Construye una app React + TypeScript de cero a build de producción

Creado por Manuel Herrera Mesén · 30 de noviembre de 2026

Parte 4 de 4 · react · typescript · node · express · frontend · serie-react

▼ EN ESTE ARTÍCULO

- Paso 00: qué vamos a construir
- Paso 01: preparar el entorno
- Paso 02: crear el monorepo
- Paso 03: el contrato
- Paso 04: la API
- Paso 05: la base del frontend
- Paso 06: el acceso
- Paso 07: la Pokédex
- Paso 08: la ficha
- Paso 09: el taller de equipo
- Paso 10: los gimnasios
- Paso 11: calidad
- Paso 12: integración continua
- Paso 13: el build de producción
- Paso 14: cierre
- Bibliografía

> Parte 4 de 4 de la serie «Aprende React construyendo una app real». Viene de la parte 3: React avanzado.

> **Nota:** 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.

En las tres partes anteriores aprendiste React con ejemplos chicos, cada uno con una idea. Una app real no llega así: llega como una carpeta vacía y una lista de pantallas, y las ideas tienen que convivir en el mismo código, con una API detrás, pruebas que la cuiden y un build que se pueda entregar.

En esta parte construyes **Dex 506**, la Pokédex de entrenador que viste en la portada de la parte 1, desde una carpeta vacía hasta el build de producción. Cada paso trae los archivos completos, los comandos exactos, lo que tienes que ver en la terminal y cómo comprobar que funcionó.

Cómo leer esta guía. Es una guía para seguir con la terminal abierta, paso a paso y en orden: cada paso parte del anterior. Cuando un paso usa un concepto de React, enlaza la sección de la serie donde se

explicó; lo que no es React (Node, Express, pnpm, la integración continua) se explica aquí, la primera vez que aparece. Los archivos que ya existen no se repiten enteros: se muestra el **cambio**, en formato diff, donde las líneas con **+** se agregan y las líneas con **-** se quitan. Al final de cada paso hay un zip con el código completo de ese paso, para comparar el tuyo o retomar desde ahí. Como en las partes anteriores, el código está en inglés y los textos de la pantalla en español.

Todo el código lo construí paso a paso, en el mismo orden que este artículo, con Node 26.13.1 y pnpm 12.21.0, en macOS 27.2 y en Linux dentro de Docker: el código de cada paso pasa su lint, sus tipos, sus pruebas y su build sin red, y es exactamente el que trae su zip. No lo probé en Windows; donde un comando cambia en PowerShell, te doy la variante.

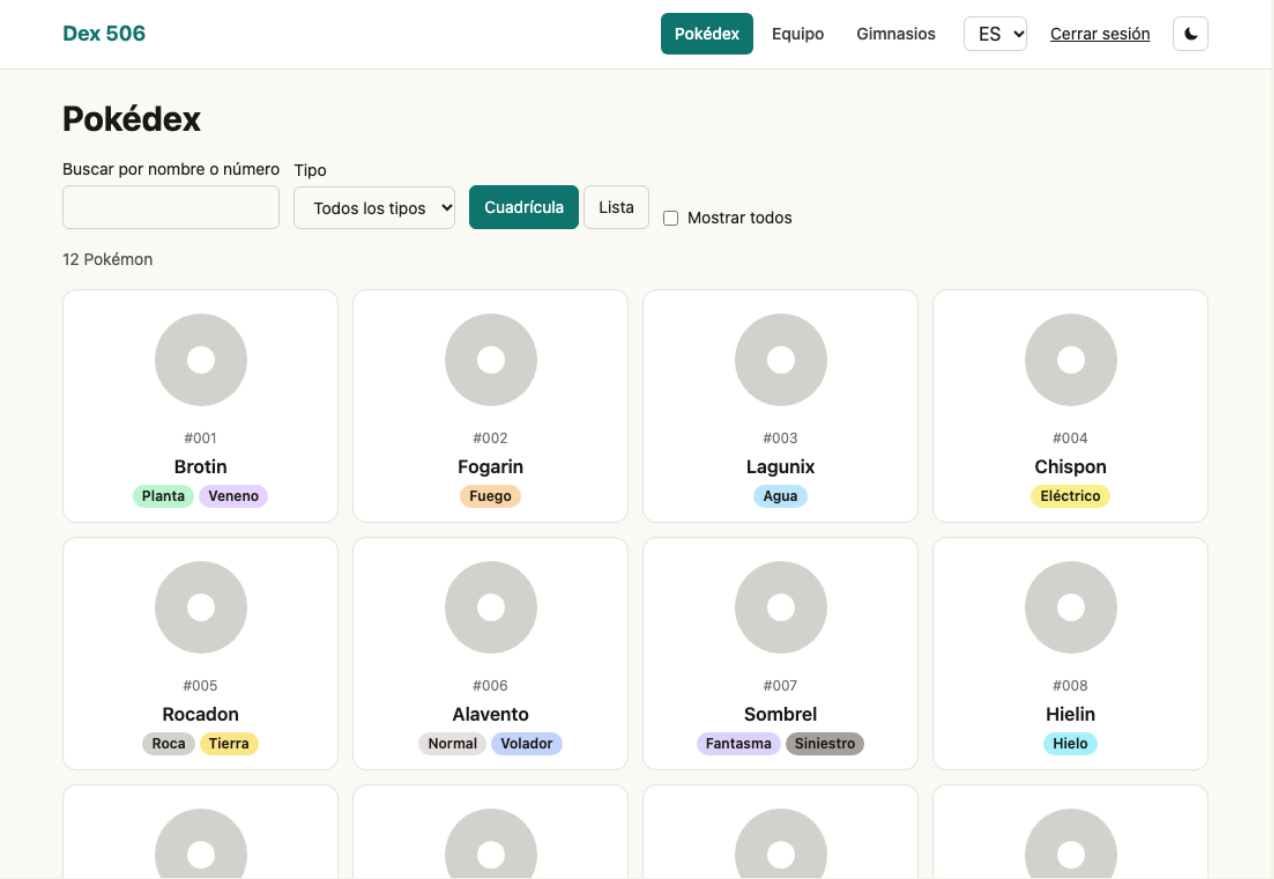
Al terminar esta parte podrás:

1. organizar un proyecto full stack en un monorepo con pnpm;
2. definir un contrato compartido entre la API y la web con Zod;
3. construir una API con Node y Express 5, con sesión, validación, errores y pruebas;
4. armar las cinco pantallas de Dex 506 con todo lo de las partes 1 a 3;
5. probar la app en tres niveles, medir su rendimiento y revisar su seguridad;
6. correr todo en integración continua y generar el build de producción.

