@ -14,6 +18,7 @@

   it('cambia de idioma y lo recuerda', async () => {
     renderApp('/')
+    await screen.findByText('Hola, Ana')
     await userEvent.selectOptions(screen.getByRole('combobox', { name: 'Idioma' }),
'en')
     expect(screen.getByRole('heading', { name: "Your trainer's Pokédex"
})).toBeInTheDocument()
     expect(localStorage.getItem('dex506.locale')).toBe('en')
```

Test and sign in

```
pnpm test
```

Vitest should finish with **`Tests 37 passed (37)`**. Then, with **`pnpm dev`**, open **`http://localhost:5173`**: the app takes you to the login. Sign in with **`ana@dex506.test`** and **`dex506-demo`**, and you will see "Hola, Ana" (hello, Ana). Try a wrong password: the error appears, and the email stays in its field. Reload the page while signed in: the app asks the API and keeps you in.

> The usual mistake: the cookie does not arrive and the app goes back to the login after every reload. It happens if the web app calls the API on another origin, without the proxy, or if you mix **`localhost`** and **`127.0.0.1`**: the cookie belongs to a domain name, and those are two. Always open **`http://localhost:5173`**.

```
git add .
git commit -m "paso 06: el acceso"
```

Download the code for step 06.

Step 07: The Pokédex

In this step you will build the list of the 151 Pokémon, with a search box, a type filter, pagination, and two views.

The data

The Pokédex files go in **apps/web/src/features/pokedex/**.

apps/web/src/features/pokedex/api.ts

```
import {
  PokemonSummarySchema,
  PokemonTypeSchema,
  pageSchema,
  type Page,
  type PokemonQuery,
  type PokemonSummary,
  type PokemonType,
} from '@dex506/contracts'
import { api } from '../../shared/api/client.ts'

const PageSchema = pageSchema(PokemonSummarySchema)

export async function fetchPokemonPage(query: PokemonQuery):
Promise<Page<PokemonSummary>> {
  const { data } = await api.get('/pokemon', { params: query })
  return PageSchema.parse(data)
}

export async function fetchTypes(): Promise<PokemonType[]> {
  const { data } = await api.get('/types')
  return PokemonTypeSchema.array().parse(data)
}
```

pageSchema(PokemonSummarySchema) builds, with the contract's generic function, the schema for a page of summaries. **params: query** makes axios turn the object into the URL: **/pokemon?q=lag&page=1&limit=24**. The types are validated too, against the contract's types schema.

apps/web/src/features/pokedex/usePokemonList.ts

```
import { keepPreviousData, useQuery } from '@tanstack/react-query'
import type { PokemonQuery } from '@dex506/contracts'
import { fetchPokemonPage } from './api.ts'

/**
 * One page of the Pokédex. The query key includes the filters: each combination is a
separate cache
```

```
 * entry. With `keepPreviousData`, changing pages keeps showing the previous one until
the new one
 * arrives, without flashing blank.
 */
export function usePokemonList(query: PokemonQuery) {
  return useQuery({
    queryKey: ['pokemon', 'list', query],
    queryFn: () => fetchPokemonPage(query),
    placeholderData: keepPreviousData,
  })
}
```

It is the hook from "TanStack Query", with the whole query in the query key: each combination of filters is a separate cache entry, and **`keepPreviousData`** keeps the previous page visible while the next one arrives.

The view preference

Grid or list is an interface preference that survives a page reload: a Zustand store with **`persist`**, as in "Global State".

apps/web/src/features/pokedex/preferences.ts

```
import { create } from 'zustand'
import { persist } from 'zustand/middleware'

export type View = 'grid' | 'list'

type Preferences = {
  view: View
  setView: (view: View) => void
}

/** Pokédex preferences that survive a page reload (persist → localStorage). */
export const usePreferences = create<Preferences>()(
  persist(
    (set) => ({
      view: 'grid',
      setView: (view) => set({ view }),
    }),
    { name: 'dex506.preferences' },
  ),
)
```

The shared pieces

Three pieces that several screens will use, in **`apps/web/src/shared/`**. The generic **`useDebounce`** from "Advanced TypeScript for React":

apps/web/src/shared/hooks/useDebounce.ts

```
import { useEffect, useState } from 'react'

/** The value, but only once it has stopped changing for `delayMs`: so we do not request
on every key. */
export function useDebounce<T>(value: T, delayMs = 300): T {
  const [debounced, setDebounced] = useState(value)

  useEffect(() => {
    const timer = setTimeout(() => setDebounced(value), delayMs)
    return () => clearTimeout(timer) // if the value changes sooner, the previous wait
is canceled
  }, [value, delayMs])

  return debounced
}
```

A Pokémon's image, with a custom silhouette if the image does not load:

apps/web/src/shared/ui/PokemonSprite.tsx

```
import { useState } from 'react'

/**
 * A Pokémon's image, or our own silhouette if it does not load (offline, or with the
made-up
 * creatures in fixtures/, whose URLs do not exist on purpose).
 */
export function PokemonSprite({
  src,
  alt,
  size = 96,
}: {
  src: string
  alt: string
  size?: number
}) {
  const [failed, setFailed] = useState(false)
  if (failed) {
    return (
      <svg
        role="img"
        aria-label={alt}
        width={size}
        height={size}
        viewBox="0 0 96 96"
        className="text-stone-300 dark:text-stone-700"
      >
        <circle cx="48" cy="48" r="40" fill="currentColor" />
```

```
        <circle cx="48" cy="48" r="12" fill="white" />
      </svg>
    )
  }
  return (
    <img
      src={src}
      alt={alt}
      width={size}
      height={size}
      loading="lazy"
      onError={() => setFailed(true)}
    />
  )
}
```

`loading="lazy"` makes the browser download the image only when it is about to appear on screen. The test creatures have image URLs that do not exist on purpose, and that is why you see the silhouette.

A type badge, with the color from **`@theme`**:

apps/web/src/shared/ui/TypeBadge.tsx

```
import type { PokemonType } from '@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'

// Tailwind only generates the classes it finds written out in full in the code: that is
why there
// is a map with all 18 classes and not `bg-type-${type}`.
const COLORS: Record<PokemonType, string> = {
  normal: 'bg-type-normal',
  fighting: 'bg-type-fighting',
  flying: 'bg-type-flying',
  poison: 'bg-type-poison',
  ground: 'bg-type-ground',
  rock: 'bg-type-rock',
  bug: 'bg-type-bug',
  ghost: 'bg-type-ghost',
  steel: 'bg-type-steel',
  fire: 'bg-type-fire',
  water: 'bg-type-water',
  grass: 'bg-type-grass',
  electric: 'bg-type-electric',
  psychic: 'bg-type-psychic',
  ice: 'bg-type-ice',
  dragon: 'bg-type-dragon',
  dark: 'bg-type-dark',
  fairy: 'bg-type-fairy',
}
```

```
export function TypeBadge({ type }: { type: PokemonType }) {
  const { t } = useI18n()
  return (
    <span
      className={`${COLORS[type]} rounded-full px-2 py-0.5 text-xs font-semibold text-
stone-900`}
    >
      {t.types[type]}
    </span>
  )
}
```

`COLORS` lists the 18 complete classes: it is the care from "Styles", since Tailwind only generates the classes it finds written out in full. **`Record<PokemonType, string>`** requires a color for every type: if the contract added one, this would not compile.

The card and the page

apps/web/src/features/pokedex/PokemonCard.tsx

```
import type { PokemonSummary } from '@dex506/contracts'
import { PokemonSprite } from '../../shared/ui/PokemonSprite.tsx'
import { TypeBadge } from '../../shared/ui/TypeBadge.tsx'
import type { View } from './preferences.ts'

export function PokemonCard({ pokemon, view }: { pokemon: PokemonSummary; view: View })
{
  const number = `#${String(pokemon.id).padStart(3, '0')}`
  return (
    <article
      className={`flex items-center gap-3 rounded-xl border border-stone-200 bg-white p-
3 dark:border-stone-800 dark:bg-stone-900 ${
        view === 'grid' ? 'flex-col text-center' : ''
      }`}
    >
      <PokemonSprite src={pokemon.spriteUrl} alt={pokemon.name} size={view === 'grid' ?
96 : 56} />
      <div className="space-y-1">
        <p className="text-xs text-stone-500 dark:text-stone-400">{number}</p>
        <h2 className="font-semibold capitalize">{pokemon.name}</h2>
        <p className={`flex gap-1 ${view === 'grid' ? 'justify-center' : ''}`}>
          {pokemon.types.map((type) => (
            <TypeBadge key={type} type={type} />
          ))}
        </p>
      </div>
    </article>
```

```
  )
}
```

apps/web/src/features/pokedex/PokedexPage.tsx

```tsx
import { useQuery } from '@tanstack/react-query'
import { useCallback, useEffect, useEffectEvent, useMemo, useState } from 'react'
import { useSearchParams } from 'react-router'
import { POKEMON_TYPES, PokemonTypeSchema, type PokemonType } from '@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { useDebounce } from '../../shared/hooks/useDebounce.ts'
import { fetchTypes } from './api.ts'
import { PokemonCard } from './PokemonCard.tsx'
import { usePreferences } from './preferences.ts'
import { usePokemonList } from './usePokemonList.ts'

const LIMIT = 24

/**
 * The filters live in the URL: they can be shared and survive a reload. Each change
replaces the
 * history entry (`replace`), so "back" does not undo the search one letter at a time.
`filters` and
 * `update` keep their identity until the URL changes.
 */
function useFilters() {
  const [params, setParams] = useSearchParams()
  const filters = useMemo(() => {
    const type = PokemonTypeSchema.safeParse(params.get('type'))
    return {
      q: params.get('q') ?? '',
      type: type.success ? type.data : undefined,
      page: Math.max(1, Number(params.get('page')) || 1),
    }
  }, [params])

  const update = useCallback(
    (next: Partial<typeof filters>) => {
      const merged = { ...filters, ...next }
      const search: Record<string, string> = {}
      if (merged.q) search.q = merged.q
      if (merged.type) search.type = merged.type
      if (merged.page > 1) search.page = String(merged.page)
      setParams(search, { replace: true })
    },
    [filters, setParams],
  )

  return [filters, update] as const
}
```

```tsx
export function PokedexPage() {
  const { t } = useI18n()
  const [filters, setFilters] = useFilters()
  const view = usePreferences((state) => state.view)
  const setView = usePreferences((state) => state.setView)

  // What you type goes to local state first; to the URL (and the API) only once you
stop typing.
  const [text, setText] = useState(filters.q)
  const debouncedText = useDebounce(text.trim(), 300)
  // The effect runs only when the debounced text changes. Reading the current filters
is an
  // event (useEffectEvent): a filter changing does not run it again.
  const pushSearch = useEffectEvent((q: string) => {
    if (q !== filters.q) setFilters({ q, page: 1 })
  })
  useEffect(() => {
    pushSearch(debouncedText)
  }, [debouncedText])

  // If the URL changes from outside (the "Pokédex" link, "back"), the field follows it.
It is state
  // adjusted during render, the way React recommends instead of another effect.
  const [urlText, setUrlText] = useState(filters.q)
  if (filters.q !== urlText) {
    setUrlText(filters.q)
    if (filters.q !== debouncedText) setText(filters.q)
  }

  const types = useQuery({ queryKey: ['types'], queryFn: fetchTypes, staleTime: Infinity
})
  const list = usePokemonList({
    q: filters.q || undefined,
    type: filters.type,
    page: filters.page,
    limit: LIMIT,
  })
  const pages = list.data ? Math.max(1, Math.ceil(list.data.total / LIMIT)) : 1

  return (
    <section className="space-y-4">
      <h1 className="text-3xl font-bold">{t.pokedex.title}</h1>

      <div className="flex flex-wrap items-end gap-3">
        <label className="flex flex-col gap-1 text-sm">
          {t.pokedex.search}
          <input
            type="search"
            value={text}
```

```
            onChange={(event) => setText(event.target.value)}
            className="rounded-md border border-stone-300 px-3 py-2 dark:border-stone-
700 dark:bg-stone-900"
          />
        </label>
        <div className="flex flex-col gap-1 text-sm">
          {/* Explicit label (htmlFor + id): wrapped in the <label>, the <select>'s name
would
              include the selected option. */}
          <label htmlFor="pokedex-type">{t.pokedex.type}</label>
          <select
            id="pokedex-type"
            value={filters.type ?? ''}
            onChange={(event) =>
              setFilters({
                type: (event.target.value || undefined) as PokemonType | undefined,
                page: 1,
              })
            }
            className="rounded-md border border-stone-300 px-3 py-2 dark:border-stone-
700 dark:bg-stone-900"
          >
            <option value="">{t.pokedex.allTypes}</option>
            {(types.data ?? POKEMON_TYPES).map((type) => (
              <option key={type} value={type}>
                {t.types[type as PokemonType] ?? type}
              </option>
            ))}
          </select>
        </div>
        <fieldset className="flex gap-1 text-sm">
          <legend className="sr-only">{t.pokedex.view}</legend>
          {(['grid', 'list'] as const).map((option) => (
            <button
              key={option}
              type="button"
              aria-pressed={view === option}
              onClick={() => setView(option)}
              className={`rounded-md border px-3 py-2 ${
                view === option
                  ? 'border-brand-700 bg-brand-700 text-white'
                  : 'border-stone-300 dark:border-stone-700'
              }`}
            >
              {t.pokedex[option]}
            </button>
          ))}
        </fieldset>
      </div>
```

```
      {list.isPending ? (
        <output className="block">{t.pokedex.loading}</output>
      ) : list.isError ? (
        <div role="alert" className="space-y-2">
          <p>{t.pokedex.error}</p>
          <button type="button" onClick={() => void list.refetch()}
className="underline">
            {t.pokedex.retry}
          </button>
        </div>
      ) : list.data.data.length === 0 ? (
        <p>{t.pokedex.empty}</p>
      ) : (
        <>
          <p className="text-sm text-stone-600 dark:text-stone-400">
            {t.pokedex.results(list.data.total)}
          </p>
          <ul
            aria-busy={list.isPlaceholderData}
            className={
              view === 'grid'
                ? 'grid grid-cols-2 gap-3 sm:grid-cols-3 md:grid-cols-4'
                : 'flex flex-col gap-2'
            }
          >
            {list.data.data.map((pokemon) => (
              <li key={pokemon.id}>
                <PokemonCard pokemon={pokemon} view={view} />
              </li>
            ))}
          </ul>
          <nav
            aria-label={t.pokedex.pageOf(filters.page, pages)}
            className="flex items-center gap-3"
          >
            <button
              type="button"
              disabled={filters.page <= 1}
              onClick={() => setFilters({ page: filters.page - 1 })}
              className="rounded-md border border-stone-300 px-3 py-1 disabled:opacity-
40 dark:border-stone-700"
            >
              {t.pokedex.previous}
            </button>
            <span className="text-sm">{t.pokedex.pageOf(filters.page, pages)}</span>
            <button
              type="button"
              disabled={filters.page >= pages}
              onClick={() => setFilters({ page: filters.page + 1 })}
```

```
              className="rounded-md border border-stone-300 px-3 py-1 disabled:opacity-
40 dark:border-stone-700"
            >
              {t.pokedex.next}
            </button>
          </nav>
        </>
      )}
    </section>
  )
}
```

- **The filters live in the URL**, with **`useSearchParams`** and **`replace`**, as in "Routing with React Router". **`useFilters`** is a custom hook that reads the filters from the URL, validates them (a **`type`** that does not exist is ignored), and changes them.
- **`filters`** uses **`useMemo`** and **`update`** uses **`useCallback`**, as in "Performance": that way their identity does not change as long as the URL does not change. The page receives **`update`** under the name **`setFilters`**.
- **What you type goes first to local state**, **`text`**, and to the URL only once you stop typing, with **`useDebounce`**: that way there is no request per letter. The effect that carries it to the URL runs only when the debounced text changes; reading the current filters is an event, with **`useEffectEvent`**, as in "useEffect Without Fear".
- **If the URL changes from outside**, for example with the "Pokédex" link in the navigation, the field follows it. It is state adjusted during render (**`urlText`**), the approach React recommends instead of another effect: when the URL's **`q`** changes and it is not the one you typed, the field takes the new one.
- **The type select** has its label separate, like the language one from step 05.
- **The four states of a list that comes from the server:** loading, error (with a button to retry), empty, and with data.
- **The pagination** is a **`<nav>`** with its **`aria-label`**, and the buttons are disabled on the first and last pages.
- **The view buttons** use **`aria-pressed`**, which tells screen readers which one is active.

The changes

The new route, inside the ones that require a session:

apps/web/src/app/routes.tsx

```
--- a/apps/web/src/app/routes.tsx
+++ b/apps/web/src/app/routes.tsx
@@ -1,6 +1,7 @@
 import type { RouteObject } from 'react-router'
 import { LoginPage } from '../features/auth/LoginPage.tsx'
 import { RequireAuth } from '../features/auth/RequireAuth.tsx'
```

```
+import { PokedexPage } from '../features/pokedex/PokedexPage.tsx'
 import { HomePage } from './HomePage.tsx'
 import { Layout } from './Layout.tsx'
 import { NotFoundPage } from './NotFoundPage.tsx'
@@ -17,7 +18,10 @@
       {
         // A route without a path: it only wraps its children with the session check.
         element: <RequireAuth />,
-        children: [{ index: true, element: <HomePage /> }],
+        children: [
+          { index: true, element: <HomePage /> },
+          { path: 'pokedex', element: <PokedexPage /> },
+        ],
       },
       { path: '*', element: <NotFoundPage /> },
     ],
```

Zustand, as a new dependency:

apps/web/package.json

```
--- a/apps/web/package.json
+++ b/apps/web/package.json
@@ -15,7 +15,8 @@
     "axios": "1.21.1",
     "react": "19.3.1",
     "react-dom": "19.3.1",
-    "react-router": "8.5.1"
+    "react-router": "8.5.1",
+    "zustand": "5.1.0"
   },
   "devDependencies": {
     "@tailwindcss/vite": "4.3.5",
```

`resolveJsonModule` allows importing a **`.json`** file: the test handlers need it, since they read the creatures from **`fixtures/`**.

apps/web/tsconfig.app.json

```
--- a/apps/web/tsconfig.app.json
+++ b/apps/web/tsconfig.app.json
@@ -4,7 +4,8 @@
     "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
     "lib": ["ES2023", "DOM", "DOM.Iterable"],
     "types": ["vite/client"],
-    "jsx": "react-jsx"
+    "jsx": "react-jsx",
+    "resolveJsonModule": true
```

```
  },
  "include": ["src"]
}
```

The Pokédex texts and the names of the 18 types, in both languages:

apps/web/src/i18n/es.ts

```
--- a/apps/web/src/i18n/es.ts
+++ b/apps/web/src/i18n/es.ts
@@ -24,6 +24,43 @@
     unavailable: 'No se pudo comprobar la sesión. Revisa tu conexión.',
     retry: 'Reintentar',
   },
+  types: {
+    normal: 'Normal',
+    fighting: 'Lucha',
+    flying: 'Volador',
+    poison: 'Veneno',
+    ground: 'Tierra',
+    rock: 'Roca',
+    bug: 'Bicho',
+    ghost: 'Fantasma',
+    steel: 'Acero',
+    fire: 'Fuego',
+    water: 'Agua',
+    grass: 'Planta',
+    electric: 'Eléctrico',
+    psychic: 'Psíquico',
+    ice: 'Hielo',
+    dragon: 'Dragón',
+    dark: 'Siniestro',
+    fairy: 'Hada',
+  },
+  pokedex: {
+    title: 'Pokédex',
+    search: 'Buscar por nombre o número',
+    type: 'Tipo',
+    allTypes: 'Todos los tipos',
+    view: 'Vista',
+    grid: 'Cuadrícula',
+    list: 'Lista',
+    loading: 'Cargando Pokémon…',
+    error: 'No se pudo cargar la Pokédex.',
+    retry: 'Reintentar',
+    empty: 'Ningún Pokémon coincide con la búsqueda.',
+    previous: 'Anterior',
+    next: 'Siguiente',
+    pageOf: (page: number, pages: number) => `Página ${page} de ${pages}`,
```

```
+    results: (total: number) => (total === 1 ? '1 Pokémon' : `${total} Pokémon`),
+  },
   notFound: { title: 'Esta página no existe', back: 'Volver al inicio' },
   routeError: { title: 'Algo salió mal', retry: 'Recargar la página' },
   footer: {
```

apps/web/src/i18n/en.ts

```
--- a/apps/web/src/i18n/en.ts
+++ b/apps/web/src/i18n/en.ts
@@ -27,6 +27,43 @@
     unavailable: 'Could not check your session. Check your connection.',
     retry: 'Try again',
   },
+  types: {
+    normal: 'Normal',
+    fighting: 'Fighting',
+    flying: 'Flying',
+    poison: 'Poison',
+    ground: 'Ground',
+    rock: 'Rock',
+    bug: 'Bug',
+    ghost: 'Ghost',
+    steel: 'Steel',
+    fire: 'Fire',
+    water: 'Water',
+    grass: 'Grass',
+    electric: 'Electric',
+    psychic: 'Psychic',
+    ice: 'Ice',
+    dragon: 'Dragon',
+    dark: 'Dark',
+    fairy: 'Fairy',
+  },
+  pokedex: {
+    title: 'Pokédex',
+    search: 'Search by name or number',
+    type: 'Type',
+    allTypes: 'All types',
+    view: 'View',
+    grid: 'Grid',
+    list: 'List',
+    loading: 'Loading Pokémon…',
+    error: 'Could not load the Pokédex.',
+    retry: 'Try again',
+    empty: 'No Pokémon match your search.',
+    previous: 'Previous',
+    next: 'Next',
+    pageOf: (page: number, pages: number) => `Page ${page} of ${pages}`,
+    results: (total: number) => (total === 1 ? '1 Pokémon' : `${total} Pokémon`),
```

```
+  },
   notFound: { title: 'This page does not exist', back: 'Back to the start' },
   routeError: { title: 'Something went wrong', retry: 'Reload the page' },
   footer: {
```

The tests

The handlers add the list and the types, with the made-up creatures, filtering the way the real API does:

apps/web/src/test/handlers.ts

```
--- a/apps/web/src/test/handlers.ts
+++ b/apps/web/src/test/handlers.ts
@@ -1,5 +1,7 @@
 import { http, HttpResponse } from 'msw/http'
 import type { User } from '@dex506/contracts'
+import fixtures from '../../../../fixtures/pokemon.json'
+import { POKEMON_TYPES, type Pokemon } from '@dex506/contracts'

 export const ANA: User = { id: 'u1', email: 'ana@dex506.test', trainerName: 'Ana' }

@@ -13,3 +15,28 @@
     { status: 401 },
   ),
 )
+
+export const POKEDEX = fixtures as Pokemon[]
+
+/** /api/pokemon with the creatures from fixtures/, filtering like the real API. */
+export const pokemonList = http.get('/api/pokemon', ({ request }) => {
+  const url = new URL(request.url)
+  const q = url.searchParams.get('q')?.toLowerCase()
+  const type = url.searchParams.get('type')
+  const page = Number(url.searchParams.get('page') ?? 1)
+  const limit = Number(url.searchParams.get('limit') ?? 24)
+  const matches = POKEDEX.filter(
+    (p) =>
+      (!q || p.name.includes(q)) && (!type || p.types.includes(type as Pokemon['types']
[number])),
+  )
+  return HttpResponse.json({
+    data: matches
+      .slice((page - 1) * limit, page * limit)
+      .map(({ id, name, types, spriteUrl }) => ({ id, name, types, spriteUrl })),
+    page,
+    limit,
+    total: matches.length,
+  })
+})
```

```
+
+export const types = http.get('/api/types', () => HttpResponse.json(POKEMON_TYPES))
```

apps/web/src/features/pokedex/PokedexPage.test.tsx

```
import { screen, waitFor, within } from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import { http, HttpResponse } from 'msw/http'
import { beforeEach, describe, expect, it } from 'vitest'
import { pokemonList, signedIn, types } from '../../test/handlers.ts'
import { renderApp } from '../../test/renderApp.tsx'
import { server } from '../../test/server.ts'

describe('Pokédex', () => {
  beforeEach(() => server.use(signedIn, types, pokemonList))

  it('muestra la lista con el total de resultados', async () => {
    renderApp('/pokedex')
    expect(await screen.findByText('12 Pokémon')).toBeInTheDocument()
    expect(screen.getByRole('heading', { name: 'brotin' })).toBeInTheDocument()
  })

  it('la búsqueda espera a que se deje de escribir y queda en la URL', async () => {
    const { router } = renderApp('/pokedex')
    await screen.findByText('12 Pokémon')
    await userEvent.type(screen.getByRole('searchbox'), 'lag')
    expect(await screen.findByText('1 Pokémon')).toBeInTheDocument()
    expect(router.state.location.search).toBe('?q=lag')
  })

  it('el filtro por tipo viene de la URL y se puede cambiar', async () => {
    const { router } = renderApp('/pokedex?type=grass')
    expect(await screen.findByText('2 Pokémon')).toBeInTheDocument()
    await userEvent.selectOptions(screen.getByRole('combobox', { name: 'Tipo' }),
'fire')
    expect(await screen.findByText('1 Pokémon')).toBeInTheDocument()
    expect(router.state.location.search).toBe('?type=fire')
  })

  it('pagina con anterior y siguiente', async () => {
    server.use(
      http.get('/api/pokemon', ({ request }) => {
        const page = Number(new URL(request.url).searchParams.get('page') ?? 1)
        const first = (page - 1) * 24 + 1
        return HttpResponse.json({
          data: [
            {
              id: first,
              name: `criatura-${first}`,
              types: ['normal'],
```

```
              spriteUrl: 'https://dex506.invalid/x.png',
            },
          ],
          page,
          limit: 24,
          total: 30,
        })
      }),
    )
    renderApp('/pokedex')
    expect(await screen.findByText('Página 1 de 2')).toBeInTheDocument()
    await userEvent.click(screen.getByRole('button', { name: 'Siguiente' }))
    expect(await screen.findByRole('heading', { name: 'criatura-25'
})).toBeInTheDocument()
    expect(screen.getByRole('button', { name: 'Siguiente' })).toBeDisabled()
  })

  it('el enlace «Pokédex» limpia la búsqueda, y el campo la sigue', async () => {
    const { router } = renderApp('/pokedex?q=lag')
    expect(await screen.findByRole('searchbox')).toHaveValue('lag')
    await userEvent.click(screen.getByRole('link', { name: 'Pokédex' }))
    expect(screen.getByRole('searchbox')).toHaveValue('')
    await new Promise((resolve) => setTimeout(resolve, 400))
    expect(router.state.location.search).toBe('')
  })

  it('si la API falla muestra el error y permite reintentar', async () => {
    let failing = true
    server.use(
      http.get('/api/pokemon', () =>
        failing
          ? HttpResponse.json({ error: { code: 'UNAVAILABLE', message: 'x' } }, {
status: 503 })
          : HttpResponse.json({ data: [], page: 1, limit: 24, total: 0 }),
      ),
    )
    renderApp('/pokedex')
    const alert = await screen.findByRole('alert')
    expect(alert).toHaveTextContent('No se pudo cargar la Pokédex.')
    failing = false
    await userEvent.click(within(alert).getByRole('button', { name: 'Reintentar' }))
    expect(await screen.findByText('Ningún Pokémon coincide con la
búsqueda.')).toBeInTheDocument()
  })

  it('la vista lista se recuerda al volver a entrar', async () => {
    const first = renderApp('/pokedex')
    await screen.findByText('12 Pokémon')
    await userEvent.click(screen.getByRole('button', { name: 'Lista' }))
```

```
    expect(screen.getByRole('button', { name: 'Lista' })).toHaveAttribute('aria-
pressed', 'true')
    await waitFor(() =>
      expect(localStorage.getItem('dex506.preferences')).toContain('"view":"list"'),
    )
    first.unmount()
  })
})
```

Test and use

```
pnpm install
pnpm test
```

Vitest should finish with **`Tests 44 passed (44)`**. With **`pnpm dev`**, sign in and open the Pokédex: search for "char", filter by a type, change the page and the view. With the made-up creatures, search for "lag". There are twelve, so they fit on a single page: to see the pagination you need the 151 from PokéAPI. Copy the URL into another tab and you will see the same filters. To see the error state, open the browser's developer tools, in the network tab block the URL of the request to **`/api/pokemon`** (right-click the request, "Block request URL" in Chrome), and reload: the error appears with its "Reintentar" (retry) button.

```
git add .
git commit -m "paso 07: la Pokédex"
```

Download the code for step 07.

Step 08: The Detail Page

In this step you will build a Pokémon's detail page, with React 19 in depth: lazy loading, an optimistic favorite, View Transition, tabs with **`<Activity>`**, and a comparison tool.

The new files go in **`apps/web/src/features/pokemon/`**.

The data

apps/web/src/features/pokemon/api.ts

```
import { PokemonSchema, type Pokemon } from '@dex506/contracts'
import { z } from 'zod'
import { api } from '../../shared/api/client.ts'

export async function fetchPokemon(id: number): Promise<Pokemon> {
  const { data } = await api.get(`/pokemon/${id}`)
```

```
  return PokemonSchema.parse(data)
}

export async function fetchFavorites(): Promise<number[]> {
  const { data } = await api.get('/favorites')
  return z.array(z.number()).parse(data)
}

export async function setFavorite(id: number, favorite: boolean): Promise<void> {
  if (favorite) await api.put(`/favorites/${id}`)
  else await api.delete(`/favorites/${id}`)
}
```

`setFavorite` uses **`PUT`** to mark and **`DELETE`** to unmark, the API's two idempotent routes.

apps/web/src/features/pokemon/queries.ts

```
import { useQuery } from '@tanstack/react-query'
import { fetchFavorites, fetchPokemon } from './api.ts'

export const pokemonKey = (id: number) => ['pokemon', 'detail', id] as const
export const FAVORITES_KEY = ['favorites'] as const

export function usePokemon(id: number) {
  return useQuery({
    queryKey: pokemonKey(id),
    queryFn: () => fetchPokemon(id),
    staleTime: Infinity,
  })
}

export function useFavorites() {
  return useQuery({ queryKey: FAVORITES_KEY, queryFn: fetchFavorites })
}
```

`pokemonKey` is a function that builds a Pokémon's key: the detail page and the comparison tool use it the same way, and if you visited the detail page, the comparison tool finds the Pokémon in the cache. **`staleTime: Infinity`**: a Pokémon's data does not change.

The optimistic favorite

apps/web/src/features/pokemon/FavoriteButton.tsx

```
import { useMutation, useQueryClient } from '@tanstack/react-query'
import { useOptimistic, useState, useTransition } from 'react'
import { useI18n } from '../../i18n/i18nContext.ts'
import { setFavorite } from './api.ts'
import { FAVORITES_KEY, useFavorites } from './queries.ts'
```

```tsx
export function FavoriteButton({ pokemonId }: { pokemonId: number }) {
  const { t } = useI18n()
  const queryClient = useQueryClient()
  const favorites = useFavorites()
  const saved = favorites.data?.includes(pokemonId) ?? false

  // While the request is in flight, the interface shows the new value. If it fails, when the
  // transition ends React goes back to the real value (`saved`) by itself: nothing to undo by hand.
  const [shown, setShown] = useOptimistic(saved)
  const [isPending, startTransition] = useTransition()
  const [failed, setFailed] = useState(false)

  const mutation = useMutation({
    mutationFn: (favorite: boolean) => setFavorite(pokemonId, favorite),
    onSettled: () => queryClient.invalidateQueries({ queryKey: FAVORITES_KEY }),
  })

  function toggle() {
    // The button stays enabled while saving, so it keeps the keyboard focus: a second click waits.
    if (isPending) return
    setFailed(false)
    startTransition(async () => {
      setShown(!saved)
      try {
        await mutation.mutateAsync(!saved)
      } catch {
        setFailed(true)
      }
    })
  }

  return (
    <div className="space-y-1">
      <button
        type="button"
        aria-pressed={shown}
        disabled={favorites.isPending}
        onClick={toggle}
        className="rounded-md border border-accent-600 px-3 py-1 text-sm text-accent-600 aria-pressed:bg-accent-600 aria-pressed:text-white"
      >
        {/* The name does not change with the state: aria-pressed says whether it is marked. */}
        <span aria-hidden="true">{shown ? '★' : '☆'}</span> {t.detail.favorite}
      </button>
      {failed && (
```

```
        <p role="alert" className="text-sm text-accent-600">
          {t.detail.favoriteFailed}
        </p>
      )}
    </div>
  )
}
```

It is the pattern from "useOptimistic" with TanStack Query, and an accessible toggle button:

- the real value, **saved**, comes from the favorites query, and **shown** is what you see while it is saving;
- **setShown** goes inside **startTransition**, and **mutateAsync** waits for the save;
- **onSettled** invalidates the favorites whether it went well or badly: the cache goes back to the server's truth;
- if it fails, React goes back to the real value on its own and **failed** shows the notice;
- the button's name does not change with its state, "Favorito" (favorite): **aria-pressed** says it, and the star is decorative (**aria-hidden**). A screen reader announces "Favorito, button, pressed";
- while it is saving, the button is not disabled: a disabled button loses keyboard focus. **toggle** ignores a second click while one is in progress.

The comparison tool

Choosing two Pokémon to compare is interface state that the detail page and the Pokédex share: a Zustand store, without **persist**, because it does not need to survive a reload.

apps/web/src/features/pokemon/compare.ts

```
import { create } from 'zustand'

export const MAX_COMPARE = 2

type CompareState = {
  ids: number[]
  toggle: (id: number) => void
  clear: () => void
}

/** The Pokémon chosen for comparison: at most two; a third replaces the oldest. */
export const useCompare = create<CompareState>()((set) => ({
  ids: [],
  toggle: (id) =>
    set(({ ids }) => ({
      ids: ids.includes(id)
        ? ids.filter((current) => current !== id)
        : [...ids, id].slice(-MAX_COMPARE),
    })),
```

```
  clear: () => set({ ids: [] }),
}))
```

`slice(-MAX_COMPARE)` keeps the last two: choosing a third one replaces the oldest.

apps/web/src/features/pokemon/CompareBar.tsx

```tsx
import { useQueries } from '@tanstack/react-query'
import type { Stats } from '@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { fetchPokemon } from './api.ts'
import { useCompare } from './compare.ts'
import { pokemonKey } from './queries.ts'

const STAT_KEYS: (keyof Stats)[] = [
  'hp',
  'attack',
  'defense',
  'specialAttack',
  'specialDefense',
  'speed',
]

/** The comparison of up to two Pokémon, side by side. It reads the selection from the
Zustand store. */
export function CompareBar() {
  const { t } = useI18n()
  const ids = useCompare((state) => state.ids)
  const clear = useCompare((state) => state.clear)
  // One query per chosen Pokémon, with the same key as the detail page: if it was
visited, it comes from the cache.
  const results = useQueries({
    queries: ids.map((id) => ({
      queryKey: pokemonKey(id),
      queryFn: () => fetchPokemon(id),
      staleTime: Infinity,
    })),
  })
  const pokemon = results.flatMap((result) => (result.data ? [result.data] : []))
  if (ids.length === 0) return null

  return (
    <aside aria-label={t.compare.title} className="rounded-xl border border-brand-600 p-
4">
      <div className="mb-2 flex items-center justify-between">
        <h2 className="font-semibold">{t.compare.title}</h2>
        <button type="button" onClick={clear} className="text-sm underline">
          {t.compare.clear}
        </button>
```

```
      </div>
      {pokemon.length < 2 ? (
        <p className="text-sm">{t.compare.pickAnother}</p>
      ) : (
        <table className="w-full text-sm">
          <thead>
            <tr>
              <th scope="col" className="text-left">
                {t.compare.stat}
              </th>
              {pokemon.map((p) => (
                <th key={p.id} scope="col" className="text-right capitalize">
                  {p.name}
                </th>
              ))}
            </tr>
          </thead>
          <tbody>
            {STAT_KEYS.map((key) => (
              <tr key={key}>
                <th scope="row" className="text-left font-normal">
                  {t.stats[key]}
                </th>
                {pokemon.map((p) => (
                  <td key={p.id} className="text-right tabular-nums">
                    {p.stats[key]}
                  </td>
                ))}
              </tr>
            ))}
          </tbody>
        </table>
      )}
    </aside>
  )
}
```

`useQueries` is the version of **`useQuery`** for a list of queries that changes: one for each chosen Pokémon. The table uses **`<th scope="col">`** and **`<th scope="row">`**, so a screen reader reads each number with its stat and its Pokémon.

The page

apps/web/src/features/pokemon/PokemonDetailPage.tsx

```
import { Activity, useState, ViewTransition } from 'react'
import { Link, useParams } from 'react-router'
import type { Pokemon, Stats } from '@dex506/contracts'
```

```
import { useI18n } from '../../i18n/i18nContext.ts'
import { toApiError } from '../../shared/api/client.ts'
import { PokemonSprite } from '../../shared/ui/PokemonSprite.tsx'
import { TypeBadge } from '../../shared/ui/TypeBadge.tsx'
import { useCompare } from './compare.ts'
import { FavoriteButton } from './FavoriteButton.tsx'
import { usePokemon } from './queries.ts'

type Tab = 'stats' | 'moves'
const STAT_KEYS: (keyof Stats)[] = [
  'hp',
  'attack',
  'defense',
  'specialAttack',
  'specialDefense',
  'speed',
]

function StatsPanel({ pokemon }: { pokemon: Pokemon }) {
  const { t } = useI18n()
  return (
    <dl className="grid grid-cols-[auto_1fr_auto] items-center gap-x-3 gap-y-2">
      {STAT_KEYS.map((key) => (
        <div key={key} className="contents">
          <dt className="text-sm">{t.stats[key]}</dt>
          <dd aria-hidden="true" className="h-2 rounded-full bg-stone-200 dark:bg-stone-
800">
            <div
              className="h-2 rounded-full bg-brand-600"
              style={{ width: `${Math.min(100, (pokemon.stats[key] / 160) * 100)}%` }}
            />
          </dd>
          <dd className="text-sm tabular-nums">{pokemon.stats[key]}</dd>
        </div>
      ))}
    </dl>
  )
}

function MovesPanel({ moves }: { moves: string[] }) {
  const { t } = useI18n()
  const [filter, setFilter] = useState('')
  const visible = moves.filter((move) => move.includes(filter.toLowerCase()))
  return (
    <div className="space-y-2">
      <label className="flex flex-col gap-1 text-sm">
        {t.detail.filterMoves}
        <input
          value={filter}
          onChange={(event) => setFilter(event.target.value)}
```

```
          className="rounded-md border border-stone-300 px-3 py-2 dark:border-stone-700
dark:bg-stone-900"
        />
      </label>
      <ul className="flex flex-wrap gap-1">
        {visible.map((move) => (
          <li key={move} className="rounded bg-stone-100 px-2 py-0.5 text-sm dark:bg-
stone-800">
            {move}
          </li>
        ))}
      </ul>
    </div>
  )
}

export default function PokemonDetailPage() {
  const { t } = useI18n()
  const id = Number(useParams().id)
  const pokemon = usePokemon(id)
  const [tab, setTab] = useState<Tab>('stats')
  const comparing = useCompare((state) => state.ids.includes(id))
  const toggleCompare = useCompare((state) => state.toggle)

  if (pokemon.isPending) return <output className="block">{t.detail.loading}</output>
  if (pokemon.isError) {
    const notFound = toApiError(pokemon.error).error.code === 'POKEMON_NOT_FOUND'
    return (
      <section role="alert" className="space-y-2">
        <p>{notFound ? t.detail.notFound : t.detail.error}</p>
        <Link to="/pokedex" className="underline">
          {t.detail.back}
        </Link>
      </section>
    )
  }

  const { data } = pokemon
  return (
    <article className="space-y-6">
      <Link to="/pokedex" className="text-sm underline">
        {t.detail.back}
      </Link>
      <header className="flex flex-wrap items-center gap-6">
        {/* The same name as in the Pokédex card: the browser animates the image from
one place to the other. */}
        <ViewTransition name={`pokemon-${data.id}`}>
          <PokemonSprite src={data.artworkUrl ?? data.spriteUrl} alt={data.name} size=
{160} />
        </ViewTransition>
```

```
        <div className="space-y-2">
          <p className="text-sm text-stone-500 dark:text-stone-400">
            #{String(data.id).padStart(3, '0')}
          </p>
          <h1 className="text-3xl font-bold capitalize">{data.name}</h1>
          <p className="flex gap-1">
            {data.types.map((type) => (
              <TypeBadge key={type} type={type} />
            ))}
          </p>
          <p className="text-sm">{t.detail.size(data.height / 10, data.weight / 10)}</p>
          <div className="flex flex-wrap items-start gap-2">
            <FavoriteButton pokemonId={data.id} />
            <button
              type="button"
              aria-pressed={comparing}
              onClick={() => toggleCompare(data.id)}
              className="rounded-md border border-stone-300 px-3 py-1 text-sm
dark:border-stone-700"
            >
              {t.detail.compare}
            </button>
          </div>
        </div>
      </header>

      <div
        role="tablist"
        aria-label={t.detail.tabs}
        className="flex gap-2 border-b border-stone-200 dark:border-stone-800"
      >
        {(['stats', 'moves'] as const).map((option) => (
          <button
            key={option}
            type="button"
            role="tab"
            id={`tab-${option}`}
            aria-selected={tab === option}
            aria-controls={`panel-${option}`}
            // The ARIA tabs pattern: only the selected tab is in the Tab order, and the
arrows move
            // between tabs.
            tabIndex={tab === option ? 0 : -1}
            onClick={() => setTab(option)}
            onKeyDown={(event) => {
              if (event.key !== 'ArrowRight' && event.key !== 'ArrowLeft') return
              const next = option === 'stats' ? 'moves' : 'stats'
              setTab(next)
              document.getElementById(`tab-${next}`)?.focus()
            }}
```

```
            className="border-b-2 border-transparent px-3 py-2 aria-selected:border-
brand-700 aria-selected:font-semibold"
          >
            {t.detail[option]}
          </button>
        ))}
      </div>
      {/* With <Activity>, the hidden tab is not unmounted: the moves filter keeps what
was typed. */}
      <Activity mode={tab === 'stats' ? 'visible' : 'hidden'}>
        <section role="tabpanel" id="panel-stats" aria-labelledby="tab-stats">
          <StatsPanel pokemon={data} />
        </section>
      </Activity>
      <Activity mode={tab === 'moves' ? 'visible' : 'hidden'}>
        <section role="tabpanel" id="panel-moves" aria-labelledby="tab-moves">
          <MovesPanel moves={data.moves} />
        </section>
      </Activity>
    </article>
  )
}
```

- **`export default`**: the route loads it with **`lazy`** and imports the module's default value.
- **A Pokémon that does not exist** is a 404 from the API, with the code **`POKEMON_NOT_FOUND`**, and it is shown with its way back.
- **`<ViewTransition name={…}>`** has the same name as the card's image: when you go from the Pokédex to the detail page, the browser animates the image from one place to the other. There is a condition: both images have to be in the same transition. If the detail page is not in the cache yet, the first thing you see is "Cargando la ficha..." (loading the detail page), and the image arrives later, without the shared journey. When you go back to a Pokémon you already visited, you do see the full animation.
- **The compare button** follows the same rule as the favorite: its name is always "Comparar" (compare), and **`aria-pressed`** says whether it is selected.
- **The tabs** follow the ARIA tabs pattern: **`role="tablist"`**, **`role="tab"`**, and **`role="tabpanel"`**, connected with **`aria-controls`** and **`aria-labelledby`**. Only the selected tab is in the tab order (**`tabIndex`**), and the arrow keys move to the other one and take the focus with them.
- **`<Activity>`** hides the tab you do not see without unmounting it, as in "Activity": the moves filter keeps what you typed.
- **`StatsPanel`** draws one bar per stat. The bar is decorative (**`aria-hidden`**): the number next to it is the data.

The card as a link

The Pokédex card becomes a link to the detail page:

apps/web/src/features/pokedex/PokemonCard.tsx