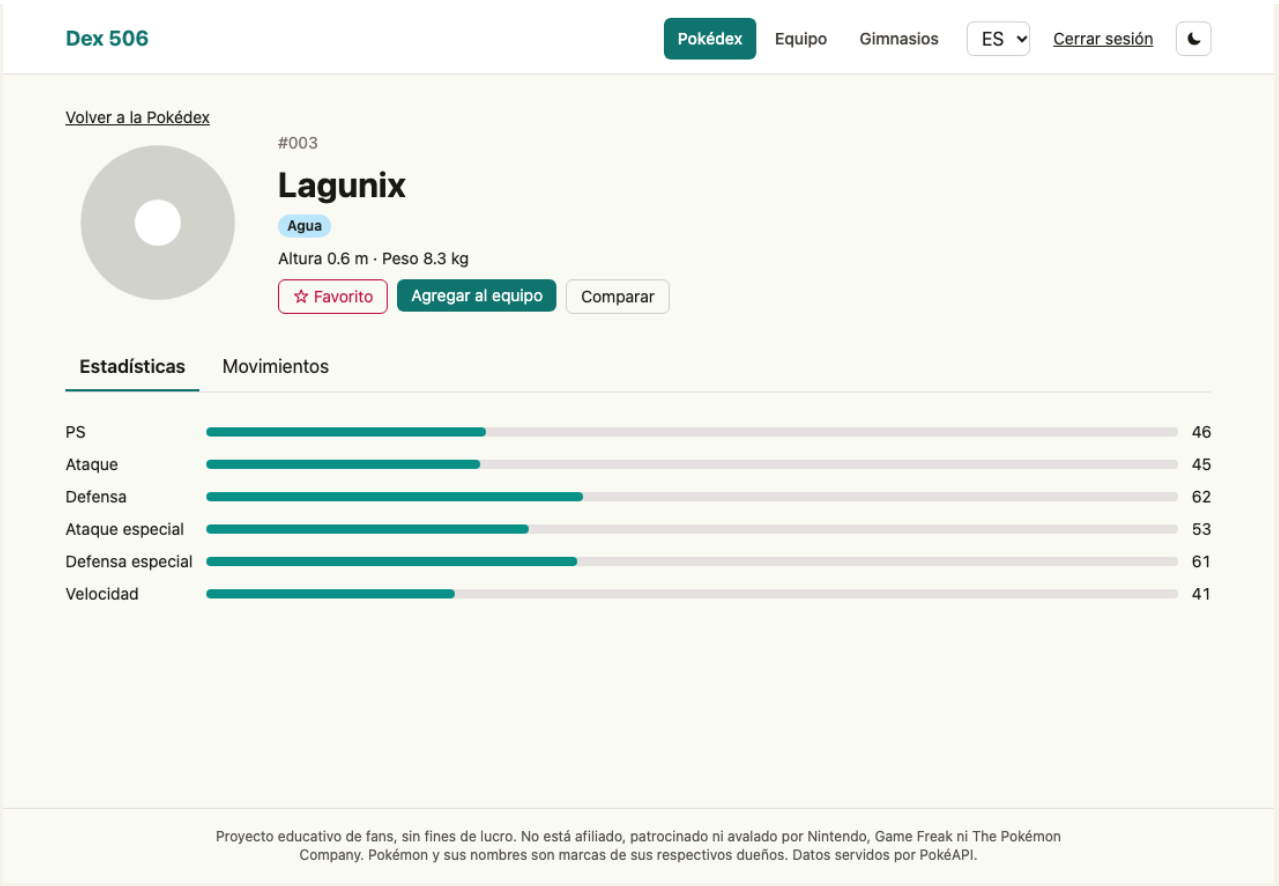
Paso 00: qué vamos a construir

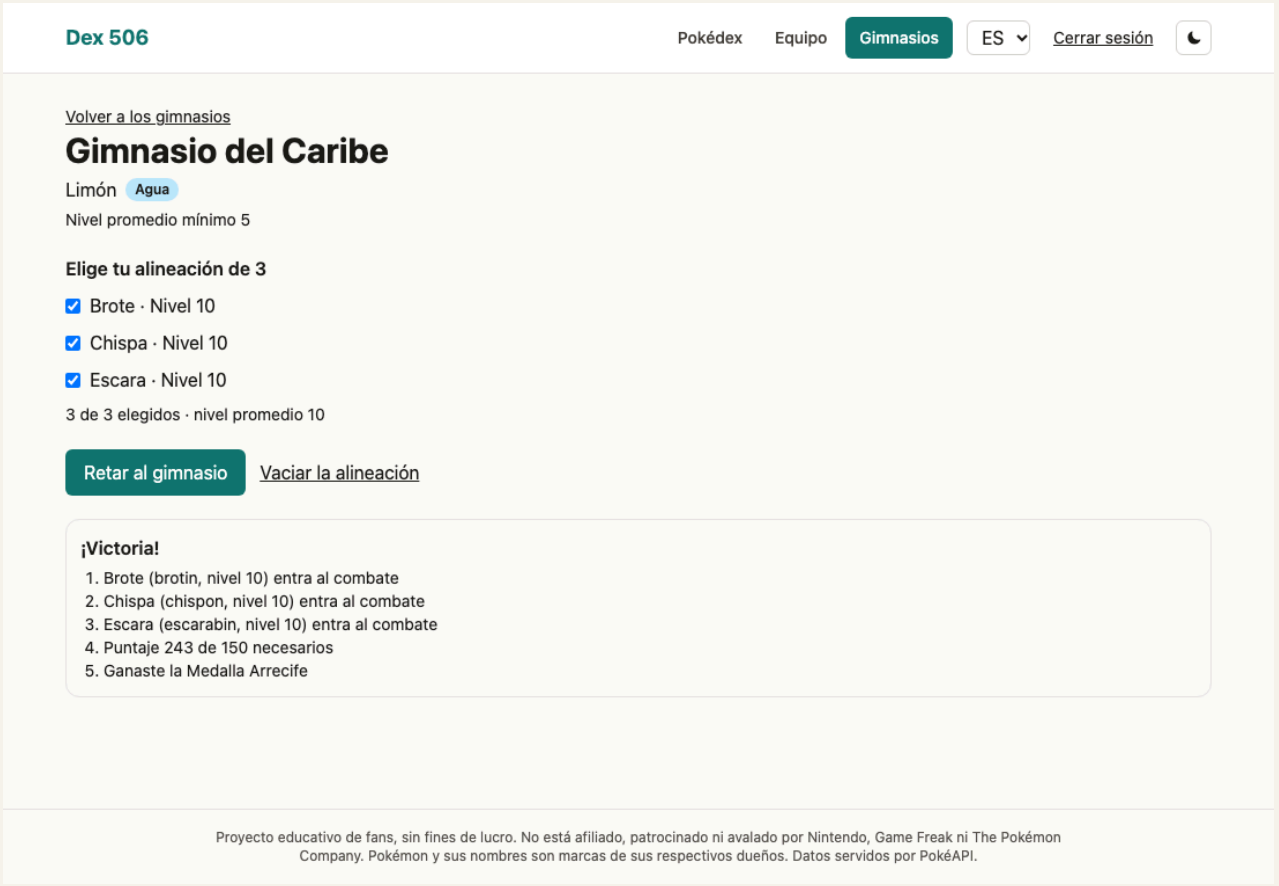
En este paso vas a ver la app terminada, qué tiene que hacer y cómo se reparte el código.

Las cinco pantallas



1. **Acceso.** Inicias sesión con un correo y una contraseña. Las pantallas de tu equipo y de los gimnasios piden sesión; si entras sin ella, la app te lleva al login y, al entrar, te devuelve adonde ibas.
2. **Pokédex.** Los primeros 151 Pokémon, con buscador, filtro por tipo, paginación y vista de grilla o de lista. El filtro vive en la dirección, así que se puede compartir.
3. **Ficha.** El detalle de un Pokémon: su imagen, tipos, tamaño, estadísticas y movimientos, en pestañas. Desde ahí lo marcas como favorito, lo agregas al equipo o lo comparas con otro.
4. **Taller de equipo.** Tu equipo de hasta seis miembros. Cada miembro tiene un apodo, un nivel y de uno a cuatro movimientos, y se agrega o se edita en un formulario de dos pasos.
5. **Gimnasios.** Ocho gimnasios inventados, uno por región de Costa Rica. Eliges una alineación de tres miembros de tu equipo, retas al gimnasio y, si ganas, te llevas su medalla.





Las capturas usan las criaturas inventadas con las que se prueba la app. Con los datos reales, cada tarjeta muestra su Pokémon.

Lo que no se ve

Además de las pantallas, la app tiene que cumplir con esto:

- **Pruebas:** cada regla de negocio, cada ruta de la API y cada pantalla tienen pruebas, y un recorrido de punta a punta en un navegador real.
- **Accesibilidad:** se usa con teclado y con lector de pantalla, y pasa una auditoría automática en sus dos temas, claro y oscuro.
- **Rendimiento:** escribir en el buscador con los 151 Pokémon en pantalla responde dentro de un cuadro. Es la medición que viste en «Rendimiento».
- **Seguridad:** la sesión vive en una cookie **`httpOnly`** y ningún secreto llega al navegador.
- **Un build de producción** que la API sirve junto con la web, desde un solo proceso.

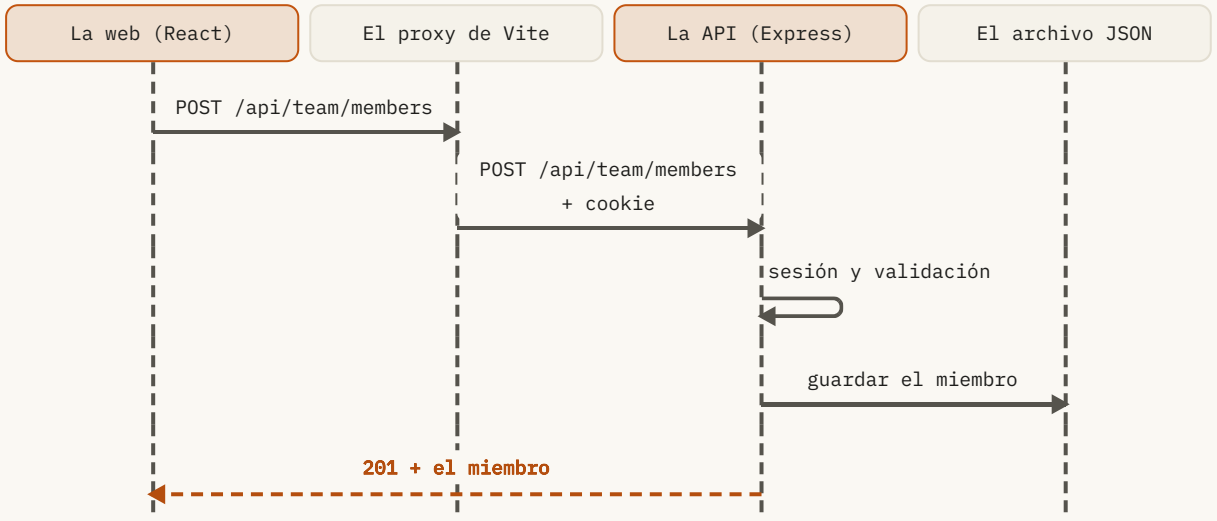
Cómo se reparte el código

Dex 506 es un **monorepo**: un solo repositorio con varios paquetes que se desarrollan juntos.

Paquete	Qué es
`packages/contracts`	Los esquemas de Zod y las reglas de negocio que comparten la API y la web
`apps/api`	La API, con Node y Express, que guarda los datos en un archivo JSON
`apps/web`	La web, con React, Vite y Tailwind

Así viaja un pedido, desde que tocas un botón hasta que la pantalla cambia:

Un pedido de Dex 506, de la pantalla a los datos



El formulario del taller manda el miembro nuevo. La web y la API se sirven desde el mismo origen, así que el pedido va a la misma dirección de la página.

En desarrollo, Vite reenvía lo que empieza con /api a la API. El navegador agrega solo la cookie de la sesión.

La API comprueba la cookie y valida el cuerpo con el mismo esquema de Zod que usó el formulario.

El cambio se escribe en el archivo JSON, de forma atómica.

La API responde con el miembro creado; si algo falla, con un error de forma conocida, que la web muestra en el campo que corresponde.

Los datos de los 151 Pokémon salen de **PokéAPI**, una API pública y gratuita. Este repositorio no guarda ningún dato ni imagen de Pokémon: un script los descarga a tu máquina, a carpetas que git ignora, y las pruebas usan doce criaturas inventadas.

Paso 01: preparar el entorno

En este paso vas a instalar las herramientas y comprobar sus versiones.

Node.js

Node.js es el programa que ejecuta JavaScript fuera del navegador: la API de Dex 506 corre en él, y también las herramientas de la web (Vite, Vitest). Esta guía usa Node 26, la versión LTS (*Long Term Support*, la que recibe correcciones durante más tiempo). Instálala desde nodejs.org o con un gestor de versiones como nvm.

Node ejecuta archivos **`.ts`** directamente, quitando los tipos al cargarlos, sin un paso de compilación: lo hace por defecto desde las versiones 22.18 y 23.6, y es una función estable desde la 25.2 (Node.js, *Modules: TypeScript*). Esta app lo aprovecha: la API no tiene build, y Node la ejecuta tal como la escribes.

pnpm

pnpm es un gestor de paquetes, como npm: descarga las bibliotecas que usa el proyecto. Esta guía lo usa por dos razones: maneja bien los monorepos, con sus **workspaces** (varios paquetes en un repositorio, que se instalan juntos y se usan entre sí), y guarda cada versión de cada biblioteca una sola vez en el disco, aunque la usen varios proyectos.

La documentación de pnpm recomienda instalarlo con su script en macOS y Linux, y con npm en Windows (pnpm, *Installation*):

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

En Windows, en PowerShell:

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

Cierra la terminal y abre otra, para que tome el comando nuevo.

git y el editor

git guarda la historia de tus cambios: si un paso sale mal, vuelves al anterior. Instálalo desde git-scm.com y, si es la primera vez que lo usas, dile tu nombre y tu correo, que van en cada commit (si no, el primer **`git commit`** falla con «Author identity unknown»):

```
git config --global user.name "Tu nombre"
git config --global user.email "tu@correo.com"
```

Como editor sirve cualquiera; con Visual Studio Code, las extensiones de Oxc (el proyecto de oxlint, la herramienta que revisa el código de la app, del paso 02) y de Prettier (el formateador) marcan los errores y dan formato al guardar.

Comprobar las versiones

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

Tienes que ver **`v26`** en la primera línea (la guía usa **`v26.13.1`**) y **`12`** en la segunda (la guía usa **`12.21.0`**). Cualquier versión de git reciente sirve. El proyecto fija la versión exacta de pnpm en su **`package.json`**: si la que tienes es otra, pnpm descarga y usa la del proyecto (pnpm, *CLI settings*).

Cuentas

Para seguir la guía no necesitas ninguna cuenta. Solo el paso 12, la integración continua, usa una cuenta gratuita de GitHub.

> *"El error frecuente: **`pnpm: command not found`**. Pasa si sigues en la terminal donde lo instalaste: ábrela de nuevo."*

Paso 02: crear el monorepo

En este paso vas a crear la carpeta del proyecto con la configuración que comparten todos sus paquetes.

Crea una carpeta vacía, entra en ella y empieza un repositorio de git:

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

Todos los archivos de este paso van en la raíz de **dex-506**.

El paquete raíz

package.json describe el proyecto: su nombre, las versiones de Node y pnpm que pide, los **scripts** (comandos con nombre que corres con **pnpm <nombre>**) y las herramientas que comparten todos los paquetes.

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** evita publicarlo en el registro de npm por error.
- **"type": "module"** dice que los archivos **.js** y **.ts** usan módulos de ES (**import** y **export**).
- **packageManager** fija pnpm 12.21.0 y **engines** pide Node 26.13 o más nuevo.
- **pnpm -r** (*recursive*) corre el script en cada paquete que lo tenga, y **--if-present** hace que los paquetes que no lo tienen no fallen. **--filter** elige en qué paquetes correr un comando, y **--parallel** los corre a la vez: **pnpm --parallel --filter "./apps/*" dev** levanta juntas la API y la web, que llegan en los pasos 04 y 05.
- **devDependencies** son las herramientas que solo se usan para desarrollar y probar (el linter, Vitest); **dependencies**, las bibliotecas que el código necesita para funcionar.

- Las versiones van exactas, sin **^**: así quien sigue esta guía instala lo mismo que la guía probó.

pnpm-workspace.yaml dice dónde están los paquetes del workspace: dentro de **apps/** y de **packages/**.

pnpm-workspace.yaml

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

TypeScript

tsconfig.base.json es la configuración de TypeScript que heredan todos los paquetes:

tsconfig.base.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** activa todas las comprobaciones estrictas, y **noUncheckedIndexedAccess** hace que leer **lista[i]** dé **T | undefined**: un índice puede no existir.
- **noEmit**: TypeScript solo revisa tipos, no genera JavaScript. Node ejecuta los **.ts** de la API quitando los tipos, y Vite arma la web.
- **erasableSyntaxOnly** prohíbe la sintaxis de TypeScript que no se puede borrar sin más, como los **enum**. Es lo que necesita Node para quitar los tipos al cargar un archivo.
- **allowImportingTsExtensions** permite escribir **import './rules.ts'** con la extensión, que es como Node encuentra el archivo.
- **verbatimModuleSyntax** obliga a marcar con **import type** lo que solo es un tipo, para que se borre entero.

Las pruebas, el lint y el formato

`vitest.config.ts` hace que un solo **`pnpm test`** corra las pruebas de todos los paquetes: cada paquete tiene su propia configuración, y este archivo solo los lista.

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/*'],
  },
})
```

Un **linter** lee el código y marca errores probables sin ejecutarlo. Dex 506 usa **oxlint**, el mismo de la plantilla de Vite de la parte 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` avisa cuando queda un **`console.log`** olvidado, y deja pasar **`warn`**, **`error`** e **`info`**, que la API usa a propósito.

> – **Para profundizar**
>
> el estándar de los proyectos de TypeScript es ESLint con typescript-eslint. Dex 506 usa oxlint porque typescript-eslint todavía no soporta TypeScript 7, la versión de esta serie. Para un proyecto en TypeScript 6, ESLint sigue siendo una buena opción.

Prettier da formato al código: comillas, punto y coma, largo de línea. Sus reglas y los archivos que no toca:

.prettierrc.json

```
{
```

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

.prettierignore

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

Lo que git no guarda

.gitignore lista lo que git no tiene que guardar: lo que se genera (**node_modules**, **dist**), los secretos (**.env**) y los datos de PokéAPI, que se descargan en el paso 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 le dice a nvm, el gestor de versiones de Node, qué versión usar en esta carpeta:

.nvmrc

```
26
```

El README y la licencia

El **`README.md`** explica qué es el proyecto, el aviso de propiedad intelectual y los comandos:

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                   |
```

Y **`LICENSE`**, la licencia MIT, que permite usar y modificar el código:

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).
```

Instalar y comprobar

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

`pnpm install` descarga las herramientas de la raíz y crea **`pnpm-lock.yaml`**, el **lockfile**: la lista exacta de cada paquete instalado, con su versión. Guárdalo en git: así cada instalación posterior es idéntica. **`pnpm lint`** termina sin avisos, y **`pnpm typecheck`** no hace nada todavía, porque no hay paquetes.

Guarda el paso en git:

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

Descarga el código del paso 02.

Paso 03: el contrato

En este paso vas a escribir el paquete que comparten la API y la web: la forma de cada dato y las reglas del juego.

Por qué un paquete compartido

La API y la web hablan de lo mismo: un Pokémon, un miembro del equipo, un gimnasio. Si cada una define su propia versión, tarde o temprano dejan de coincidir, y el error aparece en producción. En «Arquitectura, calidad y seguridad» viste la solución: un **contrato**, una definición compartida de qué se envía y qué se recibe. En Dex 506 es un paquete del workspace, **`@dex506/contracts`**.

El paquete

Todos los archivos de este paso van en **`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` dice qué importa quien escribe **`import … from '@dex506/contracts'`**: el archivo **`src/index.ts`**, en TypeScript, sin compilar. Funciona porque los dos que lo usan entienden TypeScript: Node quita los tipos al cargarlo, y Vite lo transforma al armar la web.

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` es la configuración de un proyecto dentro de las pruebas del workspace, y **`name`** identifica sus pruebas en la salida. **`environment: 'node'`**: estas pruebas no necesitan un DOM.

Los esquemas

Cada dato se define una vez, como un esquema de Zod, y su tipo de TypeScript sale del esquema con **`z.infer`**, como en «Formularios» y en «TypeScript avanzado».

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`** va con **`as const`**, así **`z.enum`** y el tipo **`PokemonType`** conocen los 18 tipos exactos.
- **`PokemonSummarySchema`** es un **`pick`** del Pokémon completo: lo que muestra la lista, sin estadísticas ni movimientos.
- **`TeamMemberSchema`** extiende el miembro que se envía con el **`id`** que le pone la API.
- **`ApiErrorSchema`** es la forma de todos los errores de la API: un código, un mensaje y, si es un error de validación, un mensaje por campo.
- **`pageSchema`** es una función genérica: recibe el esquema de un elemento y devuelve el de una página de elementos. **`PokemonQuerySchema`** usa **`z.coerce`** porque los parámetros de una dirección siempre llegan como texto.

Las reglas

Las reglas de negocio son funciones puras: reciben datos y devuelven un resultado, sin tocar nada más. Por eso se pueden probar solas y usar en los dos lados.

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` es el corazón de los gimnasios: suma el nivel de cada miembro, un décimo de sus estadísticas y 40 puntos si su tipo es fuerte contra el gimnasio.

packages/contracts/src/index.ts

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

`index.ts` reúne lo que el paquete exporta.

Las pruebas

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'])
  })
})
```

Son pruebas unitarias, como las de «Pruebas»: **`describe`** agrupa las de una función, y la última comprueba que el esquema rechace un apodo y una lista de movimientos demasiado largos, y en qué campos.