```
--- a/apps/web/src/features/pokedex/PokemonCard.tsx
+++ b/apps/web/src/features/pokedex/PokemonCard.tsx
@@ -1,3 +1,5 @@
+import { ViewTransition } from 'react'
+import { Link } from 'react-router'
 import type { PokemonSummary } from '@dex506/contracts'
 import { PokemonSprite } from '../../shared/ui/PokemonSprite.tsx'
 import { TypeBadge } from '../../shared/ui/TypeBadge.tsx'
@@ -7,14 +9,21 @@
   const number = `#${String(pokemon.id).padStart(3, '0')}`
   return (
     <article
-      className={`flex items-center gap-3 rounded-xl border border-stone-200 bg-white
p-3 dark:border-stone-800 dark:bg-stone-900 ${
+      className={`relative flex items-center gap-3 rounded-xl border border-stone-200
bg-white p-3 dark:border-stone-800 dark:bg-stone-900 ${
         view === 'grid' ? 'flex-col text-center' : ''
       }`}
     >
-      <PokemonSprite src={pokemon.spriteUrl} alt={pokemon.name} size={view === 'grid' ?
96 : 56} />
+      <ViewTransition name={`pokemon-${pokemon.id}`}>
+        <PokemonSprite src={pokemon.spriteUrl} alt="" size={view === 'grid' ? 96 : 56}
/>
+      </ViewTransition>
       <div className="space-y-1">
         <p className="text-xs text-stone-500 dark:text-stone-400">{number}</p>
-        <h2 className="font-semibold capitalize">{pokemon.name}</h2>
+        <h2 className="font-semibold capitalize">
+          {/* The link covers the whole card with a pseudo-element: a single target for
the keyboard. */}
+          <Link to={`/pokedex/${pokemon.id}`} className="after:absolute after:inset-0">
+            {pokemon.name}
+          </Link>
+        </h2>
         <p className={`flex gap-1 ${view === 'grid' ? 'justify-center' : ''}`}>
           {pokemon.types.map((type) => (
             <TypeBadge key={type} type={type} />
```

- The link is on the name, and its **`after:absolute after:inset-0`** stretches it over the whole card: you can tap anywhere, and for the keyboard it is a single destination.
- The image changes to **`alt=""`**: the name is already in the link, and repeating it would make a screen reader say it twice.
- The image's **`<ViewTransition>`** has the name the detail page repeats.

The Pokédex shows the comparison tool at the top:

apps/web/src/features/pokedex/PokedexPage.tsx

```
--- a/apps/web/src/features/pokedex/PokedexPage.tsx
+++ b/apps/web/src/features/pokedex/PokedexPage.tsx
@@ -4,6 +4,7 @@
 import { POKEMON_TYPES, PokemonTypeSchema, type PokemonType } from '@dex506/contracts'
 import { useI18n } from '../../i18n/i18nContext.ts'
 import { useDebounce } from '../../shared/hooks/useDebounce.ts'
+import { CompareBar } from '../pokemon/CompareBar.tsx'
 import { fetchTypes } from './api.ts'
 import { PokemonCard } from './PokemonCard.tsx'
 import { usePreferences } from './preferences.ts'
@@ -80,6 +81,7 @@
   return (
     <section className="space-y-4">
       <h1 className="text-3xl font-bold">{t.pokedex.title}</h1>
+      <CompareBar />

       <div className="flex flex-wrap items-end gap-3">
         <label className="flex flex-col gap-1 text-sm">
```

The detail page's route, with lazy loading:

apps/web/src/app/routes.tsx

```
--- a/apps/web/src/app/routes.tsx
+++ b/apps/web/src/app/routes.tsx
@@ -2,6 +2,7 @@
 import { LoginPage } from '../features/auth/LoginPage.tsx'
 import { RequireAuth } from '../features/auth/RequireAuth.tsx'
 import { PokedexPage } from '../features/pokedex/PokedexPage.tsx'
+import { DetailLoading } from '../features/pokemon/DetailLoading.tsx'
 import { HomePage } from './HomePage.tsx'
 import { Layout } from './Layout.tsx'
 import { NotFoundPage } from './NotFoundPage.tsx'
@@ -21,6 +22,18 @@
         children: [
           { index: true, element: <HomePage /> },
           { path: 'pokedex', element: <PokedexPage /> },
+          {
+            // The detail page is downloaded only when someone opens it: it adds
nothing to the first load.
+            path: 'pokedex/:id',
+            // Opening a detail page directly (a reload, a shared link), its code is
not there yet: the
+            // router shows this meanwhile, inside the layout.
+            hydrateFallbackElement: <DetailLoading />,
+            lazy: async () => {
+              const { default: Component } =
+                await import('../features/pokemon/PokemonDetailPage.tsx')
```

```
+              return { Component }
+            },
+          },
          ],
        },
        { path: '*', element: <NotFoundPage /> },
```

If you open a detail page directly (you reload on **/pokedex/3** or someone sends you the link), its code has not been downloaded yet when the router starts. **hydrateFallbackElement** is what it shows meanwhile, inside the layout; without it, React Router shows nothing and warns in the console that it is missing:

apps/web/src/features/pokemon/DetailLoading.tsx

```tsx
import { useI18n } from '../../i18n/i18nContext.ts'

/** What the detail page shows while its code downloads, when someone opens it directly.
*/
export function DetailLoading() {
  const { t } = useI18n()
  return <output className="block">{t.detail.loading}</output>
}
```

The web app adds Zod as its own dependency: a workspace package's dependencies are not visible from another package, and **api.ts** uses it directly to validate the favorites list.

apps/web/package.json

```
--- a/apps/web/package.json
+++ b/apps/web/package.json
@@ -16,6 +16,7 @@
     "react": "19.3.1",
     "react-dom": "19.3.1",
     "react-router": "8.5.1",
+    "zod": "4.6.7",
     "zustand": "5.1.0"
   },
   "devDependencies": {
```

The new texts:

apps/web/src/i18n/es.ts

```
--- a/apps/web/src/i18n/es.ts
+++ b/apps/web/src/i18n/es.ts
@@ -61,6 +61,34 @@
     pageOf: (page: number, pages: number) => `Página ${page} de ${pages}`,
     results: (total: number) => (total === 1 ? '1 Pokémon' : `${total} Pokémon`),
```

```
   },
+  detail: {
+    back: 'Volver a la Pokédex',
+    loading: 'Cargando la ficha…',
+    error: 'No se pudo cargar la ficha.',
+    notFound: 'Ese Pokémon no existe.',
+    size: (meters: number, kilos: number) => `Altura ${meters} m · Peso ${kilos} kg`,
+    favorite: 'Favorito',
+    favoriteFailed: 'No se pudo guardar el favorito. Inténtalo de nuevo.',
+    compare: 'Comparar',
+    tabs: 'Secciones de la ficha',
+    stats: 'Estadísticas',
+    moves: 'Movimientos',
+    filterMoves: 'Filtrar movimientos',
+  },
+  stats: {
+    hp: 'PS',
+    attack: 'Ataque',
+    defense: 'Defensa',
+    specialAttack: 'Ataque especial',
+    specialDefense: 'Defensa especial',
+    speed: 'Velocidad',
+  },
+  compare: {
+    title: 'Comparación',
+    clear: 'Quitar todos',
+    pickAnother: 'Elige otro Pokémon en su ficha para comparar.',
+    stat: 'Estadística',
+  },
   notFound: { title: 'Esta página no existe', back: 'Volver al inicio' },
   routeError: { title: 'Algo salió mal', retry: 'Recargar la página' },
   footer: {
```

apps/web/src/i18n/en.ts

```
--- a/apps/web/src/i18n/en.ts
+++ b/apps/web/src/i18n/en.ts
@@ -64,6 +64,34 @@
     pageOf: (page: number, pages: number) => `Page ${page} of ${pages}`,
     results: (total: number) => (total === 1 ? '1 Pokémon' : `${total} Pokémon`),
   },
+  detail: {
+    back: 'Back to the Pokédex',
+    loading: 'Loading…',
+    error: 'Could not load this Pokémon.',
+    notFound: 'That Pokémon does not exist.',
+    size: (meters: number, kilos: number) => `Height ${meters} m · Weight ${kilos} kg`,
+    favorite: 'Favorite',
+    favoriteFailed: 'Could not save the favorite. Please try again.',
+    compare: 'Compare',
```

```
+    tabs: 'Sections',
+    stats: 'Stats',
+    moves: 'Moves',
+    filterMoves: 'Filter moves',
+  },
+  stats: {
+    hp: 'HP',
+    attack: 'Attack',
+    defense: 'Defense',
+    specialAttack: 'Special attack',
+    specialDefense: 'Special defense',
+    speed: 'Speed',
+  },
+  compare: {
+    title: 'Comparison',
+    clear: 'Remove all',
+    pickAnother: 'Pick another Pokémon on its page to compare.',
+    stat: 'Stat',
+  },
   notFound: { title: 'This page does not exist', back: 'Back to the start' },
   routeError: { title: 'Something went wrong', retry: 'Reload the page' },
   footer: {
```

The tests

The handlers add a Pokémon's detail and some in-memory favorites, which each test starts however it wants:

apps/web/src/test/handlers.ts

```
--- a/apps/web/src/test/handlers.ts
+++ b/apps/web/src/test/handlers.ts
@@ -40,3 +40,26 @@
 })

 export const types = http.get('/api/types', () => HttpResponse.json(POKEMON_TYPES))
+
+export const pokemonDetail = http.get('/api/pokemon/:id', ({ params }) => {
+  const pokemon = POKEDEX.find((p) => p.id === Number(params.id))
+  return pokemon
+    ? HttpResponse.json(pokemon)
+    : HttpResponse.json({ error: { code: 'POKEMON_NOT_FOUND', message: 'x' } }, {
status: 404 })
+})
+
+/** In-memory favorites for one test: GET lists them, PUT and DELETE change them. */
+export function favoritesHandlers(initial: number[] = []) {
+  let favorites = [...initial]
+  return [
```

```
+    http.get('/api/favorites', () => HttpResponse.json(favorites)),
+    http.put('/api/favorites/:id', ({ params }) => {
+      favorites = [...new Set([...favorites, Number(params.id)])]
+      return new HttpResponse(null, { status: 204 })
+    }),
+    http.delete('/api/favorites/:id', ({ params }) => {
+      favorites = favorites.filter((id) => id !== Number(params.id))
+      return new HttpResponse(null, { status: 204 })
+    }),
+  ]
+}
```

apps/web/src/features/pokemon/PokemonDetailPage.test.tsx

```
import { screen, waitFor, within } from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import { delay } from 'msw'
import { http, HttpResponse } from 'msw/http'
import { beforeEach, describe, expect, it } from 'vitest'
import {
  favoritesHandlers,
  pokemonDetail,
  pokemonList,
  signedIn,
  types,
} from '../../test/handlers.ts'
import { renderApp } from '../../test/renderApp.tsx'
import { server } from '../../test/server.ts'
import { useCompare } from './compare.ts'

describe('Ficha', () => {
  beforeEach(() => {
    server.use(signedIn, pokemonDetail, ...favoritesHandlers())
    useCompare.setState({ ids: [] })
  })

  it('se carga de forma diferida y muestra las estadísticas', async () => {
    renderApp('/pokedex/3')
    expect(await screen.findByRole('heading', { name: 'lagunix' })).toBeInTheDocument()
    expect(screen.getByRole('tabpanel', { name: 'Estadísticas'
})).toHaveTextContent('Velocidad41')
  })

  it('la pestaña oculta conserva lo escrito (Activity)', async () => {
    renderApp('/pokedex/3')
    await userEvent.click(await screen.findByRole('tab', { name: 'Movimientos' }))
    await userEvent.type(screen.getByLabelText('Filtrar movimientos'), 'bur')
    expect(screen.getByRole('tabpanel', { name: 'Movimientos'
})).toHaveTextContent('burbuja')
    await userEvent.click(screen.getByRole('tab', { name: 'Estadísticas' }))
```

```
    await userEvent.click(screen.getByRole('tab', { name: 'Movimientos' }))
    expect(screen.getByLabelText('Filtrar movimientos')).toHaveValue('bur')
  })

  it('las flechas del teclado cambian de pestaña y llevan el foco', async () => {
    renderApp('/pokedex/3')
    const stats = await screen.findByRole('tab', { name: 'Estadísticas' })
    stats.focus()
    await userEvent.keyboard('{ArrowRight}')
    const moves = screen.getByRole('tab', { name: 'Movimientos' })
    expect(moves).toHaveFocus()
    expect(moves).toHaveAttribute('aria-selected', 'true')
    expect(stats).toHaveAttribute('tabindex', '-1')
  })

  it('el favorito cambia al instante y se guarda', async () => {
    server.use(
      http.put('/api/favorites/:id', async () => {
        await delay(150)
        return new HttpResponse(null, { status: 204 })
      }),
    )
    renderApp('/pokedex/3')
    const button = await screen.findByRole('button', { name: 'Favorito' })
    await waitFor(() => expect(button).toBeEnabled()) // waits for the favorites to
arrive
    await userEvent.click(button)
    // the button looks checked before the server answers
    expect(screen.getByRole('button', { name: 'Favorito' })).toHaveAttribute('aria-
pressed', 'true')
  })

  it('si el servidor rechaza el favorito, vuelve al estado real y lo avisa', async () =>
{
    server.use(
      http.put('/api/favorites/:id', () =>
        HttpResponse.json({ error: { code: 'UNAVAILABLE', message: 'x' } }, { status:
503 }),
      ),
    )
    renderApp('/pokedex/3')
    const button = await screen.findByRole('button', { name: 'Favorito' })
    await waitFor(() => expect(button).toBeEnabled())
    await userEvent.click(button)
    expect(await screen.findByRole('alert')).toHaveTextContent('No se pudo guardar el
favorito.')
    expect(screen.getByRole('button', { name: 'Favorito' })).toHaveAttribute(
      'aria-pressed',
      'false',
    )
```

```
  })

  it('un Pokémon que no existe muestra un mensaje y el camino de vuelta', async () => {
    renderApp('/pokedex/999')
    const alert = await screen.findByRole('alert')
    expect(alert).toHaveTextContent('Ese Pokémon no existe.')
    expect(within(alert).getByRole('link', { name: 'Volver a la Pokédex'
})).toBeInTheDocument()
  })

  it('comparar dos Pokémon muestra sus estadísticas lado a lado en la Pokédex', async ()
=> {
    server.use(types, pokemonList)
    useCompare.setState({ ids: [3, 4] })
    renderApp('/pokedex')
    const table = await screen.findByRole('table')
    expect(within(table).getByRole('columnheader', { name: 'lagunix'
})).toBeInTheDocument()
    expect(within(table).getByRole('row', { name: 'Velocidad 41 88'
})).toBeInTheDocument()
  })

  it('desde la tarjeta de la Pokédex se llega a la Ficha', async () => {
    server.use(types, pokemonList)
    renderApp('/pokedex')
    await userEvent.click(await screen.findByRole('link', { name: 'chispon' }))
    expect(await screen.findByRole('heading', { name: 'chispon', level: 1
})).toBeInTheDocument()
  })
})
```

MSW's **`delay`** delays a response, to check what you see while it is saving; **`useCompare.setState`** empties the comparison tool before each test, because the Zustand store is a module and survives between tests.

Test and use

```
pnpm install
pnpm test
```

Vitest should finish with **`Tests 52 passed (52)`**. In the app, open a Pokémon from the Pokédex, go back, and open it again: the second time, the image travels from the card to the detail page. Mark it as a favorite, switch tabs with the mouse and with the arrow keys, type in the moves filter, and go back. Choose two to compare and go back to the Pokédex: the table is at the top.

```
git add .
git commit -m "paso 08: la ficha"
```

Download the code for step 08.

Step 09: The Team Builder

In this step you will build the team of six and the two-step form that adds and edits its members.

The new files go in **apps/web/src/features/team/**.

The data

apps/web/src/features/team/api.ts

```
import { TeamMemberSchema, type TeamMember, type TeamMemberInput } from
'@dex506/contracts'
import { api } from '../../shared/api/client.ts'

export async function fetchTeam(): Promise<TeamMember[]> {
  const { data } = await api.get('/team')
  return TeamMemberSchema.array().parse(data)
}

export async function addMember(input: TeamMemberInput): Promise<TeamMember> {
  const { data } = await api.post('/team/members', input)
  return TeamMemberSchema.parse(data)
}

export async function updateMember(id: string, input: TeamMemberInput):
Promise<TeamMember> {
  const { data } = await api.put(`/team/members/${id}`, input)
  return TeamMemberSchema.parse(data)
}

export async function removeMember(id: string): Promise<void> {
  await api.delete(`/team/members/${id}`)
}
```

apps/web/src/features/team/queries.ts

```
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'
import type { TeamMemberInput } from '@dex506/contracts'
import { addMember, fetchTeam, removeMember, updateMember } from './api.ts'

export const TEAM_KEY = ['team'] as const
```

```
export function useTeam() {
  return useQuery({ queryKey: TEAM_KEY, queryFn: fetchTeam })
}

/** Saves a member (new or edited). Once done, the cached team is stale: it gets
invalidated. */
export function useSaveMember(memberId?: string) {
  const queryClient = useQueryClient()
  return useMutation({
    mutationFn: (input: TeamMemberInput) =>
      memberId ? updateMember(memberId, input) : addMember(input),
    onSuccess: () => queryClient.invalidateQueries({ queryKey: TEAM_KEY }),
  })
}

export function useRemoveMember() {
  const queryClient = useQueryClient()
  return useMutation({
    mutationFn: removeMember,
    onSuccess: () => queryClient.invalidateQueries({ queryKey: TEAM_KEY }),
  })
}
```

Saving is a mutation, and when it succeeds, the team stored in the cache is out of date: it gets invalidated, as in "Mutations and invalidation". **`useSaveMember`** works for adding and for editing: with a **`memberId`**, it edits.

The form schema

apps/web/src/features/team/memberForm.ts

```
import { MAX_MOVES, TeamMemberInputSchema } from '@dex506/contracts'
import { z } from 'zod'

/**
 * The form's schema starts from the API contract (TeamMemberInputSchema): the rules are
the same
 * in the browser and on the server. Only the shape of the moves changes, because
useFieldArray needs
 * objects ({ name }) rather than plain strings.
 */
export const MemberFormSchema = TeamMemberInputSchema.omit({ pokemonId: true, moves:
true }).extend(
  {
    moves: z
      .array(z.object({ name: z.string().min(1, 'Elige un movimiento') }))
      .min(1, 'Elige al menos un movimiento')
```

```
      .max(MAX_MOVES, `Como mucho ${MAX_MOVES} movimientos`),
  },
)

export type MemberFormInput = z.input<typeof MemberFormSchema>
export type MemberFormValues = z.output<typeof MemberFormSchema>
```

The form starts from the contract's schema, **`TeamMemberInputSchema`**: the nickname and level rules are the same in the browser and in the API. Only the shape of the moves changes, because **`useFieldArray`** needs objects, not loose strings. **`z.input`** and **`z.output`** are the type of the data before and after validation. Here they match, but **`useForm`** takes them separately: if the schema transformed a value, for example a string into a number, each side would have its correct type.

The form

apps/web/src/features/team/MemberForm.tsx

```
import { zodResolver } from '@hookform/resolvers/zod'
import { useEffect, useRef, useState } from 'react'
import { useFieldArray, useForm } from 'react-hook-form'
import { MAX_MOVES, type ApiError, type Pokemon, type TeamMemberInput } from
'@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { MemberFormSchema, type MemberFormInput, type MemberFormValues } from
'./memberForm.ts'

type Props = {
  pokemon: Pokemon
  defaultValues?: MemberFormInput
  /** Saves and returns the ApiError if the server rejected it, or null if it went well.
*/
  onSave: (input: TeamMemberInput) => Promise<ApiError | null>
}

const inputClass =
  'w-full rounded-md border border-stone-300 px-3 py-2 aria-invalid:border-accent-600
dark:border-stone-700 dark:bg-stone-900'

function FieldError({ id, message }: { id: string; message: string | undefined }) {
  if (!message) return null
  return (
    <p id={id} className="text-sm text-accent-600">
      {message}
    </p>
  )
}

export function MemberForm({ pokemon, defaultValues, onSave }: Props) {
```

```
  const { t } = useI18n()
  const [step, setStep] = useState<1 | 2>(1)
  const [formError, setFormError] = useState<string | null>(null)

  const {
    register,
    control,
    handleSubmit,
    trigger,
    setError,
    setFocus,
    formState: { errors, isSubmitting },
  } = useForm<MemberFormInput, unknown, MemberFormValues>({
    resolver: zodResolver(MemberFormSchema),
    defaultValues: defaultValues ?? { nickname: '', level: 5, moves: [{ name: '' }] },
  })
  const moves = useFieldArray({ control, name: 'moves' })

  // When the step changes, the button that was just pressed disappears: the focus goes
to the
  // step's first field instead of getting lost. Not on the first render, only on a
change.
  const shownStep = useRef(step)
  useEffect(() => {
    if (shownStep.current === step) return
    shownStep.current = step
    setFocus(step === 2 ? 'moves.0.name' : 'nickname')
  }, [step, setFocus])

  async function next() {
    // Before moving to step 2, only step 1's fields are validated.
    if (await trigger(['nickname', 'level'])) setStep(2)
  }

  async function submit(values: MemberFormValues) {
    setFormError(null)
    const apiError = await onSave({
      pokemonId: pokemon.id,
      nickname: values.nickname,
      level: values.level,
      moves: values.moves.map((move) => move.name),
    })
    if (!apiError) return
    const fields = apiError.error.fields ?? {}
    // Each server error goes to its field, as if the form had found it.
    for (const [field, message] of Object.entries(fields)) {
      if (field === 'nickname' || field === 'level') {
        setError(field, { message })
        setStep(1)
      } else if (field === 'moves') {
```

```
        setError('moves', { message })
      }
    }
    if (Object.keys(fields).length === 0) setFormError(apiError.error.message)
  }

  // An empty move fails on its own field (moves.N.name): the first such message shows
under the list.
  const moveError = moves.fields
    .map((_, index) => errors.moves?.[index]?.name?.message)
    .find(Boolean)
  const movesError = errors.moves?.message ?? errors.moves?.root?.message ?? moveError

  return (
    <form onSubmit={handleSubmit(submit)} noValidate className="space-y-4">
      <p className="text-sm text-stone-600 dark:text-stone-400">{t.team.stepOf(step, 2)}
</p>

      <div hidden={step !== 1} className="space-y-3">
        <div className="space-y-1">
          <label className="block space-y-1">
            <span className="text-sm font-medium">{t.team.nickname}</span>
            <input
              {...register('nickname')}
              aria-invalid={errors.nickname ? true : undefined}
              aria-describedby={errors.nickname ? 'nickname-error' : undefined}
              className={inputClass}
            />
          </label>
          {/* Outside the <label>: otherwise the error would become part of the field's
name. */}
          <FieldError id="nickname-error" message={errors.nickname?.message} />
        </div>
        <div className="space-y-1">
          <label className="block space-y-1">
            <span className="text-sm font-medium">{t.team.level}</span>
            <input
              type="number"
              {...register('level', { valueAsNumber: true })}
              aria-invalid={errors.level ? true : undefined}
              aria-describedby={errors.level ? 'level-error' : undefined}
              className={inputClass}
            />
          </label>
          {/* Outside the <label>: otherwise the error would become part of the field's
name. */}
          <FieldError id="level-error" message={errors.level?.message} />
        </div>
        <button
          type="button"
```

```
            onClick={() => void next()}
            className="rounded-md bg-brand-700 px-4 py-2 text-white"
          >
            {t.team.next}
          </button>
        </div>

        <fieldset
          hidden={step !== 2}
          className="space-y-3"
          aria-describedby={movesError ? 'moves-error' : undefined}
        >
          <legend className="text-sm font-medium">{t.team.moves(MAX_MOVES)}</legend>
          {moves.fields.map((field, index) => (
            <div key={field.id} className="flex gap-2">
              <label className="flex-1">
                <span className="sr-only">{t.team.moveN(index + 1)}</span>
                <select
                  {...register(`moves.${index}.name`)}
                  aria-invalid={errors.moves?.[index]?.name ? true : undefined}
                  className={inputClass}
                >
                  <option value="">{t.team.pickMove}</option>
                  {pokemon.moves.map((move) => (
                    <option key={move} value={move}>
                      {move}
                    </option>
                  ))}
                </select>
              </label>
              <button
                type="button"
                onClick={() => moves.remove(index)}
                disabled={moves.fields.length === 1}
                aria-label={t.team.removeMove(index + 1)}
                className="rounded-md border border-stone-300 px-3 dark:border-stone-700"
              >
                ×
              </button>
            </div>
          ))}
          {moves.fields.length < MAX_MOVES && (
            <button
              type="button"
              onClick={() => moves.append({ name: '' })}
              className="text-sm underline"
            >
              {t.team.addMove}
            </button>
          )}
```

```
        <FieldError id="moves-error" message={movesError} />
        <div className="flex gap-2">
          <button
            type="button"
            onClick={() => setStep(1)}
            className="rounded-md border border-stone-300 px-4 py-2 dark:border-stone-
700"
          >
            {t.team.back}
          </button>
          <button
            type="submit"
            disabled={isSubmitting}
            className="rounded-md bg-brand-700 px-4 py-2 text-white disabled:opacity-60"
          >
            {isSubmitting ? t.team.saving : t.team.save}
          </button>
        </div>
      </fieldset>

      {formError && (
        <p role="alert" className="text-sm text-accent-600">
          {formError}
        </p>
      )}
    </form>
  )
}
```

It is React Hook Form with Zod, from "Forms", with three new things:

- **Two steps.** The first asks for the nickname and the level; the second, the moves. **`trigger`** validates only the first step's fields before moving on. The step you do not see stays with **`hidden`**, without unmounting: what you typed is kept when you go back. When the step changes, the button you just pressed disappears, so the effect with **`setFocus`** moves the focus to the first field of the new step, so the keyboard is not left stranded. The **`shownStep`** ref makes that happen only on a step change, not on mount.
- **Dynamic moves.** **`useFieldArray`** manages a list of fields that grows and shrinks: **`append`** adds a move, **`remove`** removes one, and each field has its own **`id`** for its **`key`**.
- **An empty move** fails on its own field, **`moves.N.name`**: **`moveError`** takes the first of those messages to show it below the list, and each empty select is left with **`aria-invalid`**.
- **Server errors go to their field.** **`onSave`** returns the **`ApiError`** if the API rejected the member. If it carries per-field errors, each one goes to its field with **`setError`**, and an error on the nickname or the level goes back to the first step so it can be seen. If it carries no fields, like a full team, it is shown as a form alert.

The pages

apps/web/src/features/team/MemberPages.tsx

```
import { Link, useNavigate, useParams } from 'react-router'
import type { TeamMemberInput } from '@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { toApiError } from '../../shared/api/client.ts'
import { usePokemon } from '../pokemon/queries.ts'
import { MemberForm } from './MemberForm.tsx'
import { useSaveMember, useTeam } from './queries.ts'

/** A member's form: a new one (empty `memberId`) or an edit. */
function MemberEditor({ pokemonId, memberId }: { pokemonId: number; memberId?: string })
{
  const { t } = useI18n()
  const navigate = useNavigate()
  const pokemon = usePokemon(pokemonId)
  const team = useTeam()
  const save = useSaveMember(memberId)
  const member = memberId ? team.data?.find((candidate) => candidate.id === memberId) :
undefined

  if (pokemon.isPending || (memberId && team.isPending))
    return <output className="block">{t.team.loading}</output>
  if (pokemon.isError || (memberId && !member)) return <p role="alert">{t.team.notFound}
</p>

  async function onSave(input: TeamMemberInput) {
    try {
      await save.mutateAsync(input)
      navigate('/team')
      return null
    } catch (error) {
      return toApiError(error)
    }
  }

  return (
    <section className="mx-auto max-w-md space-y-4">
      <h1 className="text-2xl font-bold">
        {member ? t.team.editTitle(member.nickname) :
t.team.addTitle(pokemon.data.name)}
      </h1>
      <MemberForm
        pokemon={pokemon.data}
        onSave={onSave}
        defaultValues={
          member && {
            nickname: member.nickname,
```

```
            level: member.level,
            moves: member.moves.map((name) => ({ name })),
          }
        }
      />
      <Link to="/team" className="text-sm underline">
        {t.team.cancel}
      </Link>
    </section>
  )
}

export function AddMemberPage() {
  return <MemberEditor pokemonId={Number(useParams().pokemonId)} />
}

export function EditMemberPage() {
  const { memberId } = useParams()
  const team = useTeam()
  const member = team.data?.find((candidate) => candidate.id === memberId)
  const { t } = useI18n()
  if (team.isPending) return <output className="block">{t.team.loading}</output>
  if (!member) return <p role="alert">{t.team.notFound}</p>
  return <MemberEditor pokemonId={member.pokemonId} memberId={member.id} />
}
```

`MemberEditor` is the same for adding and editing: it loads the Pokémon and, when editing, the member, and passes their data as **`defaultValues`**. **`navigate('/team')`** goes in **`onSave`**, the save handler: that is where you know it was saved.

apps/web/src/features/team/TeamPage.tsx

```
import { Link } from 'react-router'
import { MAX_TEAM, averageLevel } from '@dex506/contracts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { useRemoveMember, useTeam } from './queries.ts'

export function TeamPage() {
  const { t } = useI18n()
  const team = useTeam()
  const remove = useRemoveMember()

  if (team.isPending) return <output className="block">{t.team.loading}</output>
  if (team.isError) return <p role="alert">{t.team.error}</p>

  return (
    <section className="space-y-4">
      <h1 className="text-3xl font-bold">{t.team.title}</h1>
      {/* Derived data: computed while rendering, not stored in separate state. */}
```

```
      <p>{t.team.summary(team.data.length, MAX_TEAM, averageLevel(team.data))}</p>
      {team.data.length === 0 ? (
        <p>
          {t.team.empty}{' '}
          <Link to="/pokedex" className="underline">
            {t.nav.pokedex}
          </Link>
        </p>
      ) : (
        <ul className="grid gap-3 sm:grid-cols-2">
          {team.data.map((member) => (
            <li
              key={member.id}
              className="space-y-1 rounded-xl border border-stone-200 p-3 dark:border-
stone-800"
            >
              <h2 className="font-semibold">{member.nickname}</h2>
              <p className="text-sm">{t.team.levelN(member.level)}</p>
              <p className="text-sm text-stone-600 dark:text-stone-400">
                {member.moves.join(' · ')}
              </p>
              <div className="flex gap-3 text-sm">
                <Link to={`/team/${member.id}`} className="underline">
                  {t.team.edit(member.nickname)}
                </Link>
                <button
                  type="button"
                  onClick={() => {
                    // Enabled while deleting, so it keeps the focus; a second click
waits.
                    if (!remove.isPending) remove.mutate(member.id)
                  }}
                  className="underline"
                >
                  {t.team.remove(member.nickname)}
                </button>
              </div>
            </li>
          ))}
        </ul>
      )}
      {remove.isError && (
        <p role="alert" className="text-accent-600">
          {t.team.removeFailed}
        </p>
      )}
    </section>
  )
}
```

The summary, with the average level, is derived data: it is computed during render with the contract's **`averageLevel`**. The remove button is not disabled while removing (it would lose the keyboard focus): it ignores a second click, and if the API fails, the page says so.

The changes

The team routes:

apps/web/src/app/routes.tsx

```
--- a/apps/web/src/app/routes.tsx
+++ b/apps/web/src/app/routes.tsx
@@ -3,6 +3,8 @@
 import { RequireAuth } from '../features/auth/RequireAuth.tsx'
 import { PokedexPage } from '../features/pokedex/PokedexPage.tsx'
 import { DetailLoading } from '../features/pokemon/DetailLoading.tsx'
+import { AddMemberPage, EditMemberPage } from '../features/team/MemberPages.tsx'
+import { TeamPage } from '../features/team/TeamPage.tsx'
 import { HomePage } from './HomePage.tsx'
 import { Layout } from './Layout.tsx'
 import { NotFoundPage } from './NotFoundPage.tsx'
@@ -34,6 +36,9 @@
               return { Component }
             },
           },
+          { path: 'team', element: <TeamPage /> },
+          { path: 'team/add/:pokemonId', element: <AddMemberPage /> },
+          { path: 'team/:memberId', element: <EditMemberPage /> },
         ],
       },
       { path: '*', element: <NotFoundPage /> },
```

The link in the navigation:

apps/web/src/app/Layout.tsx

```
--- a/apps/web/src/app/Layout.tsx
+++ b/apps/web/src/app/Layout.tsx
@@ -26,6 +26,9 @@
              <NavLink to="/pokedex" className={linkClass}>
                {t.nav.pokedex}
              </NavLink>
+             <NavLink to="/team" className={linkClass}>
+               {t.nav.team}
+             </NavLink>
            </nav>
            {/* Explicit label (htmlFor + id): wrapped in the <label>, the <select>'s
name would
                include the selected option. */}
```

The detail page offers to add the Pokémon to the team:

apps/web/src/features/pokemon/PokemonDetailPage.tsx

```
--- a/apps/web/src/features/pokemon/PokemonDetailPage.tsx
+++ b/apps/web/src/features/pokemon/PokemonDetailPage.tsx
@@ -109,6 +109,12 @@
            <p className="text-sm">{t.detail.size(data.height / 10, data.weight / 10)}
</p>
            <div className="flex flex-wrap items-start gap-2">
              <FavoriteButton pokemonId={data.id} />
+             <Link
+               to={`/team/add/${data.id}`}
+               className="rounded-md bg-brand-700 px-3 py-1 text-sm text-white"
+             >
+               {t.detail.addToTeam}
+             </Link>
              <button
                type="button"
                aria-pressed={comparing}
```

Zod's validation messages come out in English unless you configure its language. The language provider configures it along with everything else:

apps/web/src/i18n/I18nProvider.tsx

```
--- a/apps/web/src/i18n/I18nProvider.tsx
+++ b/apps/web/src/i18n/I18nProvider.tsx
@@ -1,4 +1,5 @@
 import { useEffect, useState, type ReactNode } from 'react'
+import { z } from 'zod'
 import { en } from './en.ts'
 import { es, type Messages } from './es.ts'
 import { I18nContext, type Locale } from './i18nContext.ts'
```

```
@@ -17,6 +18,8 @@
   useEffect(() => {
     localStorage.setItem(STORAGE_KEY, locale)
     document.documentElement.lang = locale
+    // Zod's validation messages (forms) in the same language as the interface.
+    z.config(locale === 'es' ? z.locales.es() : z.locales.en())
   }, [locale])

   return <I18nContext value={{ locale, setLocale, t: MESSAGES[locale] }}>{children}
</I18nContext>
```

React Hook Form and its Zod adapter, with the same **`minimumReleaseAge`** exception from step 05 for the adapter:

apps/web/package.json

```
--- a/apps/web/package.json
+++ b/apps/web/package.json
@@ -11,10 +11,12 @@
   },
   "dependencies": {
     "@dex506/contracts": "workspace:*",
+    "@hookform/resolvers": "7.0.7",
     "@tanstack/react-query": "5.108.0",
     "axios": "1.21.1",
     "react": "19.3.1",
     "react-dom": "19.3.1",
+    "react-hook-form": "7.92.0",
     "react-router": "8.5.1",
     "zod": "4.6.7",
     "zustand": "5.1.0"
```

pnpm-workspace.yaml

```
--- a/pnpm-workspace.yaml
+++ b/pnpm-workspace.yaml
@@ -5,3 +5,4 @@
   - '@tanstack/query-core@5.108.0'
   - '@tanstack/react-query@5.108.0'
   - msw@3.0.7
+  - '@hookform/resolvers@7.0.7'
```

The new texts:

apps/web/src/i18n/es.ts

```
--- a/apps/web/src/i18n/es.ts
+++ b/apps/web/src/i18n/es.ts
```

```
@@ -73,6 +73,7 @@
     tabs: 'Secciones de la ficha',
     stats: 'Estadísticas',
     moves: 'Movimientos',
+    addToTeam: 'Agregar al equipo',
     filterMoves: 'Filtrar movimientos',
   },
   stats: {
@@ -89,6 +90,34 @@
     pickAnother: 'Elige otro Pokémon en su ficha para comparar.',
     stat: 'Estadística',
   },
+  team: {
+    title: 'Tu equipo',
+    loading: 'Cargando el equipo…',
+    error: 'No se pudo cargar el equipo.',
+    notFound: 'Ese miembro no existe.',
+    empty: 'Todavía no tienes miembros. Elige un Pokémon en la',
+    summary: (size: number, max: number, average: number) =>
+      `${size} de ${max} miembros · nivel promedio ${average}`,
+    levelN: (level: number) => `Nivel ${level}`,
+    edit: (name: string) => `Editar a ${name}`,
+    remove: (name: string) => `Quitar a ${name}`,
+    removeFailed: 'No se pudo quitar al miembro. Inténtalo de nuevo.',
+    addTitle: (name: string) => `Agregar a ${name}`,
+    editTitle: (name: string) => `Editar a ${name}`,
+    stepOf: (step: number, total: number) => `Paso ${step} de ${total}`,
+    nickname: 'Apodo (máximo 12 caracteres)',
+    level: 'Nivel (1 a 100)',
+    moves: (max: number) => `Movimientos (máximo ${max})`,
+    moveN: (n: number) => `Movimiento ${n}`,
+    pickMove: 'Elige un movimiento',
+    addMove: 'Agregar otro movimiento',
+    removeMove: (n: number) => `Quitar el movimiento ${n}`,
+    next: 'Siguiente',
+    back: 'Atrás',
+    save: 'Guardar',
+    saving: 'Guardando…',
+    cancel: 'Cancelar',
+  },
   notFound: { title: 'Esta página no existe', back: 'Volver al inicio' },
   routeError: { title: 'Algo salió mal', retry: 'Recargar la página' },
   footer: {
```

apps/web/src/i18n/en.ts

```
--- a/apps/web/src/i18n/en.ts
+++ b/apps/web/src/i18n/en.ts
@@ -76,6 +76,7 @@
     tabs: 'Sections',
```

```
     stats: 'Stats',
     moves: 'Moves',
+    addToTeam: 'Add to team',
     filterMoves: 'Filter moves',
   },
   stats: {
@@ -92,6 +93,34 @@
     pickAnother: 'Pick another Pokémon on its page to compare.',
     stat: 'Stat',
   },
+  team: {
+    title: 'Your team',
+    loading: 'Loading your team…',
+    error: 'Could not load your team.',
+    notFound: 'That member does not exist.',
+    empty: 'You have no members yet. Pick a Pokémon in the',
+    summary: (size: number, max: number, average: number) =>
+      `${size} of ${max} members · average level ${average}`,
+    levelN: (level: number) => `Level ${level}`,
+    edit: (name: string) => `Edit ${name}`,
+    remove: (name: string) => `Remove ${name}`,
+    removeFailed: 'Could not remove the member. Please try again.',
+    addTitle: (name: string) => `Add ${name}`,
+    editTitle: (name: string) => `Edit ${name}`,
+    stepOf: (step: number, total: number) => `Step ${step} of ${total}`,
+    nickname: 'Nickname (up to 12 characters)',
+    level: 'Level (1 to 100)',
+    moves: (max: number) => `Moves (up to ${max})`,
+    moveN: (n: number) => `Move ${n}`,
+    pickMove: 'Pick a move',
+    addMove: 'Add another move',
+    removeMove: (n: number) => `Remove move ${n}`,
+    next: 'Next',
+    back: 'Back',
+    save: 'Save',
+    saving: 'Saving…',
+    cancel: 'Cancel',
+  },
   notFound: { title: 'This page does not exist', back: 'Back to the start' },
   routeError: { title: 'Something went wrong', retry: 'Reload the page' },
   footer: {
```

The tests

An in-memory team per test, with the four operations:

apps/web/src/test/handlers.ts

```
--- a/apps/web/src/test/handlers.ts
```

```
+++ b/apps/web/src/test/handlers.ts
@@ -1,7 +1,7 @@
 import { http, HttpResponse } from 'msw/http'
 import type { User } from '@dex506/contracts'
 import fixtures from '../../../../fixtures/pokemon.json'
-import { POKEMON_TYPES, type Pokemon } from '@dex506/contracts'
+import { POKEMON_TYPES, type Pokemon, type TeamMember } from '@dex506/contracts'

 export const ANA: User = { id: 'u1', email: 'ana@dex506.test', trainerName: 'Ana' }

@@ -63,3 +63,29 @@
     }),
   ]
 }
+
+/** The team in memory for one test: GET lists it, POST/PUT/DELETE change it. */
+export function teamHandlers(initial: TeamMember[] = []) {
+  let team = [...initial]
+  let next = team.length + 1
+  return [
+    http.get('/api/team', () => HttpResponse.json(team)),
+    http.post('/api/team/members', async ({ request }) => {
+      const member = { ...((await request.json()) as Omit<TeamMember, 'id'>), id:
`m${next++}` }
+      team = [...team, member]
+      return HttpResponse.json(member, { status: 201 })
+    }),
+    http.put('/api/team/members/:id', async ({ request, params }) => {
+      const member = {
+        ...((await request.json()) as Omit<TeamMember, 'id'>),
+        id: String(params.id),
+      }
+      team = team.map((current) => (current.id === member.id ? member : current))
+      return HttpResponse.json(member)
+    }),
+    http.delete('/api/team/members/:id', ({ params }) => {
+      team = team.filter((current) => current.id !== params.id)
+      return new HttpResponse(null, { status: 204 })
+    }),
+  ]
+}
```

apps/web/src/features/team/Team.test.tsx

```tsx
import { screen, waitFor, within } from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import { http, HttpResponse } from 'msw/http'
import { beforeEach, describe, expect, it } from 'vitest'
import { pokemonDetail, signedIn, teamHandlers } from '../../test/handlers.ts'
import { renderApp } from '../../test/renderApp.tsx'
```

```
import { server } from '../../test/server.ts'

const LAGU = { id: 'm1', pokemonId: 3, nickname: 'Lagu', level: 12, moves: ['chorro'] }

async function fillStep1(nickname: string, level: string) {
  const nicknameInput = await screen.findByLabelText('Apodo (máximo 12 caracteres)')
  await userEvent.clear(nicknameInput)
  await userEvent.type(nicknameInput, nickname)
  const levelInput = screen.getByLabelText('Nivel (1 a 100)')
  await userEvent.clear(levelInput)
  await userEvent.type(levelInput, level)
  await userEvent.click(screen.getByRole('button', { name: 'Siguiente' }))
}

describe('taller de equipo', () => {
  beforeEach(() => server.use(signedIn, pokemonDetail, ...teamHandlers()))

  it('agrega un miembro en dos pasos y vuelve al equipo', async () => {
    renderApp('/team/add/3')
    await fillStep1('Lagu', '12')
    await waitFor(() => expect(screen.getByLabelText('Movimiento 1')).toHaveFocus())
    await userEvent.selectOptions(screen.getByLabelText('Movimiento 1'), 'chorro')
    await userEvent.click(screen.getByRole('button', { name: 'Agregar otro movimiento'
}))
    await userEvent.selectOptions(screen.getByLabelText('Movimiento 2'), 'burbuja')
    await userEvent.click(screen.getByRole('button', { name: 'Guardar' }))
    expect(await screen.findByRole('heading', { name: 'Lagu' })).toBeInTheDocument()
    expect(screen.getByText('chorro · burbuja')).toBeInTheDocument()
    expect(screen.getByText('1 de 6 miembros · nivel promedio 12')).toBeInTheDocument()
  })

  it('un movimiento vacío no se guarda y dice por qué', async () => {
    renderApp('/team/add/3')
    await fillStep1('Lagu', '12')
    await userEvent.click(screen.getByRole('button', { name: 'Guardar' }))
    expect(await screen.findByText('Elige un movimiento', { selector: 'p'
})).toBeInTheDocument()
    expect(screen.getByLabelText('Movimiento 1')).toHaveAttribute('aria-invalid',
'true')
  })

  it('si quitar a un miembro falla, lo avisa', async () => {
    server.use(
      http.delete('/api/team/members/:id', () => new HttpResponse(null, { status: 500
})),
      ...teamHandlers([LAGU]),
    )
    renderApp('/team')
    await userEvent.click(await screen.findByRole('button', { name: 'Quitar a Lagu' }))
```