Instalar y comprobar

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

`pnpm install` instala Zod para el paquete nuevo. **`pnpm typecheck`** revisa sus tipos, y **`pnpm test`** corre sus cuatro pruebas: Vitest tiene que terminar con **`Tests 4 passed (4)`**.

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

Descarga el código del paso 03.

Paso 04: la API

En este paso vas a construir la API de Dex 506 con Node y Express, con su sesión, su validación, sus errores y sus pruebas.

Lo justo de Express

En la parte 2 viste lo justo de HTTP desde el lado de quien pide. Ahora estás del otro lado: un **servidor HTTP** es un programa que escucha pedidos en un puerto y responde a cada uno. **Express** es el framework de Node más usado para escribirlos. Sus tres ideas:

- una **ruta** une un método y una dirección con la función que responde: **`router.get('/pokemon', (req, res) => …)`**. **`router`** es un grupo de rutas, que se crea con **`Router()`** de Express; **`req`** es el pedido, y **`res`**, la respuesta que armas;
- un **middleware** es una función que pasa por cada pedido antes de la ruta: puede leerlo, cambiarlo, responder o llamar a **`next()`** para seguir;
- un **manejador de errores** es un middleware con cuatro parámetros, **`(err, req, res, next)`**, al que llega cualquier error de una ruta.

Express 5, la versión de esta guía, también lleva a ese manejador los errores de una ruta **`async`**, sin un **`try`**/**`catch`** en cada una.

El paquete

Todos los archivos de la API van en **`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:*"`** usa el paquete del paso 03, desde el mismo repositorio.
- **`dev`** corre el servidor con **`node --watch`**, que lo reinicia cuando cambias un archivo, y **`--env-file-if-exists=.env`** carga las **variables de entorno** del archivo **`.env`** si existe. Una variable de entorno es un valor que el programa lee del sistema al arrancar, como el puerto: así el mismo código sirve en tu máquina y en un servidor, con valores distintos.
- **`cookie-parser`** lee las cookies de cada pedido, y **`supertest`** hace pedidos a la app en las pruebas sin levantar un servidor.

apps/api/tsconfig.json

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

`module` y **`moduleResolution`** en **`nodenext`** hacen que TypeScript resuelva los imports como Node, y **`types: ["node"]`** suma los tipos de Node (**`process`**, **`Buffer`**).

apps/api/vitest.config.ts

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

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

`.env.example` documenta las variables de entorno de la API. El **`.env`** de verdad no se guarda en git; se crea copiando este:

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
```

Los errores de la API

Todas las respuestas de error tienen la forma del **`ApiError`** del contrato. **`HttpError`** es un error que ya sabe con qué código HTTP y qué **`code`** responder:

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` lee un parámetro de la dirección, como el **`:id`** de **`/team/members/:id`**, y falla con un 400 si no es un texto.

apps/api/src/middleware/errors.ts

```
import type { ErrorRequestHandler, RequestHandler } from 'express'
```

```ts
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** responde 404 a cualquier dirección que ninguna ruta atendió.
- **errorHandler** va al final. Si la respuesta ya empezó a salir (un archivo enviado a medias), se la deja a Express, que la cierra. Si no: un **HttpError** sale con su código; un **ZodError**, el error de una validación fallida, sale como 400 con un mensaje por campo; un error que ya trae un código 4xx, como el de **express.json** con un JSON mal formado (400) o demasiado grande (413), sale con su código; cualquier otro se registra en la consola y sale como 500, sin mostrar detalles internos.

Practicar los estados de red

Para practicar los estados de carga y de error de la parte 2, este middleware permite pedirle a la API una demora o una falla, agregando un parámetro a la dirección de un pedido a la API (no a la de una página de la web: más adelante, el paso 07 muestra cómo provocar la falla desde el navegador y el paso 11 lo hace en una prueba):

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 tarda un segundo y medio, y **/api/pokemon?fail=1** responde 503, como si el servidor estuviera caído. En producción no hacen nada: nadie puede frenar ni tirar la API desde la dirección. **NODE_ENV** es la variable de entorno con la que, por convención, Node y sus bibliotecas saben si corren en producción (**production**) o no.

Los datos

La API guarda los entrenadores, las sesiones, los equipos, los favoritos y las medallas en un archivo JSON, **apps/api/.data/db.json**, que se crea al arrancar la primera vez. Para una app de aprendizaje alcanza, y se lee a simple vista; una app con muchos usuarios usaría una base de datos.

apps/api/src/store.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)
  }
}
```

- Las contraseñas no se guardan: se guarda un **hash** con **`scrypt`**, una función que convierte la contraseña en un texto del que no se puede volver atrás, con una **sal** (un valor al azar distinto por contraseña). **`timingSafeEqual`** compara sin revelar por el tiempo cuántos caracteres coinciden.
- **`verifyPassword`** usa la versión asíncrona de **`scrypt`**: calcular el hash tarda, y mientras tanto la API sigue atendiendo otros pedidos. **`hashPassword`** es síncrona porque solo corre al arrancar.
- Cada sesión guarda su usuario y cuándo vence: así la sesión vence también en el servidor, no solo la cookie en el navegador.
- **`demoUsers`** crea las dos cuentas de prueba, Ana y Leo, las dos con la contraseña **`dex506-demo`**.
- **`save`** escribe de forma **atómica**: primero a un archivo temporal y después lo renombra. Renombrar reemplaza el archivo en un solo paso, así que si el proceso se corta a mitad de una escritura, el JSON anterior queda intacto.
- **`update`** encadena las escrituras, una después de otra: dos pedidos al mismo tiempo nunca escriben el archivo a la vez. Cada archivo temporal tiene además un nombre al azar.
- **`fromFile`** solo empieza con datos vacíos si el archivo no existe. Si está corrupto o no se puede leer, la API no arranca: mejor un error claro que reemplazar los datos de los entrenadores.
- **`Store.inMemory()`** no escribe nada: es lo que usan las pruebas. Para empezar de cero con los datos de los entrenadores, **`pnpm --filter @dex506/api reset`** borra el archivo.

Los Pokémon se cargan una vez, al arrancar, y se validan contra el contrato:

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))
}
```

Con **`DEX_DATA=pokeapi`**, la API usa los que descarga el script de la próxima sección; con **`DEX_DATA=fixtures`**, las criaturas inventadas. Si un archivo está corrupto, la API falla al arrancar, no en medio de un pedido.

Descargar los Pokémon

El script **`seed`** descarga los Pokémon del 1 al 151 de PokéAPI:

apps/api/scripts/seed.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}`)
```

- La política de uso justo de PokéAPI pide guardar en caché local cada recurso que se consulta (PokéAPI, *documentación v2*). Por eso el script guarda cada respuesta en **`apps/api/.cache/pokeapi/`** y, la próxima vez, la lee de ahí.
- Espera 250 milisegundos entre una descarga y la siguiente: PokéAPI es gratuita y la sostienen sus mantenedores.
- **`normalize`** reduce la respuesta de PokéAPI a la forma del contrato, y **`PokemonSchema.parse`** la valida.

`reset` borra los datos de los entrenadores, sin tocar los 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})`)
```

Las criaturas de las pruebas

Las pruebas no usan los datos de PokéAPI, que no se guardan en el repositorio: usan doce criaturas inventadas, en **`fixtures/`**, en la raíz del 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"]
```

```
  }
]
```

Los gimnasios

Los ocho gimnasios son datos fijos de la 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` es el nivel promedio que exige cada gimnasio, **`threshold`** el puntaje que hay que alcanzar, y **`weakTo`**, los tipos que suman la ventaja de **`lineupScore`**.

La sesión

Al iniciar sesión, la API crea un **token** al azar, lo guarda en el **`Store`** junto al usuario y lo manda en una cookie **`httpOnly`**, como viste en «Arquitectura, calidad y seguridad».

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 deja pasar el pedido solo si su cookie corresponde a una sesión guardada, y deja el usuario en **res.locals**, el lugar de Express para pasar datos de un middleware a la ruta; **currentUser** lo lee.

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`** valida el cuerpo con el esquema del contrato, compara la contraseña y crea la sesión. La sesión y la cookie duran ocho horas (**`SESSION_MS`**, en milisegundos). Con un correo que no existe, compara igual la contraseña contra un hash falso (**`FAKE_HASH`**): si respondiera más rápido, el tiempo de respuesta revelaría qué cuentas existen. Y cada login borra las sesiones vencidas, para que el archivo no crezca para siempre.
- **`POST /api/auth/logout`** borra la sesión y la cookie.
- **`GET /api/auth/me`** devuelve el usuario de la sesión, o 401: es lo que la web consulta al abrirse para saber si hay alguien adentro.

Las rutas

Los Pokémon y los tipos, sin sesión:

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'
```

```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` filtra por texto (en el nombre o el número) y por tipo, y devuelve una página. **`findPokemon`** responde 404 si el número no existe; la usan también las otras rutas.

El equipo, con sesión:

apps/api/src/routes/team.ts

```ts
import { randomUUID } from 'node:crypto'
```

```ts
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
}
```

- Cada ruta lleva **`auth`** por separado, por lo que dice su comentario: con **`router.use`**, una dirección que no existe respondería 401 en vez de 404.
- **`checkMoves`** comprueba que los movimientos sean del Pokémon elegido, con un error en el campo **`moves`**.
- Un séptimo miembro es un 422: los datos son válidos, pero no se pueden aplicar. **`ownMember`** da 404 si el miembro no existe y 403 si es de otro entrenador.
- **`withoutOwner`** quita el **`ownerId`** de la respuesta: es un dato interno.

Los favoritos:

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 y **DELETE** son **idempotentes**: repetirlos deja el mismo resultado. Marcar dos veces el mismo favorito no lo duplica.

Los gimnasios y el reto:

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
}
```

El reto valida la alineación (tres miembros distintos de tu equipo), exige el nivel promedio del gimnasio, calcula el puntaje con **`lineupScore`** del contrato, arma una **bitácora** del combate (la lista de lo que pasó) y, si ganas, guarda la medalla.

La app y el servidor

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` arma la app con lo que recibe y la devuelve **sin escuchar** ningún puerto. Así las pruebas la usan con datos propios, y el servidor de verdad la levanta:

- **`app.disable('x-powered-by')`** deja de anunciar que el servidor es Express.
- **`express.json({ limit: '10kb' })`** lee el cuerpo JSON de cada pedido y rechaza los más grandes.
- El orden importa: los middlewares van antes de las rutas, y **`notFound`** y **`errorHandler`**, al final.

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 lee la configuración del entorno, carga los datos y escucha en el puerto 3001. Con **NODE_ENV=production**, la cookie de la sesión pide HTTPS.

Las pruebas

helpers.ts arma una app con las criaturas inventadas y datos en memoria, nueva para cada prueba, y un **agente** de supertest que guarda la cookie entre pedidos, como un navegador:

apps/api/test/helpers.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

```
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

```
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')
  })
})
```

Son pruebas de integración: cada una recorre la ruta entera, del pedido a la respuesta, con la validación, la sesión, los datos y el manejador de errores. Dos más prueban piezas sueltas: el **`Store`**, con veinte cambios a la vez y un archivo corrupto, y los errores de un cuerpo mal formado o demasiado grande.

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')
  })
})
```

Instalar, descargar y probar

```
pnpm install
pnpm test
```

Vitest tiene que terminar con **`Tests 28 passed (28)`**: las cuatro del contrato y las de la API.

Para levantar la API con los Pokémon de verdad, descárgalos una vez (tarda alrededor de un minuto) y crea el **`.env`** desde el ejemplo:

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

En Windows, en PowerShell, el segundo comando es **`Copy-Item apps/api/.env.example apps/api/.env`**. **`--filter`** corre el script solo en el paquete **`@dex506/api`**. La API anuncia **`API de Dex 506 en http://localhost:3001 (151 Pokémon, datos: pokeapi)`** y queda ocupando la terminal. Abre otra terminal, en la misma carpeta, y pide dos direcciones con **`curl`**, un programa de la terminal que hace pedidos HTTP (en Windows, escribe **`curl.exe`**: en PowerShell, **`curl`** es otro comando):

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

La primera responde **`{"ok":true}`**, y la segunda, una página con un solo Pokémon. Con las criaturas inventadas (**`DEX_DATA=fixtures`**, más abajo), busca **`q=lag`** en lugar de **`q=pika`**. Para detener la API, vuelve a su terminal y presiona **`Ctrl+C`**: lo mismo vale para cualquier servidor de esta guía. Para trabajar sin red, cambia **`DEX_DATA=pokeapi`** por **`DEX_DATA=fixtures`** en **`apps/api/.env`**: la API usa las criaturas inventadas.

> *"El error frecuente: la API no arranca y dice «No hay datos de PokéAPI». Falta correr el **`seed`**, o el **`.env`** dice **`DEX_DATA=pokeapi`** sin los datos descargados. Corre el **`seed`** o usa **`DEX_DATA=fixtures`**."*

> *"El error frecuente: la API dice «No se pudo usar el puerto 3001» y termina. Otro programa usa ese puerto, casi siempre una API que quedó levantada en otra terminal: detenla con **`Ctrl+C`**. El **`listen`** de **`server.ts`** revisa ese error para decirlo en vez de quedarse callado."*

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

Descarga el código del paso 04. El zip no trae los datos de PokéAPI: después de descomprimirlo, corre el **`seed`** o usa **`DEX_DATA=fixtures`**.

Paso 05: la base del frontend

En este paso vas a armar la web: Vite, Tailwind, el router, el cliente de datos, los idiomas, el tema y el layout que comparten todas las pantallas.

Todos los archivos de la web van en **`apps/web/`**. Esta vez no se usa la plantilla de Vite de la parte 1: la web es un paquete más del monorepo, y cada archivo se escribe a mano para entender qué hace.

El paquete y su configuración

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"
  }
}
```

Son las bibliotecas de las partes 1 a 3: React, React Router, TanStack Query, axios y Tailwind, y para las pruebas, Vitest, Testing Library, jsdom y MSW. **`typecheck`** es **`tsc -b`**, que revisa los dos proyectos de TypeScript de la web (más abajo).

`index.html` es la página que carga el navegador, como en «Tu primer proyecto con 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>
```

Y su ícono, un SVG propio:

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` suma el plugin de Tailwind de la parte 1 y un **proxy**: en desarrollo, Vite reenvía a la API todo lo que empieza con **`/api`**. Para el navegador, la web y la API vienen del mismo origen, y la cookie de la sesión viaja sin configurar CORS, como viste en «Arquitectura, calidad y seguridad».

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'],
  },
})
```

La web tiene dos proyectos de TypeScript, como la plantilla de Vite: uno para el código que corre en el navegador y otro para los archivos de configuración, que corren en 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` une los dos, y **`tsc -b`** (*build mode*) los revisa juntos. **`jsx: "react-jsx"`** es lo que permite escribir JSX, y **`lib: ["DOM"]`**, usar **`document`** y **`window`**.

Los estilos

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`** define los colores propios, como en «Estilos»: la marca, un turquesa y un coral, y un color claro por tipo de Pokémon.
- **`@custom-variant dark`** cambia cómo funciona **`dark:`**: el tema oscuro no lo decide la preferencia del sistema, sino un atributo **`data-theme`** en **`<html>`**, que pone el **`ThemeProvider`** de más abajo. Así la

persona elige el tema en la app.

- **`@apply`** aplica clases de Tailwind dentro del CSS, para los colores del fondo y del texto.

El punto de entrada

apps/web/src/main.tsx

```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>,
)
```

El router y el **`QueryClient`** se crean una vez, fuera del render, como en «Estado global».

apps/web/src/app/queryClient.ts

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

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

Un dato se considera fresco durante 30 segundos, y una query que falla se reintenta una vez, no las tres de «TanStack Query».

apps/web/src/app/Providers.tsx

```tsx
import { QueryClientProvider, type QueryClient } from '@tanstack/react-query'
```

```tsx
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` junta todos los proveedores en un componente. Las pruebas lo usan igual que la app, y cada pantalla nueva que necesite uno se suma aquí.

Los idiomas

La web está en español y en inglés, con el diccionario de «Internacionalización». Todos los archivos de este bloque van en **`apps/web/src/i18n/`**.

apps/web/src/i18n/es.ts

```tsx
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` es el tipo del diccionario en español: **`typeof es`**. El inglés tiene que tener la misma forma:

apps/web/src/i18n/en.ts

```
import type { Messages } from './es.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` hace que una clave que falta o que sobra no compile: cada pantalla nueva suma sus textos a los dos idiomas.

apps/web/src/i18n/i18nContext.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

```
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)`** pasa la función, sin llamarla: React usa su resultado solo en el primer render, para leer el idioma guardado (en desarrollo, con **`StrictMode`**, la llama dos veces). Es una **inicialización perezosa**.
- El efecto sincroniza con algo externo a React: **`localStorage`** y el atributo **`lang`** de **`<html>`**, que usan los lectores de pantalla para pronunciar en el idioma correcto.
- El Context y su hook van en un archivo aparte del proveedor: así cada archivo exporta o componentes o funciones, que es lo que pide el refresco en caliente de Vite.

El tema

Con la misma forma, el tema claro u oscuro. Los dos archivos van en **`apps/web/src/shared/theme/`**:

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

```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

```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>
}
```

Si no hay un tema guardado, se usa la preferencia del sistema, con **`matchMedia`**.