```
    expect(await screen.findByRole('alert')).toHaveTextContent('No se pudo quitar al
miembro.')
  })

  it('no pasa al paso 2 con un apodo largo, y el error queda asociado al campo', async
() => {
    renderApp('/team/add/3')
    await fillStep1('UnApodoDemasiadoLargo', '12')
    const nickname = screen.getByLabelText('Apodo (máximo 12 caracteres)')
    expect(nickname).toHaveAttribute('aria-invalid', 'true')
    expect(nickname).toHaveAccessibleDescription(
      'Demasiado grande: se esperaba que texto tuviera <=12 caracteres',
    )
    expect(screen.getByText('Paso 1 de 2')).toBeInTheDocument()
  })

  it('un error del servidor en un campo se pinta en ese campo', async () => {
    server.use(
      http.post('/api/team/members', () =>
        HttpResponse.json(
          {
            error: {
              code: 'VALIDATION',
              message: 'x',
              fields: { moves: 'lagunix no conoce: brasa' },
            },
          },
          { status: 400 },
        ),
      ),
    )
    renderApp('/team/add/3')
    await fillStep1('Lagu', '12')
    await userEvent.selectOptions(screen.getByLabelText('Movimiento 1'), 'chorro')
    await userEvent.click(screen.getByRole('button', { name: 'Guardar' }))
    expect(await screen.findByText('lagunix no conoce: brasa')).toBeInTheDocument()
  })

  it('con el equipo lleno, el 422 se muestra como alerta', async () => {
    server.use(
      http.post('/api/team/members', () =>
        HttpResponse.json(
          { error: { code: 'TEAM_FULL', message: 'Tu equipo ya tiene 6 miembros' } },
          { status: 422 },
        ),
      ),
    )
    renderApp('/team/add/3')
    await fillStep1('Lagu', '12')
    await userEvent.selectOptions(screen.getByLabelText('Movimiento 1'), 'chorro')
```

```
    await userEvent.click(screen.getByRole('button', { name: 'Guardar' }))
    expect(await screen.findByRole('alert')).toHaveTextContent('Tu equipo ya tiene 6
miembros')
  })

  it('edita un miembro con sus datos cargados y lo quita', async () => {
    server.use(...teamHandlers([LAGU]))
    renderApp('/team/m1')
    expect(await screen.findByLabelText('Apodo (máximo 12
caracteres)')).toHaveValue('Lagu')
    await fillStep1('Laguito', '20')
    await userEvent.click(screen.getByRole('button', { name: 'Guardar' }))
    const card = (await screen.findByRole('heading', { name: 'Laguito'
})).closest('li')!
    await userEvent.click(within(card).getByRole('button', { name: 'Quitar a Laguito'
}))
    expect(await screen.findByText('0 de 6 miembros · nivel promedio
0')).toBeInTheDocument()
  })
})
```

Test and use

```
pnpm install
pnpm test
```

Vitest should finish with **`Tests 59 passed (59)`**. In the app, from a detail page, tap "Agregar al equipo" (add to team): type a nickname that is too long and look at the error on its field; go on to step 2, choose moves, and save. Add six and try a seventh: the API responds with the full-team 422, and the form shows it.

```
git add .
git commit -m "paso 09: el taller de equipo"
```

Download the code for step 09.

Step 10: The Gyms

In this step you will build the eight gyms, the lineup with Redux Toolkit, and the challenge, with the handling of its errors.

Why Redux here

The lineup for each gym is interface state, and Zustand would be enough, as with the view preference. Dex 506 uses Redux Toolkit so you build it once in a real app, as "Architecture, Quality, and Security" says.

The store

apps/web/src/app/store/store.ts

```
import { configureStore } from '@reduxjs/toolkit'
import { useDispatch, useSelector } from 'react-redux'
import { lineupReducer } from '../../features/gyms/lineupSlice.ts'

/** A function and not a single store: each test creates its own, with no state leaking between tests. */
export function makeStore() {
  // configureStore connects the Redux DevTools (if the browser has the extension) and, in
  // development, checks that state is not mutated.
  return configureStore({ reducer: { lineup: lineupReducer } })
}

export type AppStore = ReturnType<typeof makeStore>
export type RootState = ReturnType<AppStore['getState']>
export type AppDispatch = AppStore['dispatch']

export const useAppSelector = useSelector.withTypes<RootState>()
export const useAppDispatch = useDispatch.withTypes<AppDispatch>()
```

- **`makeStore`** is a function, as in "Global State": the app creates one store and each test creates its own.
- **`useAppSelector`** and **`useAppDispatch`** are **`useSelector`** and **`useDispatch`** with the store's types, made once with **`withTypes`**: that way no component has to write **`RootState`**.

The lineup

The gym files go in **`apps/web/src/features/gyms/`**.

apps/web/src/features/gyms/lineupSlice.ts

```
import { createSelector, createSlice, type PayloadAction } from '@reduxjs/toolkit'

export const LINEUP_SIZE = 3

/** The lineup chosen for each gym: gymId → member ids, in the order they were picked. */
type LineupState = Record<string, string[]>

const lineupSlice = createSlice({
```

```
  name: 'lineup',
  initialState: {} as LineupState,
  reducers: {
    // Redux Toolkit uses Immer: this code "mutates" a draft and RTK produces a new
state.
    toggleMember(state, action: PayloadAction<{ gymId: string; memberId: string }>) {
      const { gymId, memberId } = action.payload
      const lineup = state[gymId] ?? []
      if (lineup.includes(memberId)) state[gymId] = lineup.filter((id) => id !==
memberId)
      else if (lineup.length < LINEUP_SIZE) state[gymId] = [...lineup, memberId]
    },
    clearLineup(state, action: PayloadAction<string>) {
      delete state[action.payload]
    },
    /** Removes from every lineup the members who are no longer on the team. */
    keepOnly(state, action: PayloadAction<string[]>) {
      for (const gymId of Object.keys(state)) {
        state[gymId] = (state[gymId] ?? []).filter((id) => action.payload.includes(id))
      }
    },
  },
})

export const { toggleMember, clearLineup, keepOnly } = lineupSlice.actions
export const lineupReducer = lineupSlice.reducer

type WithLineup = { lineup: LineupState }
const EMPTY: string[] = []

/**
 * A gym's lineup. createSelector memoizes the result: as long as that gym's state does
not change,
 * it returns the SAME array, and the component does not re-render because of a new
array with the
 * same contents.
 */
export const selectLineup = createSelector(
  [(state: WithLineup) => state.lineup, (_state: WithLineup, gymId: string) => gymId],
  (lineup, gymId) => lineup[gymId] ?? EMPTY,
)
```

- The state keeps one lineup per gym: an object from **`gymId`** to the list of members, in the order they were chosen.
- **`toggleMember`** adds or removes a member, with a maximum of three. **`keepOnly`** removes from every lineup the members who are no longer on the team.
- **`selectLineup`** is a selector made with **`createSelector`** that takes an argument, the **`gymId`**. **`EMPTY`** is a fixed empty array: if the gym has no lineup, it always returns the same one, not a new one on every

render. To be fair, **`createSelector`** adds little here: the stored array is already always the same one. It is there so you see it in its simplest form; it pays off when the selector computes something new, such as a filter or a total.

The data

apps/web/src/features/gyms/api.ts

```
import { ChallengeResultSchema, GymSchema, type ChallengeResult, type Gym } from
'@dex506/contracts'
import { z } from 'zod'
import { api } from '../../shared/api/client.ts'

export async function fetchGyms(): Promise<Gym[]> {
  const { data } = await api.get('/gyms')
  return GymSchema.array().parse(data)
}

export async function fetchBadges(): Promise<string[]> {
  const { data } = await api.get('/badges')
  return z.array(z.string()).parse(data)
}

export async function challengeGym(gymId: string, memberIds: string[]):
Promise<ChallengeResult> {
  const { data } = await api.post(`/gyms/${gymId}/challenge`, { memberIds })
  return ChallengeResultSchema.parse(data)
}
```

apps/web/src/features/gyms/queries.ts

```
export const GYMS_KEY = ['gyms'] as const
export const BADGES_KEY = ['badges'] as const
```

The two keys go in a file of their own because both pages use them.

The pages

apps/web/src/features/gyms/GymsPage.tsx

```
import { useQuery } from '@tanstack/react-query'
import { Link } from 'react-router'
import { useI18n } from '../../i18n/i18nContext.ts'
import { TypeBadge } from '../../shared/ui/TypeBadge.tsx'
import { fetchBadges, fetchGyms } from './api.ts'
import { BADGES_KEY, GYMS_KEY } from './queries.ts'
```

```tsx
export function GymsPage() {
  const { t } = useI18n()
  const gyms = useQuery({ queryKey: GYMS_KEY, queryFn: fetchGyms, staleTime: Infinity })
  const badges = useQuery({ queryKey: BADGES_KEY, queryFn: fetchBadges })

  if (gyms.isPending) return <output className="block">{t.gyms.loading}</output>
  if (gyms.isError) return <p role="alert">{t.gyms.error}</p>

  return (
    <section className="space-y-4">
      <h1 className="text-3xl font-bold">{t.gyms.title}</h1>
      <p>{t.gyms.badges(badges.data?.length ?? 0, gyms.data.length)}</p>
      <ul className="grid gap-3 sm:grid-cols-2">
        {gyms.data.map((gym) => {
          const won = badges.data?.includes(gym.id) ?? false
          return (
            <li
              key={gym.id}
              className="relative space-y-1 rounded-xl border border-stone-200 p-3
dark:border-stone-800"
            >
              <h2 className="font-semibold">
                <Link to={`/gyms/${gym.id}`} className="after:absolute after:inset-0">
                  {gym.name}
                </Link>
              </h2>
              <p className="text-sm">
                {gym.city} · {t.gyms.minLevel(gym.minAvgLevel)}
              </p>
              <p className="flex items-center gap-2">
                <TypeBadge type={gym.type} />
                {won && (
                  <span className="text-sm font-semibold text-brand-700 dark:text-brand-
100">
                    {t.gyms.won(gym.badge)}
                  </span>
                )}
              </p>
            </li>
          )
        })}
      </ul>
    </section>
  )
}
```

apps/web/src/features/gyms/GymPage.tsx

```tsx
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'
import { useEffect } from 'react'
```

```
import { Link, useParams } from 'react-router'
import { averageLevel } from '@dex506/contracts'
import { useAppDispatch, useAppSelector } from '../../app/store/store.ts'
import { useI18n } from '../../i18n/i18nContext.ts'
import { toApiError } from '../../shared/api/client.ts'
import { TypeBadge } from '../../shared/ui/TypeBadge.tsx'
import { useTeam } from '../team/queries.ts'
import { challengeGym, fetchGyms } from './api.ts'
import { BADGES_KEY, GYMS_KEY } from './queries.ts'
import { LINEUP_SIZE, clearLineup, keepOnly, selectLineup, toggleMember } from
'./lineupSlice.ts'

export function GymPage() {
  const { t } = useI18n()
  const gymId = useParams().id ?? ''
  const queryClient = useQueryClient()
  const dispatch = useAppDispatch()
  const gyms = useQuery({ queryKey: GYMS_KEY, queryFn: fetchGyms, staleTime: Infinity })
  const team = useTeam()
  const lineup = useAppSelector((state) => selectLineup(state, gymId))

  // If a member left the team, it cannot stay in the lineup stored in Redux either.
  useEffect(() => {
    if (team.data) dispatch(keepOnly(team.data.map((member) => member.id)))
  }, [team.data, dispatch])

  const challenge = useMutation({
    mutationFn: () => challengeGym(gymId, lineup),
    onSuccess: (result) => {
      if (result.won) void queryClient.invalidateQueries({ queryKey: BADGES_KEY })
    },
  })

  const gym = gyms.data?.find((candidate) => candidate.id === gymId)
  if (gyms.isPending || team.isPending) return <output className="block">
{t.gyms.loading}</output>
  if (!gym || team.isError) return <p role="alert">{t.gyms.notFound}</p>

  const selected = team.data.filter((member) => lineup.includes(member.id))
  const average = averageLevel(selected)
  const challengeError = challenge.isError ? toApiError(challenge.error).error.message :
null

  return (
    <section className="space-y-5">
      <Link to="/gyms" className="text-sm underline">
        {t.gyms.back}
      </Link>
      <header className="space-y-1">
        <h1 className="text-3xl font-bold">{gym.name}</h1>
```

```
        <p className="flex items-center gap-2">
          {gym.city} <TypeBadge type={gym.type} />
        </p>
        <p className="text-sm">{t.gyms.minLevel(gym.minAvgLevel)}</p>
      </header>

      <fieldset className="space-y-2">
        <legend className="font-semibold">{t.gyms.pickLineup(LINEUP_SIZE)}</legend>
        {team.data.length < LINEUP_SIZE && (
          <p>
            {t.gyms.needMembers(LINEUP_SIZE)}{' '}
            <Link to="/pokedex" className="underline">
              {t.nav.pokedex}
            </Link>
          </p>
        )}
        {team.data.map((member) => {
          const checked = lineup.includes(member.id)
          return (
            <label key={member.id} className="flex items-center gap-2">
              <input
                type="checkbox"
                checked={checked}
                disabled={!checked && lineup.length >= LINEUP_SIZE}
                onChange={() => dispatch(toggleMember({ gymId, memberId: member.id }))}
              />
              {member.nickname} · {t.team.levelN(member.level)}
            </label>
          )
        })}
        <p className="text-sm">{t.gyms.lineupSummary(selected.length, LINEUP_SIZE,
average)}</p>
      </fieldset>

      <div className="flex gap-3">
        <button
          type="button"
          disabled={lineup.length !== LINEUP_SIZE || challenge.isPending}
          onClick={() => challenge.mutate()}
          className="rounded-md bg-brand-700 px-4 py-2 text-white disabled:opacity-50"
        >
          {challenge.isPending ? t.gyms.challenging : t.gyms.challenge}
        </button>
        <button type="button" onClick={() => dispatch(clearLineup(gymId))}
className="underline">
          {t.gyms.clear}
        </button>
      </div>

      {challengeError && (
```

```
        <p role="alert" className="text-accent-600">
          {challengeError}
        </p>
      )}
      {/* The live region is always on the page: a screen reader announces what appears
inside it. */}
      <div aria-live="polite">
        {challenge.data && (
          <section className="space-y-1 rounded-xl border border-stone-200 p-3
dark:border-stone-800">
            <h2 className="font-semibold">{challenge.data.won ? t.gyms.victory :
t.gyms.defeat}</h2>
            <ol className="list-decimal pl-5 text-sm">
              {challenge.data.log.map((line) => (
                <li key={line}>{line}</li>
              ))}
            </ol>
          </section>
        )}
      </div>
    </section>
  )
}
```

- The lineup lives in Redux, and the team, in TanStack Query: each piece of data in its place, as in part 2's "Not all state is the same" table. The effect with **`keepOnly`** keeps the two in agreement when the team changes.
- When the lineup already has three chosen, the other members are disabled.
- The challenge is a mutation. If the API rejects it with a 422, for example because of a low average level, **`toApiError`** brings its message and it is shown as an alert. If you win, the badges are invalidated and the gym list shows it.
- The result appears inside a region with **`aria-live="polite"`** that is always on the page: a screen reader announces what appears inside, without interrupting what it was reading. If the region appeared together with the text, it might not be announced.

The changes

The store is created once in **`main.tsx`** and arrives through **`Providers`**:

apps/web/src/main.tsx

```
--- a/apps/web/src/main.tsx
+++ b/apps/web/src/main.tsx
@@ -5,14 +5,16 @@
 import { Providers } from './app/Providers.tsx'
 import { createQueryClient } from './app/queryClient.ts'
 import { routes } from './app/routes.tsx'
```

```
+import { makeStore } from './app/store/store.ts'
 import './index.css'

 const router = createBrowserRouter(routes)
 const queryClient = createQueryClient()
+const store = makeStore()

 createRoot(document.getElementById('root')!).render(
   <StrictMode>
-    <Providers queryClient={queryClient}>
+    <Providers queryClient={queryClient} store={store}>
       <RouterProvider router={router} />
     </Providers>
   </StrictMode>,
```

apps/web/src/app/Providers.tsx

```
--- a/apps/web/src/app/Providers.tsx
+++ b/apps/web/src/app/Providers.tsx
@@ -1,24 +1,24 @@
 import { QueryClientProvider, type QueryClient } from '@tanstack/react-query'
 import type { ReactNode } from 'react'
+import { Provider as ReduxProvider } from 'react-redux'
 import { AuthProvider } from '../features/auth/AuthProvider.tsx'
 import { I18nProvider } from '../i18n/I18nProvider.tsx'
 import { ThemeProvider } from '../shared/theme/ThemeProvider.tsx'
+import type { AppStore } from './store/store.ts'

-/** Everything the app needs around it: server data, language, theme, and session. */
-export function Providers({
-  queryClient,
-  children,
-}: {
-  queryClient: QueryClient
-  children: ReactNode
-}) {
+type Props = { queryClient: QueryClient; store: AppStore; children: ReactNode }
+
+/** Everything the app needs around it: global state, server data, language, theme, and
session. */
+export function Providers({ queryClient, store, children }: Props) {
   return (
-    <QueryClientProvider client={queryClient}>
-      <I18nProvider>
-        <ThemeProvider>
-          <AuthProvider>{children}</AuthProvider>
-        </ThemeProvider>
-      </I18nProvider>
-    </QueryClientProvider>
+    <ReduxProvider store={store}>
```

```
+      <QueryClientProvider client={queryClient}>
+        <I18nProvider>
+          <ThemeProvider>
+            <AuthProvider>{children}</AuthProvider>
+          </ThemeProvider>
+        </I18nProvider>
+      </QueryClientProvider>
+    </ReduxProvider>
   )
 }
```

The tests build their own store:

apps/web/src/test/renderApp.tsx

```
--- a/apps/web/src/test/renderApp.tsx
+++ b/apps/web/src/test/renderApp.tsx
@@ -3,15 +3,17 @@
 import { createMemoryRouter, RouterProvider } from 'react-router'
 import { Providers } from '../app/Providers.tsx'
 import { routes } from '../app/routes.tsx'
+import { makeStore } from '../app/store/store.ts'

-/** The whole app at one route, with an in-memory router and a fresh data cache for every test. */
+/** The whole app at one route, with an in-memory router, and a fresh cache and store for every test. */
 export function renderApp(path = '/') {
   const router = createMemoryRouter(routes, { initialEntries: [path] })
   const queryClient = new QueryClient({ defaultOptions: { queries: { retry: false } } })
+  const store = makeStore()
   const result = render(
-    <Providers queryClient={queryClient}>
+    <Providers queryClient={queryClient} store={store}>
       <RouterProvider router={router} />
     </Providers>,
   )
-  return { ...result, router, queryClient }
+  return { ...result, router, queryClient, store }
 }
```

The routes and the navigation:

apps/web/src/app/routes.tsx

```
--- a/apps/web/src/app/routes.tsx
+++ b/apps/web/src/app/routes.tsx
@@ -1,6 +1,8 @@
```

```
 import type { RouteObject } from 'react-router'
 import { LoginPage } from '../features/auth/LoginPage.tsx'
 import { RequireAuth } from '../features/auth/RequireAuth.tsx'
+import { GymPage } from '../features/gyms/GymPage.tsx'
+import { GymsPage } from '../features/gyms/GymsPage.tsx'
 import { PokedexPage } from '../features/pokedex/PokedexPage.tsx'
 import { DetailLoading } from '../features/pokemon/DetailLoading.tsx'
 import { AddMemberPage, EditMemberPage } from '../features/team/MemberPages.tsx'
@@ -39,6 +41,8 @@
           { path: 'team', element: <TeamPage /> },
           { path: 'team/add/:pokemonId', element: <AddMemberPage /> },
           { path: 'team/:memberId', element: <EditMemberPage /> },
+          { path: 'gyms', element: <GymsPage /> },
+          { path: 'gyms/:id', element: <GymPage /> },
         ],
       },
       { path: '*', element: <NotFoundPage /> },
```

apps/web/src/app/Layout.tsx

```
--- a/apps/web/src/app/Layout.tsx
+++ b/apps/web/src/app/Layout.tsx
@@ -29,6 +29,9 @@
             <NavLink to="/team" className={linkClass}>
               {t.nav.team}
             </NavLink>
+            <NavLink to="/gyms" className={linkClass}>
+              {t.nav.gyms}
+            </NavLink>
           </nav>
           {/* Explicit label (htmlFor + id): wrapped in the <label>, the <select>'s
name would
               include the selected option. */}
```

Redux Toolkit and its React bindings:

apps/web/package.json

```
--- a/apps/web/package.json
+++ b/apps/web/package.json
@@ -12,11 +12,13 @@
   "dependencies": {
     "@dex506/contracts": "workspace:*",
     "@hookform/resolvers": "7.0.7",
+    "@reduxjs/toolkit": "2.14.0",
     "@tanstack/react-query": "5.108.0",
     "axios": "1.21.1",
     "react": "19.3.1",
     "react-dom": "19.3.1",
```

```
     "react-hook-form": "7.92.0",
+    "react-redux": "9.3.1",
     "react-router": "8.5.1",
     "zod": "4.6.7",
     "zustand": "5.1.0"
```

When you install, pnpm 12 warns "Agent skills awaiting approval: @reduxjs/toolkit". Redux Toolkit ships instructions for artificial intelligence coding assistants, and pnpm does not copy them into the project without your permission (pnpm, *Build settings*). The app does not need them: you can ignore the warning, or run **`pnpm approve`** and reject them so it does not show up again.

The new texts:

apps/web/src/i18n/es.ts

```
--- a/apps/web/src/i18n/es.ts
+++ b/apps/web/src/i18n/es.ts
@@ -118,6 +118,26 @@
     saving: 'Guardando…',
     cancel: 'Cancelar',
   },
+  gyms: {
+    title: 'Gimnasios',
+    loading: 'Cargando los gimnasios…',
+    error: 'No se pudieron cargar los gimnasios.',
+    notFound: 'Ese gimnasio no existe.',
+    back: 'Volver a los gimnasios',
+    badges: (won: number, total: number) => `Medallas: ${won} de ${total}`,
+    won: (badge: string) => `Ganaste la ${badge}`,
+    minLevel: (level: number) => `Nivel promedio mínimo ${level}`,
+    pickLineup: (size: number) => `Elige tu alineación de ${size}`,
+    needMembers: (size: number) =>
+      `Necesitas al menos ${size} miembros en tu equipo. Agrégalos desde la`,
+    lineupSummary: (selected: number, size: number, average: number) =>
+      `${selected} de ${size} elegidos · nivel promedio ${average}`,
+    challenge: 'Retar al gimnasio',
+    challenging: 'Combatiendo…',
+    clear: 'Vaciar la alineación',
+    victory: '¡Victoria!',
+    defeat: 'Derrota',
+  },
   notFound: { title: 'Esta página no existe', back: 'Volver al inicio' },
   routeError: { title: 'Algo salió mal', retry: 'Recargar la página' },
   footer: {
```

apps/web/src/i18n/en.ts

```
--- a/apps/web/src/i18n/en.ts
```

```
+++ b/apps/web/src/i18n/en.ts
@@ -121,6 +121,25 @@
     saving: 'Saving…',
     cancel: 'Cancel',
   },
+  gyms: {
+    title: 'Gyms',
+    loading: 'Loading gyms…',
+    error: 'Could not load the gyms.',
+    notFound: 'That gym does not exist.',
+    back: 'Back to the gyms',
+    badges: (won: number, total: number) => `Badges: ${won} of ${total}`,
+    won: (badge: string) => `You won the ${badge}`,
+    minLevel: (level: number) => `Minimum average level ${level}`,
+    pickLineup: (size: number) => `Pick your lineup of ${size}`,
+    needMembers: (size: number) => `You need at least ${size} team members. Add them
from the`,
+    lineupSummary: (selected: number, size: number, average: number) =>
+      `${selected} of ${size} picked · average level ${average}`,
+    challenge: 'Challenge the gym',
+    challenging: 'Battling…',
+    clear: 'Clear the lineup',
+    victory: 'Victory!',
+    defeat: 'Defeat',
+  },
   notFound: { title: 'This page does not exist', back: 'Back to the start' },
   routeError: { title: 'Something went wrong', retry: 'Reload the page' },
   footer: {
```

The tests

The handlers add the gyms, the badges, and the challenge:

apps/web/src/test/handlers.ts

```
--- a/apps/web/src/test/handlers.ts
+++ b/apps/web/src/test/handlers.ts
@@ -89,3 +89,22 @@
     }),
   ]
 }
+
+export const GYM_CARIBE = {
+  id: 'caribe',
+  name: 'Gimnasio del Caribe',
+  city: 'Limón',
+  type: 'water',
+  badge: 'Medalla Arrecife',
+  lineupSize: 3,
```

```
+  minAvgLevel: 5,
+  threshold: 150,
+  weakTo: ['grass', 'electric'],
+} as const
+
+export function gymHandlers(badges: string[] = []) {
+  return [
+    http.get('/api/gyms', () => HttpResponse.json([GYM_CARIBE])),
+    http.get('/api/badges', () => HttpResponse.json(badges)),
+  ]
+}
```

The reducer and the selector are tested on their own, without React, because they are pure functions:

apps/web/src/features/gyms/lineupSlice.test.ts

```
import { describe, expect, it } from 'vitest'
import { clearLineup, keepOnly, lineupReducer, selectLineup, toggleMember } from
'./lineupSlice.ts'

describe('lineupSlice', () => {
  it('agrega y quita miembros, con un máximo de 3', () => {
    let state = lineupReducer(undefined, { type: '@@init' })
    for (const memberId of ['a', 'b', 'c', 'd'])
      state = lineupReducer(state, toggleMember({ gymId: 'g', memberId }))
    expect(state.g).toEqual(['a', 'b', 'c'])
    state = lineupReducer(state, toggleMember({ gymId: 'g', memberId: 'b' }))
    expect(state.g).toEqual(['a', 'c'])
    state = lineupReducer(state, keepOnly(['c']))
    expect(state.g).toEqual(['c'])
    expect(lineupReducer(state, clearLineup('g'))).toEqual({})
  })

  it('selectLineup devuelve el mismo arreglo mientras no cambie el estado', () => {
    const state = { lineup: { g: ['a'] } }
    expect(selectLineup(state, 'g')).toBe(selectLineup(state, 'g'))
    expect(selectLineup(state, 'otro')).toEqual([])
  })
})
```

toBe compares identity: the test checks that the selector returns the same array, not an equal one.

apps/web/src/features/gyms/Gyms.test.tsx

```
import { screen } from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import { http, HttpResponse } from 'msw/http'
import { beforeEach, describe, expect, it } from 'vitest'
import { gymHandlers, signedIn, teamHandlers } from '../../test/handlers.ts'
```

```
import { renderApp } from '../../test/renderApp.tsx'
import { server } from '../../test/server.ts'

const TEAM = [
  { id: 'm1', pokemonId: 1, nickname: 'Brote', level: 10, moves: ['hoja-afilada'] },
  { id: 'm2', pokemonId: 4, nickname: 'Chispa', level: 10, moves: ['chispazo'] },
  { id: 'm3', pokemonId: 12, nickname: 'Escara', level: 10, moves: ['picadura'] },
  { id: 'm4', pokemonId: 2, nickname: 'Fogo', level: 10, moves: ['brasa'] },
]

async function pickThree() {
  for (const name of ['Brote', 'Chispa', 'Escara']) {
    await userEvent.click(await screen.findByRole('checkbox', { name: new RegExp(name)
}))
  }
}

describe('gimnasios', () => {
  beforeEach(() => server.use(signedIn, ...gymHandlers(), ...teamHandlers(TEAM)))

  it('lista los gimnasios con las medallas ganadas', async () => {
    server.use(...gymHandlers(['caribe']))
    renderApp('/gyms')
    expect(await screen.findByText('Medallas: 1 de 1')).toBeInTheDocument()
    expect(screen.getByText('Ganaste la Medalla Arrecife')).toBeInTheDocument()
  })

  it('la alineación admite 3 miembros: el cuarto queda deshabilitado', async () => {
    const { store } = renderApp('/gyms/caribe')
    await pickThree()
    expect(screen.getByRole('checkbox', { name: /Fogo/ })).toBeDisabled()
    expect(screen.getByText('3 de 3 elegidos · nivel promedio 10')).toBeInTheDocument()
    expect(store.getState().lineup.caribe).toEqual(['m1', 'm2', 'm3'])
  })

  it('el reto muestra el resultado y la bitácora', async () => {
    server.use(
      http.post('/api/gyms/:id/challenge', () =>
        HttpResponse.json({
          won: true,
          score: 210,
          log: ['Puntaje 210 de 150 necesarios', 'Ganaste la Medalla Arrecife'],
        }),
      ),
    )
    renderApp('/gyms/caribe')
    await pickThree()
    await userEvent.click(screen.getByRole('button', { name: 'Retar al gimnasio' }))
    expect(await screen.findByRole('heading', { name: '¡Victoria!'
})).toBeInTheDocument()
```

```
    expect(screen.getByText('Puntaje 210 de 150 necesarios')).toBeInTheDocument()
  })

  it('el 422 del servidor se muestra como alerta con su mensaje', async () => {
    server.use(
      http.post('/api/gyms/:id/challenge', () =>
        HttpResponse.json(
          {
            error: {
              code: 'LEVEL_TOO_LOW',
              message: 'Gimnasio del Caribe exige un nivel promedio de 50; tu alineación
tiene 10',
            },
          },
          { status: 422 },
        ),
      ),
    )
    renderApp('/gyms/caribe')
    await pickThree()
    await userEvent.click(screen.getByRole('button', { name: 'Retar al gimnasio' }))
    expect(await screen.findByRole('alert')).toHaveTextContent('exige un nivel promedio
de 50')
  })
})
```

Test and use

```
pnpm install
pnpm test
```

Vitest should finish with **`Tests 65 passed (65)`**. In the app, with three members on your team, open the Gimnasio del Caribe (Caribbean Gym), choose the lineup, and challenge it: with three members at level 10 you will see "¡Victoria!" (victory), the battle log, and the Medalla Arrecife (Reef Badge) in the list of gyms. Also try the Gimnasio del Cerro (Hill Gym), which requires an average level of 40: the API rejects it with its message. If you installed the Redux DevTools in your browser, you see each action go by.

```
git add .
git commit -m "paso 10: los gimnasios"
```

Download the code for step 10.

Step 11: Quality

In this step you will wrap up the app's quality: end-to-end tests, an accessibility audit, the performance measurement, and a security review.

Playwright

The end-to-end tests use Playwright, as in "Testing", and the accessibility audit uses **axe**, a library that checks a page against the WCAG guidelines. They are added to the root of the monorepo, with their scripts:

package.json

```
--- a/package.json
+++ b/package.json
@@ -12,11 +12,15 @@
     "lint": "oxlint",
     "format": "prettier --write .",
     "format:check": "prettier --check .",
-    "typecheck": "pnpm -r --if-present typecheck",
+    "typecheck": "pnpm -r --if-present typecheck && tsc -p e2e && tsc -p perf",
     "test": "vitest run",
-    "build": "pnpm -r --if-present build"
+    "build": "pnpm -r --if-present build",
+    "e2e": "playwright test",
+    "perf": "playwright test -c playwright.perf.config.ts"
   },
   "devDependencies": {
+    "@axe-core/playwright": "4.14.0",
+    "@playwright/test": "1.65.1",
     "@types/node": "26.6.6",
     "oxlint": "1.94.0",
     "prettier": "3.10.1",
```

`typecheck` now also checks the types of the e2e tests and the measurement, and **`e2e`** and **`perf`** run them. After installing, Playwright needs to download its browser once. **`pnpm exec`** runs a program installed in the project, like **`npx`**:

```
pnpm install
pnpm exec playwright install chromium
```

The configuration starts the API and the web app before the tests, and shuts them down when they finish:

playwright.config.ts

```
import { defineConfig, devices } from '@playwright/test'

/**
```

```
 * The e2e tests start the API (with the creatures from fixtures/ and in-memory data)
and the web
 * app, and drive a real browser. The environment variables go here and not on the
command line: that
 * way the same command works on macOS, Linux, and Windows.
 */
export default defineConfig({
  testDir: './e2e',
  fullyParallel: false,
  workers: 1,
  // The first visit to a lazy route makes Vite transform its module: assertions get
room to wait.
  expect: { timeout: 10_000 },
  use: { baseURL: 'http://localhost:5173', trace: 'retain-on-failure' },
  projects: [{ name: 'chromium', use: { ...devices['Desktop Chrome'] } }],
  webServer: [
    {
      command: 'node apps/api/src/server.ts',
      url: 'http://localhost:3001/api/health',
      env: { PORT: '3001', DEX_DATA: 'fixtures', DEX_DB: 'memory' },
      reuseExistingServer: false,
    },
    {
      command: 'pnpm --filter @dex506/web dev --port 5173 --strictPort',
      url: 'http://localhost:5173',
      reuseExistingServer: false,
    },
  ],
})
```

- **`webServer`** starts the API with the made-up creatures and in memory (**`DEX_DB=memory`**, which is added in this step), and the web app with Vite's development server. **`url`** is the address Playwright waits for a response from before starting.
- The environment variables go in the file and not in the command: that way the same **`pnpm e2e`** works on macOS, Linux, and Windows.
- **`workers: 1`** runs the tests one at a time, because they share the same API.
- **`reuseExistingServer: false`** and **`--strictPort`** make it fail if ports 3001 or 5173 are in use: before **`pnpm e2e`**, stop **`pnpm dev`** with **`Ctrl+C`**.

e2e/tsconfig.json

```
{
  "extends": "../tsconfig.base.json",
  "compilerOptions": {
    "lib": ["ES2023", "DOM"],
    "types": ["node"],
    "module": "nodenext",
```

```
    "moduleResolution": "nodenext"
  },
  "include": [".", "../playwright.config.ts"]
}
```

The API adds the in-memory mode, with the **`DEX_DB`** variable: with **`memory`**, the trainers' data is not saved to the file and is lost when the API shuts down; without the variable, or with **`file`**, the API works as before:

apps/api/src/server.ts

```
--- a/apps/api/src/server.ts
+++ b/apps/api/src/server.ts
@@ -8,7 +8,8 @@
 const dbFile = fileURLToPath(new URL('../.data/db.json', import.meta.url))

 const pokedex = await loadPokedex(mode)
-const store = await Store.fromFile(dbFile)
+// DEX_DB=memory: the trainers' data is not saved (the e2e tests use it).
+const store = process.env.DEX_DB === 'memory' ? Store.inMemory() : await
Store.fromFile(dbFile)
 const app = createApp({ pokedex, store, secureCookies: process.env.NODE_ENV ===
'production' })

 app.listen(port, (error) => {
```

apps/api/.env.example

```
--- a/apps/api/.env.example
+++ b/apps/api/.env.example
@@ -2,3 +2,5 @@
 PORT=3001
 # pokeapi: the data `pnpm seed` downloaded · fixtures: the made-up creatures in
fixtures/
 DEX_DATA=pokeapi
+# file: teams and favorites in .data/db.json · memory: lost on restart (e2e tests)
+DEX_DB=file
```

The end-to-end tests

The complete journey: sign in, search, open a detail page, build a team, and win a gym. Also the login that returns to the protected page and the simulated API outage.

e2e/journey.spec.ts

```
import { expect, test, type Page } from '@playwright/test'

async function signIn(page: Page) {
  await page.goto('/login')
```

```
  await page.getByLabel('Correo').fill('ana@dex506.test')
  await page.getByLabel('Contraseña').fill('dex506-demo')
  await page.getByRole('button', { name: 'Entrar' }).click()
  await expect(page.getByText('Hola, Ana')).toBeVisible()
}

async function addMember(page: Page, id: number, nickname: string, level: number, move:
string) {
  await page.goto(`/team/add/${id}`)
  await page.getByLabel('Apodo (máximo 12 caracteres)').fill(nickname)
  await page.getByLabel('Nivel (1 a 100)').fill(String(level))
  await page.getByRole('button', { name: 'Siguiente' }).click()
  await page.getByLabel('Movimiento 1', { exact: true }).selectOption(move)
  await page.getByRole('button', { name: 'Guardar' }).click()
  // exact: without it, Playwright matches by substring and "Escara" already matches the
form's
  // heading ("Agregar a escarabin"), before saving finishes.
  await expect(page.getByRole('heading', { name: nickname, exact: true })).toBeVisible()
}

test('de la Pokédex a la medalla: buscar, abrir la ficha, armar el equipo y ganar un
gimnasio', async ({
  page,
}) => {
  await signIn(page)

  await page.getByRole('link', { name: 'Pokédex' }).click()
  await page.getByRole('searchbox').fill('lag')
  await expect(page.getByText('1 Pokémon')).toBeVisible()
  await expect(page).toHaveURL(/\?q=lag/)

  await page.getByRole('link', { name: 'lagunix' }).click()
  await expect(page.getByRole('heading', { name: 'lagunix', level: 1 })).toBeVisible()
  const favorite = page.getByRole('button', { name: 'Favorito', exact: true })
  await favorite.click()
  await expect(favorite).toHaveAttribute('aria-pressed', 'true')

  await addMember(page, 1, 'Brote', 10, 'hoja-afilada')
  await addMember(page, 4, 'Chispa', 10, 'chispazo')
  await addMember(page, 12, 'Escara', 10, 'picadura')

  await page.goto('/gyms/caribe')
  for (const name of ['Brote', 'Chispa', 'Escara'])
    await page.getByRole('checkbox', { name: new RegExp(name) }).check()
  await page.getByRole('button', { name: 'Retar al gimnasio' }).click()
  await expect(page.getByRole('heading', { name: '¡Victoria!' })).toBeVisible()

  await page.goto('/gyms')
  await expect(page.getByText('Ganaste la Medalla Arrecife')).toBeVisible()
})
```

```
test('sin sesión, una página protegida lleva al login y vuelve después de entrar', async
({
  page,
}) => {
  await page.goto('/team')
  await expect(page.getByRole('heading', { name: 'Inicia sesión' })).toBeVisible()
  await page.getByLabel('Correo').fill('ana@dex506.test')
  await page.getByLabel('Contraseña').fill('dex506-demo')
  await page.getByRole('button', { name: 'Entrar' }).click()
  await expect(page.getByRole('heading', { name: 'Tu equipo' })).toBeVisible()
})

test('la API simula la caída con ?fail=1 y la Pokédex muestra el error', async ({ page
}) => {
  await signIn(page)
  await page.route('**/api/pokemon?*', (route) =>
    route.continue({ url: `${route.request().url()}&fail=1` }),
  )
  await page.goto('/pokedex')
  await expect(page.getByRole('alert')).toContainText('No se pudo cargar la Pokédex.')
})
```

`page.route` intercepts the browser's requests: the last test adds **`fail=1`** to the list request, and the Pokédex shows the error.

The View Transition is checked by counting the calls to **`document.startViewTransition`**, the browser API React uses:

e2e/transition.spec.ts

```
import { expect, test } from '@playwright/test'

test('ir de la Pokédex a la Ficha usa una View Transition del navegador', async ({ page
}) => {
  // Every call to document.startViewTransition, the browser API React uses, is counted.
  await page.addInitScript(() => {
    const original = document.startViewTransition?.bind(document)
    if (!original) return
    ;(window as unknown as { transitions: number }).transitions = 0
    document.startViewTransition = ((arg: Parameters<typeof original>[0]) => {
      ;(window as unknown as { transitions: number }).transitions++
      return original(arg)
    }) as typeof document.startViewTransition
  })
  await page.goto('/login')
  await page.getByLabel('Correo').fill('ana@dex506.test')
  await page.getByLabel('Contraseña').fill('dex506-demo')
  await page.getByRole('button', { name: 'Entrar' }).click()
```

```
  await expect(page.getByText('Hola, Ana')).toBeVisible()
  await page.getByRole('link', { name: 'Pokédex' }).click()
  await page.getByRole('link', { name: 'lagunix', exact: true }).click()
  await expect(page.getByRole('heading', { name: 'lagunix', level: 1 })).toBeVisible()
  const transitions = await page.evaluate(
    () => (window as unknown as { transitions: number }).transitions,
  )
  expect(transitions).toBeGreaterThan(0)
})
```

addInitScript runs a script in the page before the app's, and here it wraps the browser function in order to count it.

The accessibility audit

e2e/accessibility.spec.ts

```
import AxeBuilder from '@axe-core/playwright'
import { expect, test } from '@playwright/test'

const PAGES = ['/', '/pokedex', '/pokedex/3', '/team', '/team/add/3', '/gyms',
'/gyms/caribe']

for (const theme of ['light', 'dark'] as const) {
  test(`sin violaciones de accesibilidad (WCAG 2.1 AA) en tema ${theme}`, async ({ page
}) => {
    await page.addInitScript((t) => localStorage.setItem('dex506.theme', t), theme)
    await page.goto('/login')
    await page.getByLabel('Correo').fill('ana@dex506.test')
    await page.getByLabel('Contraseña').fill('dex506-demo')
    await page.getByRole('button', { name: 'Entrar' }).click()
    await expect(page.getByText('Hola, Ana')).toBeVisible()

    // Violations from every page are collected and compared once: that way an error on
the first
    // page does not hide the others.
    const violationsFound: string[] = []
    for (const route of PAGES) {
      await page.goto(route)
      // A page is ready when its main heading is on screen and nothing is still
loading.
      await expect(page.getByRole('heading', { level: 1 })).toBeVisible()
      await expect(page.getByText(/^Cargando/)).toHaveCount(0)
      const { violations } = await new AxeBuilder({ page })
        .withTags(['wcag2a', 'wcag2aa', 'wcag21aa'])
        .analyze()
      violationsFound.push(
        ...violations.map(
```

```
          (v) => `${route}: ${v.id} → ${v.nodes.map((n) => n.target.join(' ')).join(',
')}`,
        ),
      )
    }
    expect(violationsFound).toEqual([])
  })
}
```

It goes through seven screens in both themes and collects the violations from all of them before comparing: that way an error in the first one does not hide the rest. Each screen is ready when its main heading is on screen and nothing "Cargando" (loading) is left. It does not wait for the network to go idle: Playwright advises against that and recommends checking what is visible (Playwright, *Page: waitForLoadState*).

The first time it ran, the audit found three problems, and this step fixes them. The first: the error coral did not reach the 4.5:1 contrast ratio in the dark theme. The dark theme switches to a lighter coral (**`accent-400`**), and the light theme to a darker one (**`accent-700`**). In the light theme, the original coral (**`accent-600`**) already passed the audit; with **`accent-700`** the error text is easier to read. The four diffs that follow apply that change to the error texts and borders:

apps/web/src/index.css