El cliente de la API

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

```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

/** 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'`**: todos los pedidos van al mismo origen, y el proxy de Vite los lleva a la API.

- **`withCredentials: true`** es lo que haría falta, del lado de la web, si un día la API viviera en otro origen; además, la API tendría que aceptarlo por CORS y la cookie necesitaría **`SameSite=None`**. Con el proxy, los pedidos son del mismo origen y el navegador ya manda la cookie.
- El interceptor avisa cuando la API responde 401, la sesión venció, salvo en **`/auth/`**, donde un 401 significa «no hay sesión» o «contraseña incorrecta». Quién reacciona a ese aviso lo decide el paso 06, con **`setUnauthorizedHandler`**.
- **`toApiError`** convierte cualquier error en un **`ApiError`**: el de la API si tiene esa forma, uno de red si no hubo respuesta, o uno genérico. Así la interfaz siempre sabe qué mostrar.

El router y el 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 /> },
    ],
  },
]
```

Las rutas son datos, en el modo data de «Navegación con React Router»: la app y las pruebas usan la misma lista. **`index: true`** es la **ruta índice**, la que se muestra en la dirección exacta del padre, **`/`**.

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'
  }`
```

```tsx
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>
  )
}
```

El layout tiene el nombre de la app, la navegación, el selector de idioma, el botón del tema y el aviso de propiedad intelectual al pie. **`sr-only`** es una clase de Tailwind que oculta un texto en la pantalla pero no para los lectores de pantalla: el **`<select>`** tiene su etiqueta «Idioma» aunque no se vea. La etiqueta va separada, unida con **`htmlFor`** e **`id`**: si envolviera al **`<select>`**, su nombre para un lector de pantalla incluiría la opción elegida. El botón del tema muestra un símbolo, y su **`aria-label`** dice qué hace.

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 es el **errorElement** de la ruta raíz: si algo falla al renderizar, se ve esto en lugar de una pantalla en blanco.

Las pruebas de la web

Los archivos de apoyo de las pruebas van en **apps/web/src/test/**.

apps/web/src/test/setup.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

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

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

El servidor de MSW empieza sin handlers: cada prueba suma los que necesita con **`server.use`**. Además, cada prueba deja **`localStorage`** vacío, para que el idioma o el tema de una no afecten a la siguiente.

apps/web/src/test/renderApp.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` monta la app entera en una dirección, con **`createMemoryRouter`**: un router que guarda la dirección en memoria en lugar de usar la barra del navegador, que en jsdom no existe. Cada prueba recibe un **`QueryClient`** nuevo, sin reintentos.

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()
  })
})
```

Los cambios en la raíz

El linter suma las reglas de React y de accesibilidad (**`jsx-a11y`**), y ya revisa también la web. Desde aquí, cada paso cierra con **`pnpm test`**; si quieres adelantarte al CI del paso 12, corre también **`pnpm lint`** y **`pnpm typecheck`**, que el paso 11 junta con el resto. **`exhaustive-deps`** va como error: una dependencia que falta en un efecto no pasa.

.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 no instala una versión publicada hace menos de un día (1440 minutos, su **`minimumReleaseAge`** por defecto), para dar tiempo a detectar un paquete malicioso antes de que llegue a tu máquina (pnpm, *Dependency resolution settings*). Las versiones que lista **`minimumReleaseAgeExclude`** en **`pnpm-workspace.yaml`** salieron el mismo día en que construí la app, y la excepción permitía instalarlas. Hoy esos paquetes tienen más de un día: la excepción ya no hace falta, pero tampoco molesta.

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
```

Instalar, probar y abrir

```
pnpm install
pnpm test
```

Vitest tiene que terminar con **`Tests 32 passed (32)`**: las cuatro pruebas nuevas son las del layout.

```
pnpm dev
```

`pnpm dev` levanta a la vez la API y la web, en la misma terminal (con **`Ctrl+C`** se detienen las dos). Vite anuncia la dirección de la web, **`http://localhost:5173`**: ábrela y vas a ver el inicio, con la navegación, el selector de idioma, el botón del tema y el aviso al pie. Cambia el idioma y el tema, recarga la página, y siguen ahí.

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

Descarga el código del paso 05.

Paso 06: el acceso

En este paso vas a construir la primera pantalla: el login, la sesión que lo recuerda y las rutas que la piden.

Los archivos nuevos de este paso van en **`apps/web/src/features/auth/`**, la carpeta de la funcionalidad de acceso, como en «Organizar por funcionalidad».

Hablar con la API

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

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

Cada respuesta se valida con el esquema del contrato antes de usarla: si la API y la web quedaran en versiones distintas, el error aparece aquí, con un mensaje claro. **`fetchMe`** convierte el 401 en **`null`**: no tener sesión no es un error.

La sesión

La cookie es **`httpOnly`**, así que el JavaScript de la página no puede leerla. Para saber si hay alguien adentro, la web le pregunta a la API al abrirse: es la **rehidratación** de la sesión.

apps/web/src/features/auth/authContext.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 tiene tres valores posibles, y cada uno significa algo: **undefined** mientras se pregunta, **null** si no hay sesión y el usuario si la hay. **unavailable** es **true** si la API no pudo contestar (un 500, o sin red): no es lo mismo que no tener sesión.

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>
  )
}
```

- La sesión es un dato del servidor, así que vive en TanStack Query, con la key **`ME_KEY`**. **`staleTime: Infinity`** hace que no se vuelva a pedir sola: cambia solo cuando alguien entra o sale.
- **`signIn`** y **`signOut`** escriben directo en la caché con **`setQueryData`**, sin otro pedido.
- El efecto registra qué hacer cuando la API responde 401 en medio de la sesión: olvidar al usuario y borrar sus datos de la caché, como al salir, para que quien entre después no vea los del anterior. Su limpieza lo desregistra.
- **`retry`** vuelve a preguntar, con **`refetch`**.
- Al salir se borran de la caché los datos del entrenador anterior. No se usa **`clear()`**: borraría también la sesión, y la query activa la volvería a pedir enseguida.

Las rutas que piden sesión

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 es una ruta de layout sin dirección propia: envuelve a sus rutas hijas. Mientras pregunta, muestra un **<output>**, el elemento de HTML para un resultado o un estado (su rol es **status**; como aparece junto con su texto, un lector de pantalla puede no anunciarlo). Si la API no pudo contestar, muestra el problema con un botón para reintentar, en vez de quedarse esperando para siempre. Sin sesión, **<Navigate>** lleva al login con **replace** (el login reemplaza la entrada del historial, y «atrás» no vuelve a la página protegida) y guarda en **state** la dirección a la que querías ir, con sus filtros (**pathname** más **search**).

El 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>
  )
}

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>
  )
}
```

Es el login con una Action de «Formularios», con lo que una app real le suma:

- la Action valida con el esquema del contrato antes de llamar a la API, y traduce el código de error a un mensaje en el idioma activo;
- devuelve el correo y el campo lo usa como **`defaultValue`**: React vacía los campos no controlados al terminar la Action, y así el correo no se pierde cuando la contraseña es incorrecta;
- la Action solo guarda el usuario. La redirección la hace el **`<Navigate>`** de arriba, que se dibuja apenas hay un usuario: una navegación declarada, en vez de llamar a **`navigate()`** dentro de la Action y competir con ella;
- **`autoComplete`** le dice al navegador qué es cada campo, para que el gestor de contraseñas los llene;

- **required** marca los campos como obligatorios para los lectores de pantalla, y **noValidate** deja la validación al esquema.

Las rutas y el layout

Las rutas suman el login, y la ruta índice pasa a estar dentro de **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 /> },
     ],
   },
```

El layout ofrece entrar o salir, según haya sesión:

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()` marca que la promesa de **`signOut`** no se espera a propósito.

El inicio saluda al entrenador:

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>
   )
```

Y **`Providers`** suma el **`AuthProvider`**. Va adentro del **`QueryClientProvider`**, porque usa 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>
   )
```

Los textos nuevos, en los dos idiomas:

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: {
```

Las pruebas

Los handlers de MSW que comparten las pruebas: una sesión iniciada, con Ana, y una sin sesión.

apps/web/src/test/handlers.ts

```
import { http, HttpResponse } from 'msw/http'
```

```ts
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. */
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

```ts
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()
  })
})
```

Las pruebas del layout necesitan ahora una sesión, porque el inicio la pide:

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')
```

Probar y entrar

```
pnpm test
```

Vitest tiene que terminar con **`Tests 37 passed (37)`**. Después, con **`pnpm dev`**, abre **`http://localhost:5173`**: la app te lleva al login. Entra con **`ana@dex506.test`** y **`dex506-demo`**, y vas a ver «Hola, Ana». Prueba una contraseña incorrecta: aparece el error, y el correo sigue en su campo. Recarga la página con la sesión iniciada: la app pregunta a la API y te deja adentro.

> *“El error frecuente: la cookie no llega y la app vuelve al login después de cada recarga. Pasa si la web llama a la API en otro origen, sin el proxy, o si mezclas **`localhost`** y **`127.0.0.1`**: la cookie es de un nombre de dominio, y esos son dos. Abre siempre **`http://localhost:5173`**.”*

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

Descarga el código del paso 06.

Paso 07: la Pokédex

En este paso vas a construir la lista de los 151 Pokémon, con buscador, filtro por tipo, paginación y dos vistas.

Los datos

Los archivos de la Pokédex van en **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) arma, con la función genérica del contrato, el esquema de una página de resúmenes. **params: query** hace que axios convierta el objeto en la dirección: **/pokemon?q=lag&page=1&limit=24**. Los tipos también se validan, contra el esquema de tipos del contrato.

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

```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,
  })
}
```

Es el hook de «TanStack Query», con la consulta entera en la query key: cada combinación de filtros es una entrada distinta de la caché, y **`keepPreviousData`** deja ver la página anterior mientras llega la siguiente.

La preferencia de vista

La grilla o la lista es una preferencia de la interfaz que sobrevive a recargar la página: un store de Zustand con **`persist`**, como en «Estado global».

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

```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' },
```

```
  ),
)
```

Las piezas compartidas

Tres piezas que van a usar varias pantallas, en **`apps/web/src/shared/`**. El **`useDebounce`** genérico de «TypeScript avanzado»:

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

```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
}
```

La imagen de un Pokémon, con una silueta propia si la imagen no carga:

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

```ts
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"` hace que el navegador descargue la imagen recién cuando está por aparecer en pantalla. Las criaturas de las pruebas tienen direcciones de imagen que no existen a propósito, y por eso se ve la silueta.