```
--- a/apps/web/src/index.css
+++ b/apps/web/src/index.css
@@ -15,6 +15,7 @@
   --color-brand-900: #134e4a;
   --color-accent-400: #fb7185;
   --color-accent-600: #e11d48;
+  --color-accent-700: #be123c;

   /* One color per type, Dex 506's own. They are light backgrounds: the text on top is
always dark. */
   --color-type-normal: #e7e5e4;
```

apps/web/src/features/auth/LoginPage.tsx

```
--- a/apps/web/src/features/auth/LoginPage.tsx
+++ b/apps/web/src/features/auth/LoginPage.tsx
@@ -82,7 +82,7 @@
           />
         </label>
         {result.error && (
-          <p role="alert" className="text-sm text-accent-600">
+          <p role="alert" className="text-sm text-accent-700 dark:text-accent-400">
             {result.error}
           </p>
         )}
```

apps/web/src/features/gyms/GymPage.tsx

```
--- a/apps/web/src/features/gyms/GymPage.tsx
+++ b/apps/web/src/features/gyms/GymPage.tsx
@@ -95,7 +95,7 @@
       </div>

       {challengeError && (
-        <p role="alert" className="text-accent-600">
+        <p role="alert" className="text-accent-700 dark:text-accent-400">
           {challengeError}
         </p>
       )}
```

apps/web/src/features/pokemon/FavoriteButton.tsx

```
--- a/apps/web/src/features/pokemon/FavoriteButton.tsx
+++ b/apps/web/src/features/pokemon/FavoriteButton.tsx
@@ -42,13 +42,13 @@
         aria-pressed={shown}
         disabled={favorites.isPending}
         onClick={toggle}
-        className="rounded-md border border-accent-600 px-3 py-1 text-sm text-accent-
600 aria-pressed:bg-accent-600 aria-pressed:text-white"
+        className="rounded-md border border-accent-700 dark:border-accent-400 px-3 py-1
text-sm text-accent-700 dark:text-accent-400 aria-pressed:bg-accent-700 aria-
pressed:text-white dark:aria-pressed:bg-accent-400 dark:aria-pressed:text-stone-950"
       >
         {/* The name does not change with the state: aria-pressed says whether it is
marked. */}
         <span aria-hidden="true">{shown ? '★' : '☆'}</span> {t.detail.favorite}
       </button>
       {failed && (
-        <p role="alert" className="text-sm text-accent-600">
+        <p role="alert" className="text-sm text-accent-700 dark:text-accent-400">
           {t.detail.favoriteFailed}
         </p>
```

```
        )}
```

The second: the silhouette of an image without **`alt`** was a **`role="img"`** with no name. Now, if the image is decorative, it is hidden from screen readers:

apps/web/src/shared/ui/PokemonSprite.tsx

```
--- a/apps/web/src/shared/ui/PokemonSprite.tsx
+++ b/apps/web/src/shared/ui/PokemonSprite.tsx
@@ -15,10 +15,12 @@
 }) {
   const [failed, setFailed] = useState(false)
   if (failed) {
+    // Without alt text the image is decorative (the name is right next to it): it is
hidden from
+    // screen readers instead of announcing an unnamed image.
+    const label = alt ? { role: 'img', 'aria-label': alt } : { 'aria-hidden': true }
     return (
       <svg
-        role="img"
-        aria-label={alt}
+        {...label}
         width={size}
         height={size}
         viewBox="0 0 96 96"
```

The third: a **`<select>`** wrapped in its **`<label>`** added the selected option to its name ("Movimiento 1 Elige un movimiento", move 1, choose a move). The language and type selects already have their label separate since their own steps; the moves selects separate it here, connected with **`htmlFor`** and an id from **`useId`**, as in "Refs and the DOM", because there is one per move. The same prefix goes on the ids of the error messages, and the dark theme's red border is kept only for fields with errors:

apps/web/src/features/team/MemberForm.tsx

```
--- a/apps/web/src/features/team/MemberForm.tsx
+++ b/apps/web/src/features/team/MemberForm.tsx
@@ -1,5 +1,5 @@
 import { zodResolver } from '@hookform/resolvers/zod'
-import { useEffect, useRef, useState } from 'react'
+import { useEffect, useId, useRef, useState } from 'react'
 import { useFieldArray, useForm } from 'react-hook-form'
 import { MAX_MOVES, type ApiError, type Pokemon, type TeamMemberInput } from
'@dex506/contracts'
 import { useI18n } from '../../i18n/i18nContext.ts'
@@ -13,12 +13,12 @@
 }

 const inputClass =
```

```
-  'w-full rounded-md border border-stone-300 px-3 py-2 aria-invalid:border-accent-600
dark:border-stone-700 dark:bg-stone-900'
+  'w-full rounded-md border border-stone-300 px-3 py-2 aria-invalid:border-accent-700
dark:border-stone-700 dark:bg-stone-900 dark:aria-invalid:border-accent-400'

 function FieldError({ id, message }: { id: string; message: string | undefined }) {
   if (!message) return null
   return (
-    <p id={id} className="text-sm text-accent-600">
+    <p id={id} className="text-sm text-accent-700 dark:text-accent-400">
       {message}
     </p>
   )
@@ -28,6 +28,9 @@
   const { t } = useI18n()
   const [step, setStep] = useState<1 | 2>(1)
   const [formError, setFormError] = useState<string | null>(null)
+  // A unique prefix per form for the ids of labels and error messages: two forms on
the same page
+  // do not clash.
+  const idPrefix = useId()

   const {
     register,
@@ -96,12 +99,12 @@
             <input
               {...register('nickname')}
               aria-invalid={errors.nickname ? true : undefined}
-              aria-describedby={errors.nickname ? 'nickname-error' : undefined}
+              aria-describedby={errors.nickname ? `${idPrefix}-nickname-error` :
undefined}
               className={inputClass}
             />
           </label>
           {/* Outside the <label>: otherwise the error would become part of the field's
name. */}
-          <FieldError id="nickname-error" message={errors.nickname?.message} />
+          <FieldError id={`${idPrefix}-nickname-error`} message=
{errors.nickname?.message} />
         </div>
         <div className="space-y-1">
           <label className="block space-y-1">
@@ -110,12 +113,12 @@
               type="number"
               {...register('level', { valueAsNumber: true })}
               aria-invalid={errors.level ? true : undefined}
-              aria-describedby={errors.level ? 'level-error' : undefined}
+              aria-describedby={errors.level ? `${idPrefix}-level-error` : undefined}
               className={inputClass}
             />
```

```
            </label>
            {/* Outside the <label>: otherwise the error would become part of the field's
name. */}
-           <FieldError id="level-error" message={errors.level?.message} />
+           <FieldError id={`${idPrefix}-level-error`} message={errors.level?.message} />
          </div>
          <button
            type="button"
@@ -129,14 +132,19 @@
        <fieldset
          hidden={step !== 2}
          className="space-y-3"
-         aria-describedby={movesError ? 'moves-error' : undefined}
+         aria-describedby={movesError ? `${idPrefix}-moves-error` : undefined}
        >
          <legend className="text-sm font-medium">{t.team.moves(MAX_MOVES)}</legend>
          {moves.fields.map((field, index) => (
            <div key={field.id} className="flex gap-2">
-             <label className="flex-1">
-               <span className="sr-only">{t.team.moveN(index + 1)}</span>
+             {/* Explicit label (htmlFor + id): wrapped in the <label>, the <select>'s
name would
+                 include the selected option ("Movimiento 1 Elige un movimiento"). */}
+             <div className="flex-1">
+               <label htmlFor={`${idPrefix}-move-${index}`} className="sr-only">
+                 {t.team.moveN(index + 1)}
+               </label>
                <select
+                 id={`${idPrefix}-move-${index}`}
                  {...register(`moves.${index}.name`)}
                  aria-invalid={errors.moves?.[index]?.name ? true : undefined}
                  className={inputClass}
@@ -148,7 +156,7 @@
                    </option>
                  ))}
                </select>
-             </label>
+             </div>
              <button
                type="button"
                onClick={() => moves.remove(index)}
@@ -169,7 +177,7 @@
              {t.team.addMove}
            </button>
          )}
-         <FieldError id="moves-error" message={movesError} />
+         <FieldError id={`${idPrefix}-moves-error`} message={movesError} />
          <div className="flex gap-2">
            <button
              type="button"
```

```
@@ -189,7 +197,7 @@
       </fieldset>

       {formError && (
-        <p role="alert" className="text-sm text-accent-600">
+        <p role="alert" className="text-sm text-accent-700 dark:text-accent-400">
           {formError}
         </p>
       )}
```

The performance measurement

React Compiler is turned on in the web app, as in "React Compiler":

apps/web/vite.config.ts

```
--- a/apps/web/vite.config.ts
+++ b/apps/web/vite.config.ts
@@ -1,9 +1,16 @@
+import babel from '@rolldown/plugin-babel'
 import tailwindcss from '@tailwindcss/vite'
-import react from '@vitejs/plugin-react'
+import react, { reactCompilerPreset } from '@vitejs/plugin-react'
 import { defineConfig } from 'vite'

 export default defineConfig({
-  plugins: [react(), tailwindcss()],
+  plugins: [
+    react(),
+    // React Compiler: memoizes components and values at build time, without memo or
useMemo by hand.
+    // The decision to keep it on comes from the measurement in part 3
(perf/README.md).
+    babel({ presets: [reactCompilerPreset()] }),
+    tailwindcss(),
+  ],
   server: {
     // In development, /api goes to the Express API. To the browser everything comes
from the same
     // origin (localhost:5173), so the session cookie travels without configuring CORS.
```

apps/web/package.json

```
--- a/apps/web/package.json
+++ b/apps/web/package.json
@@ -24,6 +24,8 @@
     "zustand": "5.1.0"
   },
   "devDependencies": {
```

```
+    "@babel/core": "8.1.2",
+    "@rolldown/plugin-babel": "0.3.1",
     "@tailwindcss/vite": "4.3.5",
     "@testing-library/jest-dom": "7.0.1",
     "@testing-library/react": "16.3.3",
@@ -31,6 +33,7 @@
     "@types/react": "19.3.0",
     "@types/react-dom": "19.3.0",
     "@vitejs/plugin-react": "6.3.0",
+    "babel-plugin-react-compiler": "1.0.0",
     "jsdom": "31.0.1",
     "msw": "3.0.7",
     "tailwindcss": "4.3.5",
```

`@rolldown/plugin-babel` runs Babel inside Vite, and **`reactCompilerPreset`** is the compiler configuration that comes with the React plugin.

To measure the most expensive case, the Pokédex adds a view with all 151 on one page:

apps/web/src/features/pokedex/PokedexPage.tsx

```
--- a/apps/web/src/features/pokedex/PokedexPage.tsx
+++ b/apps/web/src/features/pokedex/PokedexPage.tsx
@@ -11,6 +11,8 @@
 import { usePokemonList } from './usePokemonList.ts'

 const LIMIT = 24
+/** The full view: all 151 on one page. It is the one used to measure performance (part
3). */
+const ALL = 151

 /**
  * The filters live in the URL: they can be shared and survive a reload. Each change
replaces the
@@ -25,6 +27,7 @@
       q: params.get('q') ?? '',
       type: type.success ? type.data : undefined,
       page: Math.max(1, Number(params.get('page')) || 1),
+      all: params.get('all') === '1',
     }
   }, [params])

@@ -34,7 +37,8 @@
       const search: Record<string, string> = {}
       if (merged.q) search.q = merged.q
       if (merged.type) search.type = merged.type
-      if (merged.page > 1) search.page = String(merged.page)
+      if (merged.page > 1 && !merged.all) search.page = String(merged.page)
+      if (merged.all) search.all = '1'
```

```
       setParams(search, { replace: true })
     },
     [filters, setParams],
@@ -73,8 +77,8 @@
   const list = usePokemonList({
     q: filters.q || undefined,
     type: filters.type,
-    page: filters.page,
-    limit: LIMIT,
+    page: filters.all ? 1 : filters.page,
+    limit: filters.all ? ALL : LIMIT,
   })
   const pages = list.data ? Math.max(1, Math.ceil(list.data.total / LIMIT)) : 1

@@ -134,6 +138,14 @@
             </button>
           ))}
         </fieldset>
+        <label className="flex items-center gap-2 text-sm">
+          <input
+            type="checkbox"
+            checked={filters.all}
+            onChange={(event) => setFilters({ all: event.target.checked, page: 1 })}
+          />
+          {t.pokedex.showAll}
+        </label>
       </div>

       {list.isPending ? (
@@ -166,28 +178,30 @@
               </li>
             ))}
           </ul>
-          <nav
-            aria-label={t.pokedex.pageOf(filters.page, pages)}
-            className="flex items-center gap-3"
-          >
-            <button
-              type="button"
-              disabled={filters.page <= 1}
-              onClick={() => setFilters({ page: filters.page - 1 })}
-              className="rounded-md border border-stone-300 px-3 py-1 disabled:opacity-
40 dark:border-stone-700"
+          {!filters.all && (
+            <nav
+              aria-label={t.pokedex.pageOf(filters.page, pages)}
+              className="flex items-center gap-3"
             >
-              {t.pokedex.previous}
-            </button>
```

```
-            <span className="text-sm">{t.pokedex.pageOf(filters.page, pages)}</span>
-            <button
-              type="button"
-              disabled={filters.page >= pages}
-              onClick={() => setFilters({ page: filters.page + 1 })}
-              className="rounded-md border border-stone-300 px-3 py-1 disabled:opacity-
40 dark:border-stone-700"
-            >
-              {t.pokedex.next}
-            </button>
-          </nav>
+              <button
+                type="button"
+                disabled={filters.page <= 1}
+                onClick={() => setFilters({ page: filters.page - 1 })}
+                className="rounded-md border border-stone-300 px-3 py-1
disabled:opacity-40 dark:border-stone-700"
+              >
+                {t.pokedex.previous}
+              </button>
+              <span className="text-sm">{t.pokedex.pageOf(filters.page, pages)}</span>
+              <button
+                type="button"
+                disabled={filters.page >= pages}
+                onClick={() => setFilters({ page: filters.page + 1 })}
+                className="rounded-md border border-stone-300 px-3 py-1
disabled:opacity-40 dark:border-stone-700"
+              >
+                {t.pokedex.next}
+              </button>
+            </nav>
+          )}
         </>
       )}
     </section>
```

apps/web/src/features/pokedex/PokedexPage.test.tsx

```
--- a/apps/web/src/features/pokedex/PokedexPage.test.tsx
+++ b/apps/web/src/features/pokedex/PokedexPage.test.tsx
@@ -58,6 +58,14 @@
     expect(screen.getByRole('button', { name: 'Siguiente' })).toBeDisabled()
   })

+  it('la vista completa pide los 151 en una página y oculta la paginación', async () =>
{
+    const { router } = renderApp('/pokedex')
+    await screen.findByText('12 Pokémon')
+    await userEvent.click(screen.getByRole('checkbox', { name: 'Mostrar todos' }))
+    expect(router.state.location.search).toBe('?all=1')
```

```
+    expect(screen.queryByRole('button', { name: 'Siguiente' })).not.toBeInTheDocument()
+  })
+
   it('el enlace «Pokédex» limpia la búsqueda, y el campo la sigue', async () => {
     const { router } = renderApp('/pokedex?q=lag')
     expect(await screen.findByRole('searchbox')).toHaveValue('lag')
```

apps/web/src/i18n/es.ts

```
--- a/apps/web/src/i18n/es.ts
+++ b/apps/web/src/i18n/es.ts
@@ -56,6 +56,7 @@
     error: 'No se pudo cargar la Pokédex.',
     retry: 'Reintentar',
     empty: 'Ningún Pokémon coincide con la búsqueda.',
+    showAll: 'Mostrar todos',
     previous: 'Anterior',
     next: 'Siguiente',
     pageOf: (page: number, pages: number) => `Página ${page} de ${pages}`,
```

apps/web/src/i18n/en.ts

```
--- a/apps/web/src/i18n/en.ts
+++ b/apps/web/src/i18n/en.ts
@@ -59,6 +59,7 @@
     error: 'Could not load the Pokédex.',
     retry: 'Try again',
     empty: 'No Pokémon match your search.',
+    showAll: 'Show all',
     previous: 'Previous',
     next: 'Next',
     pageOf: (page: number, pages: number) => `Page ${page} of ${pages}`,
```

The measurement is another Playwright configuration: the production build served with **`vite preview`** (a local server that shows the build exactly as it would be in production) and the API with the 151 Pokémon from PokéAPI.

playwright.perf.config.ts

```
import { defineConfig, devices } from '@playwright/test'

/**
 * Part 3's performance measurement (it does not run with `pnpm test` or in CI): a
production build
 * served with `vite preview` and the API with PokéAPI's 151 Pokémon (you need to have
run
 * `pnpm --filter @dex506/api seed` once). See perf/README.md.
 */
```

```
export default defineConfig({
  testDir: './perf',
  testMatch: '*.perf.ts',
  workers: 1,
  timeout: 300_000,
  use: { baseURL: 'http://localhost:4173' },
  projects: [{ name: 'chromium', use: { ...devices['Desktop Chrome'] } }],
  webServer: [
    {
      command: 'node apps/api/src/server.ts',
      url: 'http://localhost:3001/api/health',
      env: { PORT: '3001', DEX_DATA: 'pokeapi', DEX_DB: 'memory' },
      reuseExistingServer: false,
    },
    {
      command:
        'pnpm --filter @dex506/web build && pnpm --filter @dex506/web preview --port
4173 --strictPort',
      url: 'http://localhost:4173',
      reuseExistingServer: false,
      timeout: 120_000,
    },
  ],
})
```

perf/tsconfig.json

```
{
  "extends": "../tsconfig.base.json",
  "compilerOptions": {
    "lib": ["ES2023", "DOM"],
    "types": ["node"],
    "module": "nodenext",
    "moduleResolution": "nodenext"
  },
  "include": [".", "../playwright.perf.config.ts"]
}
```

perf/typing.perf.ts

```
import { mkdir, writeFile } from 'node:fs/promises'
import { cpus, platform, release, totalmem } from 'node:os'
import { expect, test } from '@playwright/test'

const VARIANT = process.env.PERF_VARIANTE ?? 'sin-nombre'
const RUNS = 5
const TEXT = 'abcdefghij' // nothing matches: the list does not change, it only re-
renders
const CPU_SLOWDOWN = 4
```

```
const median = (xs: number[]) => {
  const s = [...xs].sort((a, b) => a - b)
  return s[Math.floor(s.length / 2)] ?? 0
}
const percentile = (xs: number[], p: number) => {
  const s = [...xs].sort((a, b) => a - b)
  return s[Math.min(s.length - 1, Math.ceil((p / 100) * s.length) - 1)] ?? 0
}

test(`escribir en el buscador con los 151 en pantalla (${VARIANT})`, async ({
  page,
  browserName,
  browser,
}) => {
  // Images are not downloaded: that way the network adds no noise to the measurement
(you see the silhouettes).
  await page.route(/\.png$/, (route) => route.abort())
  await page.goto('/login')
  await page.getByLabel('Correo').fill('ana@dex506.test')
  await page.getByLabel('Contraseña').fill('dex506-demo')
  await page.getByRole('button', { name: 'Entrar' }).click()
  await expect(page.getByText('Hola, Ana')).toBeVisible()

  const cdp = await page.context().newCDPSession(page)
  const latencies: number[] = []
  for (let run = 0; run < RUNS; run++) {
    await page.goto('/pokedex?all=1')
    await expect(page.getByText('151 Pokémon')).toBeVisible()
    await cdp.send('Emulation.setCPUThrottlingRate', { rate: CPU_SLOWDOWN })
    // For each key: from keydown to the next frame (React's work plus painting).
    await page.evaluate(() => {
      const w = window as unknown as { latencies: number[] }
      w.latencies = []
      document.addEventListener(
        'keydown',
        () => {
          const start = performance.now()
          requestAnimationFrame(() => setTimeout(() =>
w.latencies.push(performance.now() - start)))
        },
        { capture: true },
      )
    })
    await page.getByRole('searchbox').pressSequentially(TEXT, { delay: 100 })
    await page.waitForTimeout(200)
    latencies.push(
      ...(await page.evaluate(() => (window as unknown as { latencies: number[]
}).latencies)),
    )
```

```
    await cdp.send('Emulation.setCPUThrottlingRate', { rate: 1 })
  }

  const result = {
    variant: VARIANT,
    date: new Date().toISOString(),
    machine: {
      cpu: cpus()[0]?.model,
      cores: cpus().length,
      memoryGB: Math.round(totalmem() / 2 ** 30),
      os: `${platform()} ${release()}`,
    },
    browser: `${browserName} ${browser.version()}`,
    cpuSlowdown: CPU_SLOWDOWN,
    keystrokes: latencies.length,
    medianMs: Number(median(latencies).toFixed(1)),
    p75Ms: Number(percentile(latencies, 75).toFixed(1)),
  }
  await mkdir('perf/resultados', { recursive: true })
  await writeFile(`perf/resultados/${VARIANT}.json`, JSON.stringify(result, null, 2) +
'\n')
  expect(latencies.length).toBe(RUNS * TEXT.length)
})
```

- **`Emulation.setCPUThrottlingRate`** is a Chrome DevTools command, which Playwright sends through a **CDP** session (*Chrome DevTools Protocol*): it makes the CPU four times slower, like a modest phone.
- Images are blocked, so the network does not add noise.
- For each key, the script measures from the moment it is pressed until the next frame: React's work plus the painting.

The method and the results I got, which are the ones in the table in part 3:

perf/README.md

```
# Medición de rendimiento (parte 3, sección «Rendimiento»)

**Qué se mide:** cuánto tarda la página en reaccionar a cada tecla en el buscador de la
Pokédex con
los 151 Pokémon en pantalla. Es un caso caro a propósito: el texto del buscador es un
estado de
`PokedexPage`, así que cada tecla vuelve a renderizar la página y, sin memoizar, sus 151
tarjetas,
aunque la lista no cambie.