La insignia de un tipo, con el color de **`@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` lista las 18 clases completas: es el cuidado de «Estilos», Tailwind solo genera las clases que encuentra escritas enteras. **`Record<PokemonType, string>`** obliga a tener un color por cada tipo: si el contrato sumara uno, esto no compilaría.

La tarjeta y la página

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

```
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
}

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>
  )
}
```

- **Los filtros viven en la dirección**, con **`useSearchParams`** y **`replace`**, como en «Navegación con React Router». **`useFilters`** es un hook propio que lee los filtros de la dirección, los valida (un **`type`** que no existe se ignora) y los cambia.
- **`filters`** va con **`useMemo`** y **`update`** con **`useCallback`**, como en «Rendimiento»: así no cambian de identidad mientras la dirección no cambie. La página recibe **`update`** con el nombre **`setFilters`**.
- **Lo que escribes va primero a un estado local**, **`text`**, y a la dirección recién cuando dejas de escribir, con **`useDebounce`**: así no hay un pedido por letra. El efecto que lo lleva a la dirección corre solo cuando cambia el texto con debounce; leer los filtros actuales es un evento, con **`useEffectEvent`**, como en «useEffect sin miedo».
- **Si la dirección cambia desde afuera**, por ejemplo con el enlace «Pokédex» de la navegación, el campo la sigue. Es un estado ajustado durante el render (**`urlText`**), la forma que recomienda React en vez de otro efecto: cuando la **`q`** de la dirección cambia y no es la que escribiste, el campo toma la nueva.
- **El select de tipo** tiene su etiqueta separada, como el de idioma del paso 05.
- **Los cuatro estados de una lista que viene del servidor:** cargando, error (con un botón para reintentar), vacía y con datos.
- **La paginación** es un **`<nav>`** con su **`aria-label`**, y los botones se deshabilitan en la primera y la última página.
- **Los botones de vista** usan **`aria-pressed`**, que les dice a los lectores de pantalla cuál está activo.

Los cambios

La ruta nueva, dentro de las que piden sesión:

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, como dependencia nueva:

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` permite importar un archivo **`.json`**: lo necesitan los handlers de las pruebas, que leen las criaturas de **`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"]
 }
```

Los textos de la Pokédex y los nombres de los 18 tipos, en los dos idiomas:

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: {
```

Las pruebas

Los handlers suman la lista y los tipos, con las criaturas inventadas, filtrando como la API real:

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()
  })
})
```

Probar y usar

```
pnpm install
pnpm test
```

Vitest tiene que terminar con **`Tests 44 passed (44)`**. Con **`pnpm dev`**, entra y abre la Pokédex: busca «char», filtra por un tipo, cambia de página y de vista. Con las criaturas inventadas, busca «lag». Son doce, así que caben en una sola página: para ver la paginación hacen falta los 151 de PokéAPI. Copia la dirección en otra pestaña y vas a ver los mismos filtros. Para ver el estado de error, abre las herramientas del navegador, en la pestaña de red bloquea la dirección del pedido a **`/api/pokemon`** (clic derecho sobre el pedido, «Block request URL» en Chrome) y recarga: aparece el error con su botón «Reintentar».

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

Descarga el código del paso 07.

Paso 08: la ficha

En este paso vas a construir el detalle de un Pokémon, con React 19 a fondo: carga diferida, favorito optimista, View Transition, pestañas con **`<Activity>`** y un comparador.

Los archivos nuevos van en **`apps/web/src/features/pokemon/`**.

Los datos

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

```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` usa **`PUT`** para marcar y **`DELETE`** para desmarcar, las dos rutas idempotentes de la API.

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

```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` es una función que arma la key de un Pokémon: la ficha y el comparador la usan igual, y si visitaste la ficha, el comparador lo encuentra en la caché. **`staleTime: Infinity`**: los datos de un Pokémon no cambian.

El favorito optimista

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

```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'

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>
  )
}
```

Es el patrón de «useOptimistic» con TanStack Query, y un botón de alternar accesible:

- el valor real, **`saved`**, sale de la query de favoritos, y **`shown`** es lo que se ve mientras se guarda;
- **`setShown`** va dentro de **`startTransition`**, y **`mutateAsync`** espera el guardado;
- **`onSettled`** invalida los favoritos haya salido bien o mal: la caché vuelve a la verdad del servidor;
- si falla, React vuelve solo al valor real y **`failed`** muestra el aviso;
- el nombre del botón no cambia con el estado, «Favorito»: lo dice **`aria-pressed`**, y la estrella es decorativa (**`aria-hidden`**). Un lector de pantalla anuncia «Favorito, botón, presionado»;
- mientras se guarda, el botón no se deshabilita: un botón deshabilitado pierde el foco del teclado. **`toggle`** ignora un segundo clic mientras hay uno en curso.

El comparador

Elegir dos Pokémon para comparar es estado de la interfaz que comparten la ficha y la Pokédex: un store de Zustand, sin **`persist`**, porque no tiene que sobrevivir a recargar.

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)` se queda con los dos últimos: elegir un tercero reemplaza al más viejo.

apps/web/src/features/pokemon/CompareBar.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 es la versión de **useQuery** para una lista de queries que cambia: una por cada Pokémon elegido. La tabla usa **<th scope="col">** y **<th scope="row">**, para que un lector de pantalla lea cada número con su estadística y su Pokémon.

La página

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`**: la ruta la carga con **`lazy`** e importa el valor por defecto del módulo.
- **Un Pokémon que no existe** es un 404 de la API, con el código **`POKEMON_NOT_FOUND`**, y se muestra con su camino de vuelta.
- **`<ViewTransition name={…}>`** lleva el mismo nombre que la imagen de la tarjeta: al pasar de la Pokédex a la ficha, el navegador anima la imagen de un lugar al otro. Hay una condición: las dos imágenes tienen que estar en la misma transición. Si la ficha todavía no está en la caché, lo primero que se ve es «Cargando la ficha…», y la imagen llega después, sin el viaje compartido. Al volver a un Pokémon ya visitado, la animación sí se ve completa.
- **El botón de comparar** sigue la misma regla que el favorito: su nombre es siempre «Comparar», y **`aria-pressed`** dice si está elegido.
- **Las pestañas** siguen el patrón de pestañas de ARIA: **`role="tablist"`**, **`role="tab"`** y **`role="tabpanel"`**, unidos con **`aria-controls`** y **`aria-labelledby`**. Solo la pestaña seleccionada está en el orden del tabulador (**`tabIndex`**), y las flechas pasan a la otra y le llevan el foco.
- **`<Activity>`** oculta la pestaña que no se ve sin desmontarla, como en «Activity»: el filtro de movimientos conserva lo escrito.

- **`StatsPanel`** dibuja una barra por estadística. La barra es decorativa (**`aria-hidden`**): el número que la acompaña es el dato.

La tarjeta como enlace

La tarjeta de la Pokédex pasa a ser un enlace a la ficha:

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} />
```

- El enlace está en el nombre, y su **`after:absolute after:inset-0`** lo estira sobre toda la tarjeta: se toca en cualquier lugar, y para el teclado es un solo destino.
- La imagen pasa a **`alt=""`**: el nombre ya está en el enlace, y repetirlo haría que un lector de pantalla lo diga dos veces.
- La **`<ViewTransition>`** de la imagen lleva el nombre que la ficha repite.

La Pokédex muestra el comparador arriba:

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">
```

La ruta de la ficha, con carga diferida:

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 /> },
```

Si abres una ficha directo (recargas en **`/pokedex/3`** o te pasan el enlace), su código todavía no se descargó cuando el router arranca. **`hydrateFallbackElement`** es lo que muestra mientras tanto, dentro del layout; sin él, React Router no muestra nada y avisa en la consola que falta:

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>
}
```

La web suma Zod como dependencia propia: las dependencias de un paquete del workspace no se ven desde otro, y **`api.ts`** lo usa directo para validar la lista de favoritos.

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": {
```

Los textos nuevos:

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: {
```

Las pruebas

Los handlers suman el detalle de un Pokémon y unos favoritos en memoria, que cada prueba arranca como quiere:

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

```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()
  })
})
```

`delay` de MSW demora una respuesta, para comprobar lo que se ve mientras se guarda; **`useCompare.setState`** deja el comparador vacío antes de cada prueba, porque el store de Zustand es un módulo y sobrevive entre pruebas.

Probar y usar

```
pnpm install
pnpm test
```

Vitest tiene que terminar con **`Tests 52 passed (52)`**. En la app, abre un Pokémon desde la Pokédex, vuelve y ábrelo otra vez: la segunda vez, la imagen viaja de la tarjeta a la ficha. Marca el favorito, cambia de pestaña con el mouse y con las flechas, escribe en el filtro de movimientos y vuelve. Elige dos para comparar y vuelve a la Pokédex: la tabla está arriba.

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

Descarga el código del paso 08.

Paso 09: el taller de equipo

En este paso vas a construir el equipo de seis y el formulario de dos pasos que agrega y edita a sus miembros.

Los archivos nuevos van en **`apps/web/src/features/team/`**.

Los datos

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)
}
```

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

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

```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 }),
  })
}
```

Guardar es una mutación, y cuando sale bien, el equipo guardado en la caché quedó viejo: se invalida, como en «Mutaciones e invalidación». **`useSaveMember`** sirve para agregar y para editar: con un **`memberId`**, edita.