**Cómo:** build de producción servido con `vite preview`, la API con los datos de `pnpm
seed`,
Chromium de Playwright con la CPU 4× más lenta (`Emulation.setCPUThrottlingRate`), las
imágenes
```

````
bloqueadas para que la red no meta ruido, y 5 corridas de 10 teclas. Por cada tecla se
mide desde
`keydown` hasta el cuadro siguiente. Se reportan la mediana y el percentil 75 de las 50
teclas.

```sh
pnpm --filter @dex506/api seed   # una vez
PERF_VARIANTE=compilador pnpm perf
```

Las variantes se arman a mano antes de medir: **sin memo** (sin el plugin de Babel en
`apps/web/vite.config.ts`), **memo manual** (sin el plugin y con `export const
PokemonCard =
memo(PokemonCardBase)`) y **compilador** (la configuración actual).

## Resultados (2026-11-17)

Máquina: Apple M3 Pro, 12 núcleos, 36 GB, macOS (darwin 27.2.0). Navegador: Chromium
159.0.8154.6.
CPU 4× más lenta.

| Variante    | Ronda 1: mediana · p75 | Ronda 2: mediana · p75 |
| ----------- | ---------------------- | ---------------------- |
| Sin memo    | 9.9 ms · 10.8 ms       | 9.8 ms · 10.2 ms       |
| Memo manual | 5.0 ms · 5.5 ms        | 4.9 ms · 5.7 ms        |
| Compilador  | 5.6 ms · 8.9 ms        | 5.1 ms · 5.5 ms        |

Los JSON de la ronda 2 están en `resultados/`.

## Decisión

**React Compiler queda activo.** Deja la latencia por tecla en la mitad, igual que
`memo` a mano,
sin tocar el código de ningún componente. Dicho con todas las letras: incluso sin
memoizar, 9.9 ms
entra en un cuadro de 16 ms, así que en esta app la diferencia existe pero no se nota.
Lo que gana
el compilador es que no hay que acordarse de memoizar cuando la app crezca.
````

The zip also includes the three results files, in **perf/resultados/**. To repeat the measurement:

```
pnpm --filter @dex506/api seed
PERF_VARIANTE=compilador pnpm perf
```

On Windows, in PowerShell: **$env:PERF_VARIANTE="compilador"; pnpm perf**.

The security review

The last file in the step puts in writing what was reviewed, where, and what is left out:

SEGURIDAD.md

```
# Revisión de seguridad (paso 11)

Revisada el 2026-11-17. Lo que se comprobó, dónde y lo que queda fuera a propósito.

| Riesgo                                          | Qué hace Dex 506
| Dónde                                                
```

```
| Huella del servidor                              | Sin la cabecera `X-Powered-By`
| `apps/api/src/app.ts`                                                          |
| Secretos en el frontend                          | No hay variables `VITE_`: todo lo
que empieza con `VITE_` termina en el JavaScript público
| `apps/web`                                                                     |

**Dependencias.** `pnpm audit` del 2026-11-17 reporta un aviso alto en `picomatch`
(ReDoS,
GHSA-fc5q-39xc-vjpf) sin versión corregida publicada. Llega solo por herramientas de
desarrollo y
build (Vite → tinyglobby), no por el código que corre en el navegador ni en la API.

**Fuera de alcance, a propósito (para una app real):** límite de intentos de login,
cabeceras de
seguridad (`Content-Security-Policy`, `Strict-Transport-Security`) y una base de datos
de verdad en
lugar del archivo JSON. Y antes de publicar la app, quitar las dos cuentas de prueba,
cuya
contraseña está en este repositorio. Los pasos 13 y 14 de la guía explican cada una.
```

Test

```
pnpm install
pnpm lint
pnpm format:check
pnpm typecheck
pnpm test
pnpm e2e
```

`pnpm install` goes first because this step adds dependencies in two **`package.json`** files (the "agent skills" warning from step 10 may show up again). **`pnpm format:check`** checks that everything has Prettier's formatting; if something does not, **`pnpm format`** fixes it. Vitest should finish with **`Tests 66 passed (66)`**, and Playwright with **`6 passed`**. If an e2e test fails, Playwright leaves a trace in **`test-results/`**: open it with **`pnpm exec playwright show-trace`** and the path to its **`trace.zip`** file, and you will see each step of the test with the screen at that moment.

```
git add .
git commit -m "paso 11: calidad"
```

Download the code for step 11.

Step 12: Continuous Integration

In this step you will make every change you push go through all the checks, on a machine that is not yours.

What it is

Continuous integration (*CI*) means running the lint, the type check, the tests, and the build automatically on every change you push, on a clean server. If something fails there, the change is not ready, even if it works on your machine. **GitHub Actions** is GitHub's continuous integration service: it reads **workflows**, YAML files in **`.github/workflows/`** that say what to run and when. Each job runs on a **runner**, a clean machine GitHub lends for that purpose. In a public repository, standard runners are free; in a private one, the free plan includes an amount of usage per month (GitHub, *GitHub Actions billing*).

The workflow

.github/workflows/ci.yml

```yaml
name: ci

# On every push to main and every pull request: if anything fails, the change is not
ready.
on:
  push:
    branches: [main]
  pull_request:

jobs:
  calidad:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v7
      # Installs pnpm at the version in package.json's packageManager field.
      - uses: pnpm/action-setup@v6
      - uses: actions/setup-node@v7
        with:
          node-version-file: .nvmrc
          cache: pnpm
      # --frozen-lockfile: if the lockfile does not match package.json, fail instead of
changing it.
      - run: pnpm install --frozen-lockfile
      - run: pnpm lint
      - run: pnpm format:check
      - run: pnpm typecheck
      # The tests use only the creatures in fixtures/: CI never calls PokéAPI.
      - run: pnpm test
      - run: pnpm build

  e2e:
    needs: calidad
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v7
      - uses: pnpm/action-setup@v6
```

```
      - uses: actions/setup-node@v7
        with:
          node-version-file: .nvmrc
          cache: pnpm
      - run: pnpm install --frozen-lockfile
      - run: pnpm exec playwright install --with-deps chromium
      - run: pnpm e2e
      # If something fails, the Playwright trace is left to download from the workflow
page.
      - if: failure()
        uses: actions/upload-artifact@v7
        with:
          name: playwright-traces
          path: test-results
```

- **on** says when it runs: on every **push** to **main** and on every **pull request**, the request to merge the changes from one branch into another, which is reviewed before it is accepted.
- **jobs** are the jobs. **calidad** runs the lint, the formatting check, the type check, the tests, and the build; **e2e**, the end-to-end tests. **needs: calidad** makes **e2e** wait for **calidad** to pass: there is no point opening a browser if the unit tests fail.
- **runs-on: ubuntu-latest** picks the runner: the Ubuntu version GitHub marks as the current one, which is not always the newest.
- **steps** are the steps, in order. **uses** runs an **action**, a step someone else published: fetching the code (**actions/checkout**), installing pnpm (**pnpm/action-setup**, which reads the version from **packageManager**), and installing Node (**actions/setup-node**, which reads **.nvmrc** and caches pnpm's store). **run** runs a command.
- **--frozen-lockfile** makes the install fail if the lockfile does not match the **package.json** files, instead of changing it: in CI you install exactly what was tested.
- The tests use only the creatures in **fixtures/**: the CI never calls PokéAPI.
- **if: failure()** runs the last step only if something failed, and uploads the Playwright traces as an **artifact**, a file you download from the workflow's page.

Pushing it

Before pushing, run the same thing the CI is going to run, so you do not see it fail because of something you could catch on your machine:

```
pnpm lint
pnpm format:check
pnpm typecheck
pnpm test
pnpm build
```

Then, on GitHub, create a new repository named **dex-506**, public or private (on a public one, CI minutes are free), and **empty**: with no README, no license, and no **.gitignore**, because you already have them and a repository with files would reject the first push.

GitHub does not accept your password for pushing over HTTPS. Before the first push, sign in from the terminal with GitHub CLI, GitHub's official command-line tool: **gh auth login**, and choose HTTPS. The other option is a **personal access token**, a long password you generate in your GitHub settings and paste when git asks for your password.

Connect your folder and push the code:

```
git add .
git commit -m "paso 12: integración continua"
git branch -M main
git remote add origin https://github.com/<tu-usuario>/dex-506.git
git push -u origin main
```

- A **branch** is a named line of commits. **git branch -M main** names yours **main**, the name the workflow expects.
- **git remote add origin …** tells git where the GitHub repository is, under the name **origin**. Replace **<tu-usuario>** with your username.
- **git push -u origin main** pushes the branch and remembers the destination, so later pushes are only **git push**.

In the repository's "Actions" tab you will see the workflow run, with its two jobs in green. From then on, every **push** runs it again.

Download the code for step 12.

Step 13: The Production Build

In this step you will generate the production build and serve the entire app from a single Node process.

The build

vite build builds the web app for production: it bundles the code into a few files, minifies it (removes whitespace and shortens names), and puts a fingerprint of each file's contents in its name (a hash, like the one for passwords, but fast and with no salt): if the contents change, the name changes. That way the browser can cache them without fear of using an old one. The result ends up in **apps/web/dist/**. The detail page ends up in a separate file, because of the lazy loading from step 08.

The API serves the web app

In development there are two servers, the API and Vite's, and the proxy connects them. In production there is no Vite server: the API also serves the web app's files, from the same origin.

apps/api/src/app.ts

```
--- a/apps/api/src/app.ts
+++ b/apps/api/src/app.ts
@@ -1,3 +1,4 @@
+import path from 'node:path'
 import cookieParser from 'cookie-parser'
 import express, { type Express } from 'express'
 import type { Gym, Pokemon } from '@dex506/contracts'
@@ -16,6 +17,8 @@
   store: Store
   gyms?: Gym[]
   secureCookies?: boolean
+  /** The web build folder: in production, the API also serves the interface. */
+  staticDir?: string
 }

 /**
@@ -27,6 +30,7 @@
   store,
   gyms = GYMS,
   secureCookies = false,
+  staticDir,
 }: AppOptions): Express {
   const app = express()
   app.disable('x-powered-by')
@@ -43,6 +47,21 @@
   app.use('/api', favoriteRoutes(store, pokedex))
   app.use('/api', gymRoutes(store, pokedex, gyms))

+  if (staticDir) {
+    // The build's files carry a hash in their name: they can be cached for a year
without risk.
+    app.use(
+      '/assets',
+      express.static(path.join(staticDir, 'assets'), { immutable: true, maxAge: '1y'
}),
+    )
+    app.use(express.static(staticDir))
+    // A SPA: every route that is neither the API nor a file returns index.html, and
React Router
+    // decides which screen to show. That way reloading /pokedex/25 is not a server
404. A path with
+    // an extension (/assets/app.js) is a file: if it does not exist, it is a 404, not
index.html.
+    app.get(/^(?!\/api(?:\/|$))(?!.*\.[a-z0-9]+$).*/i, (_req, res) => {
+      res.sendFile(path.join(staticDir, 'index.html'))
```

```
+    })
+  }
+
   app.use(notFound)
   app.use(errorHandler)
   return app
```

- **`express.static`** serves the build's files as they are: the HTML, the JavaScript, the CSS. The ones in **`assets/`**, which carry the fingerprint in their name, go out with **`immutable`** and a one-year cache: the browser does not request them again.
- The web app is an SPA: a URL like **`/pokedex/25`** is not a file, it is a screen that React Router decides. That is why every URL that is not part of the API and is not a file responds with **`index.html`**. Without that, reloading on **`/pokedex/25`** would give a 404 from the server. The regular expression says "any URL that does not start with **`/api/`** and does not end in an extension": a file that does not exist, like an **`/assets/viejo.js`** from an earlier build, is a 404 and not an **`index.html`** disguised as JavaScript.

apps/api/src/server.ts

```
--- a/apps/api/src/server.ts
+++ b/apps/api/src/server.ts
@@ -6,11 +6,18 @@
 const port = Number(process.env.PORT ?? 3001)
 const mode: DataMode = process.env.DEX_DATA === 'fixtures' ? 'fixtures' : 'pokeapi'
 const dbFile = fileURLToPath(new URL('../.data/db.json', import.meta.url))
+const production = process.env.NODE_ENV === 'production'
+const webDist = fileURLToPath(new URL('../../web/dist', import.meta.url))

 const pokedex = await loadPokedex(mode)
 // DEX_DB=memory: the trainers' data is not saved (the e2e tests use it).
 const store = process.env.DEX_DB === 'memory' ? Store.inMemory() : await
Store.fromFile(dbFile)
-const app = createApp({ pokedex, store, secureCookies: process.env.NODE_ENV ===
'production' })
+const app = createApp({
+  pokedex,
+  store,
+  secureCookies: production,
+  staticDir: production ? webDist : undefined,
+})

 app.listen(port, (error) => {
   // In Express 5 a failure to listen (a port already in use) arrives here instead of
being thrown.
```

With **`NODE_ENV=production`**, the server also serves the web app's build.

apps/api/src/production.ts

```
// The production entry point: it sets NODE_ENV before loading the server. It lives in a
file and
// not on the command line (`NODE_ENV=production node …`) because that syntax does not
work on
// Windows.
process.env.NODE_ENV = 'production'
await import('./server.ts')
```

It is the production entry point. It sets **`NODE_ENV`** before loading the server, because the command-line form (**`NODE_ENV=production node …`**) does not work on Windows.

apps/api/package.json

```
--- a/apps/api/package.json
+++ b/apps/api/package.json
@@ -8,7 +8,8 @@
     "start": "node --env-file-if-exists=.env src/server.ts",
     "seed": "node scripts/seed.ts",
     "reset": "node scripts/reset.ts",
-    "typecheck": "tsc -p tsconfig.json"
+    "typecheck": "tsc -p tsconfig.json",
+    "start:production": "node --env-file-if-exists=.env src/production.ts"
   },
   "dependencies": {
     "@dex506/contracts": "workspace:*",
```

package.json

```
--- a/package.json
+++ b/package.json
@@ -16,7 +16,8 @@
     "test": "vitest run",
     "build": "pnpm -r --if-present build",
     "e2e": "playwright test",
-    "perf": "playwright test -c playwright.perf.config.ts"
+    "perf": "playwright test -c playwright.perf.config.ts",
+    "start": "pnpm build && pnpm --filter @dex506/api start:production"
   },
   "devDependencies": {
     "@axe-core/playwright": "4.14.0",
```

`pnpm start` builds the app and starts the API in production mode.

The test

apps/api/test/static.test.ts

```
import { mkdtemp, writeFile } from 'node:fs/promises'
import { tmpdir } from 'node:os'
import path from 'node:path'
import request from 'supertest'
import { describe, expect, it } from 'vitest'
import { createApp } from '../src/app.ts'
import { loadPokedex } from '../src/pokedex.ts'
import { Store } from '../src/store.ts'

describe('producción: la API sirve la web', () => {
  it('sirve index.html para las rutas de la SPA, los archivos tal cual y deja /api para
la API', async () => {
    const dist = await mkdtemp(path.join(tmpdir(), 'dist-'))
    await writeFile(path.join(dist, 'index.html'), '<div id="root"></div>')
    await writeFile(path.join(dist, 'app.js'), 'console.log(1)')
    const app = createApp({
      pokedex: await loadPokedex('fixtures'),
      store: Store.inMemory(),
      staticDir: dist,
    })

    expect((await request(app).get('/pokedex/25')).text).toBe('<div id="root"></div>')
    expect((await request(app).get('/app.js')).text).toBe('console.log(1)')
    expect((await request(app).get('/assets/old.js')).status).toBe(404)
    expect((await request(app).get('/api')).status).toBe(404)
    const api = await request(app).get('/api/nada')
    expect(api.status).toBe(404)
    expect(api.body.error.code).toBe('NOT_FOUND')
  })
})
```

The test builds a fake build in a temporary folder and checks the three rules: a SPA URL returns **`index.html`**, a file goes out as it is, and **`/api`** still belongs to the API, with its 404 shaped as an **`ApiError`**.

The final README

README.md

```
--- a/README.md
+++ b/README.md
@@ -17,11 +17,21 @@

 ## Comandos

-| Comando          | Qué hace                                      |
-| ---------------- | --------------------------------------------- |
-| `pnpm install`   | Instala las dependencias de todos los paquetes |
-| `pnpm lint`      | Revisa el código con oxlint                    |
```

```
-| `pnpm typecheck` | Comprueba los tipos de cada paquete            |
-| `pnpm test`      | Corre las pruebas de todos los paquetes        |
-| `pnpm build`     | Genera el build de producción                  |
-| `pnpm format`    | Formatea el código con Prettier                |
+| Comando                          | Qué hace
|
+| -------------------------------- | ------------------------------------------------
--------------------- |
+| `pnpm install`                   | Instala las dependencias de todos los paquetes
|
+| `pnpm --filter @dex506/api seed` | Descarga una vez los 151 Pokémon de PokéAPI (con
caché y pausas)        |
+| `pnpm dev`                       | La API en `localhost:3001` y la web en
`localhost:5173`                |
+| `pnpm lint`                      | Revisa el código con oxlint
|
+| `pnpm format`                    | Formatea el código con Prettier
|
+| `pnpm typecheck`                 | Comprueba los tipos de cada paquete
|
+| `pnpm test`                      | Corre las pruebas de todos los paquetes (solo con
`fixtures/`)          |
+| `pnpm e2e`                       | Pruebas en un navegador real y auditoría de
accesibilidad               |
+| `pnpm build`                     | Genera el build de producción de la web
|
+| `pnpm start`                     | Build y servidor de producción: la API sirve la
web en `localhost:3001` |
+| `pnpm perf`                      | La medición de rendimiento de la parte 3 (ver
`perf/README.md`)         |
+
+Sin red, o sin correr `seed`, la API puede usar las criaturas inventadas: copia
+`apps/api/.env.example` como `apps/api/.env` y cambia `DEX_DATA=fixtures`.
+
+Cuentas de prueba: `ana@dex506.test` y `leo@dex506.test`, contraseña `dex506-demo`.
```

Test and serve

```
pnpm test
pnpm start
```

Vitest should finish with **`Tests 67 passed (67)`**. **`pnpm start`** builds the app and, while building it, Vite warns "Some chunks are larger than 500 kB after minification". It is not an error: the main file bundles React, React Router, TanStack Query, Redux Toolkit, and React Hook Form, and Vite suggests splitting it. For this app, having the detail page in its own file is enough; in a larger one, more routes with **`lazy`** make it smaller. Then the API announces **`http://localhost:3001`**: open that address, without Vite's port, and you will see the entire app served by a single process. The session cookie is now **`Secure`** (the attribute that lets

it travel only over HTTPS), and it still works without HTTPS because browsers treat **`localhost`** as a secure origin. I checked this in Chromium. If in another browser the login does not let you in (you are sent back to the form with no error), it did not save the cookie: try Chrome.

> **The usual mistake:** the app opens, but when you reload on a detail page you get "No existe la ruta GET /pokedex/25" (the route GET /pokedex/25 does not exist). The rule that returns **`index.html`** for the SPA's URLs is missing, or it comes after **`notFound`**: the order of the middlewares matters.

What it would take to deploy it

This guide ends at the production build on your machine. To put it on the internet, you need:

- **a server that runs Node 26**, with a process that restarts it if it goes down;
- **HTTPS**, because the session cookie is **`Secure`** in production;
- **the environment variables** from **`.env.example`** (**`NODE_ENV=production`** is set by **`production.ts`**);
- **to remove the test accounts**, Ana and Leo, whose password is in the repository;
- **the PokéAPI data**: run **`pnpm --filter @dex506/api seed`** once on the server;
- **a place that does not get wiped** for **`apps/api/.data/`**, or better, a real database (see the wrap-up).

```
git add .
git commit -m "paso 13: el build de producción"
```

Download the code for step 13.

Step 14: Wrap-Up

In this step you will see what Dex 506 is missing to be an app with real users, and what you learned across the whole series.

What I left out on purpose

The security review from step 11 names three things a real app needs and this guide does not build:

- **A limit on login attempts.** Today someone can try passwords nonstop. A limit per IP address and per account stops that.
- **Security headers.** **`Content-Security-Policy`** tells the browser where it can load scripts from, which cuts off many XSS attacks even if there is a bug; **`Strict-Transport-Security`** asks it to always use HTTPS.
- **A real database.** The JSON file writes atomically and in order, but it keeps everything in memory and writes it in full on every change: with many users at the same time, you need a database such as

PostgreSQL or SQLite.

Also left out are the API's error messages, which arrive in Spanish even when the interface is in English: an API with users in several languages would return only the code, and the web app would translate the message.

What you learned

In part 1 you saw what problem React solves and why TypeScript was born, and you wrote components with props, state, and styles. In part 2 your app talked to a server, navigated between screens, validated forms, and chose where each piece of data lives. In part 3 you saw how React renders under the hood, measured before optimizing, used React 19 in depth, and tested at three levels. And in this part you put it all together in a complete app: a shared contract, an API with sessions and tests, five screens, reviewed accessibility and security, continuous integration, and a production build.

Bibliography

- Node.js, Modules: TypeScript. Accessed November 18, 2026.
- pnpm, Installation, Dependency resolution settings, Build settings, and CLI settings. Accessed November 18, 2026.
- Express, Error handling. Accessed November 18, 2026.
- PokéAPI, documentación v2, with its fair use policy. Accessed November 18, 2026.
- Playwright, Page: waitForLoadState. Accessed November 18, 2026.
- GitHub, GitHub Actions billing. Accessed November 18, 2026.
- Lauren Tan, Joe Savona, and Mofei Zhang, React Compiler v1.0, October 7, 2025. Accessed November 18, 2026.