El esquema del formulario

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

```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>
```

El formulario parte del esquema del contrato, **`TeamMemberInputSchema`**: las reglas del apodo y del nivel son las mismas en el navegador y en la API. Solo cambia la forma de los movimientos, porque **`useFieldArray`** necesita objetos, no textos sueltos. **`z.input`** y **`z.output`** son el tipo de los datos antes y después de validar. Aquí coinciden, pero **`useForm`** los recibe por separado: si el esquema transformara un dato, por ejemplo un texto en número, cada lado tendría su tipo correcto.

El formulario

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>
  )
}
```

Es React Hook Form con Zod, de «Formularios», con tres cosas nuevas:

- **Dos pasos.** El primero pide el apodo y el nivel; el segundo, los movimientos. **`trigger`** valida solo los campos del primer paso antes de avanzar. El paso que no se ve queda con **`hidden`**, sin desmontarse: lo escrito se conserva al volver. Al cambiar de paso, el botón que acabas de tocar desaparece, así que el efecto con **`setFocus`** lleva el foco al primer campo del paso nuevo, para que el teclado no quede perdido. La ref **`shownStep`** hace que eso pase solo con un cambio de paso, no al montar.

- **Movimientos dinámicos.** **`useFieldArray`** maneja una lista de campos que crece y se achica: **`append`** suma un movimiento, **`remove`** quita uno, y cada campo tiene un **`id`** propio para su **`key`**.
- **Un movimiento vacío** falla en su propio campo, **`moves.N.name`**: **`moveError`** toma el primero de esos mensajes para mostrarlo debajo de la lista, y cada select vacío queda con **`aria-invalid`**.
- **Los errores del servidor van a su campo.** **`onSave`** devuelve el **`ApiError`** si la API rechazó el miembro. Si trae errores por campo, cada uno va a su campo con **`setError`**, y un error del apodo o del nivel vuelve al primer paso para que se vea. Si no trae campos, como el equipo lleno, se muestra como alerta del formulario.

Las páginas

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` es el mismo para agregar y editar: carga el Pokémon y, al editar, el miembro, y pasa sus datos como **`defaultValues`**. **`navigate('/team')`** va en **`onSave`**, el manejador del guardado: ahí se sabe que se guardó.

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>
  )
}
```

El resumen, con el nivel promedio, es un dato derivado: se calcula en el render con **`averageLevel`** del contrato. El botón de quitar no se deshabilita mientras quita (perdería el foco del teclado): ignora un segundo clic, y si la API falla, la página lo avisa.

Los cambios

Las rutas del equipo:

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 /> },
```

El enlace en la navegación:

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. */}
```

La ficha ofrece agregar el Pokémon al equipo:

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}
```

Los mensajes de validación de Zod salen en inglés salvo que se configure su idioma. El proveedor de idiomas lo configura junto con el resto:

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 y su adaptador de Zod, con la misma excepción de **`minimumReleaseAge`** del paso 05 para el adaptador:

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'
```

Los textos nuevos:

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: {
```

Las pruebas

Un equipo en memoria por prueba, con las cuatro operaciones:

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()
  })
})
```

Probar y usar

```
pnpm install
pnpm test
```

Vitest tiene que terminar con **`Tests 59 passed (59)`**. En la app, desde una ficha, toca «Agregar al equipo»: escribe un apodo demasiado largo y mira el error en su campo; sigue al paso 2, elige movimientos y guarda. Agrega seis y prueba un séptimo: la API responde con el 422 de equipo lleno, y el formulario lo muestra.

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

Descarga el código del paso 09.

Paso 10: los gimnasios

En este paso vas a construir los ocho gimnasios, la alineación con Redux Toolkit y el reto, con el manejo de sus errores.

Por qué Redux aquí

La alineación de cada gimnasio es estado de la interfaz, y Zustand alcanzaría, como en la preferencia de vista. Dex 506 usa Redux Toolkit para que lo construyas una vez en una app real, como dice «Arquitectura, calidad y seguridad».

El 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** es una función, como en «Estado global»: la app crea un store y cada prueba crea el suyo.
- **useAppSelector** y **useAppDispatch** son **useSelector** y **useDispatch** con los tipos del store, hechos una vez con **withTypes**: así ningún componente tiene que escribir **RootState**.

La alineación

Los archivos de los gimnasios van en **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,
)
```

- El estado guarda una alineación por gimnasio: un objeto de **`gymId`** a la lista de miembros, en el orden en que se eligieron.
- **`toggleMember`** suma o quita un miembro, con un máximo de tres. **`keepOnly`** quita de todas las alineaciones a los miembros que ya no están en el equipo.
- **`selectLineup`** es un selector con **`createSelector`** que recibe un argumento, el **`gymId`**. **`EMPTY`** es un arreglo vacío fijo: si el gimnasio no tiene alineación, devuelve siempre el mismo, y no uno nuevo en

cada render. Para ser justos, aquí **`createSelector`** suma poco: el arreglo guardado ya es siempre el mismo. Está para que lo veas en su forma más simple; rinde cuando el selector calcula algo nuevo, como un filtro o un total.

Los datos

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
```

Las dos keys van en un archivo propio porque las usan las dos páginas.

Las páginas

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>
  )
}
```

- La alineación vive en Redux, y el equipo, en TanStack Query: cada dato en su lugar, como en la tabla «No todo estado es igual» de la parte 2. El efecto con **`keepOnly`** mantiene los dos de acuerdo cuando el equipo cambia.
- Cuando la alineación ya tiene tres elegidos, los demás miembros quedan deshabilitados.
- El reto es una mutación. Si la API lo rechaza con un 422, por ejemplo por un nivel promedio bajo, **`toApiError`** trae su mensaje y se muestra como alerta. Si ganas, las medallas se invalidan y la lista de gimnasios la muestra.
- El resultado aparece dentro de una región con **`aria-live="polite"`** que está siempre en la página: un lector de pantalla anuncia lo que aparece adentro, sin interrumpir lo que estaba leyendo. Si la región apareciera junto con el texto, podría no anunciarse.

Los cambios

El store se crea una vez en **`main.tsx`** y llega por **`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>
   )
 }
```

Las pruebas arman su propio 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 }
 }
```

Las rutas y la navegación:

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 y su enlace con React:

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"
```

Al instalar, pnpm 12 avisa «Agent skills awaiting approval: @reduxjs/toolkit». Redux Toolkit trae instrucciones para asistentes de programación con inteligencia artificial, y pnpm no las copia al proyecto sin tu permiso (pnpm, *Build settings*). La app no las necesita: puedes ignorar el aviso, o correr **`pnpm approve`** y rechazarlas para que no vuelva a aparecer.

Los textos nuevos:

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: {
```

Las pruebas

Los handlers suman los gimnasios, las medallas y el reto:

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)),
+  ]
+}
```

El reducer y el selector se prueban solos, sin React, porque son funciones puras:

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 compara la identidad: la prueba comprueba que el selector devuelva el mismo arreglo, no uno igual.

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')
  })
})
```

Probar y usar

```
pnpm install
pnpm test
```

Vitest tiene que terminar con **`Tests 65 passed (65)`**. En la app, con tres miembros en tu equipo, abre el Gimnasio del Caribe, elige la alineación y reta: con tres miembros de nivel 10 vas a ver «¡Victoria!», la bitácora del combate y la Medalla Arrecife en la lista de gimnasios. Prueba también el Gimnasio del Cerro, que exige nivel promedio 40: la API lo rechaza con su mensaje. Si instalaste las Redux DevTools en el navegador, ves pasar cada acción.

```
git add .
git commit -m "paso 10: los gimnasios"
```

Descarga el código del paso 10.

Paso 11: calidad

En este paso vas a cerrar la calidad de la app: pruebas de punta a punta, una auditoría de accesibilidad, la medición de rendimiento y una revisión de seguridad.

Playwright

Las pruebas de punta a punta usan Playwright, como en «Pruebas», y la auditoría de accesibilidad usa **axe**, una biblioteca que revisa una página contra las pautas WCAG. Se suman a la raíz del monorepo, con sus 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` revisa también los tipos de las e2e y de la medición, y **`e2e`** y **`perf`** las corren. Después de instalar, Playwright necesita descargar su navegador una vez. **`pnpm exec`** corre un programa instalado en el proyecto, como **`npx`**:

```
pnpm install
pnpm exec playwright install chromium
```

La configuración levanta la API y la web antes de las pruebas, y las apaga al terminar:

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`** arranca la API con las criaturas inventadas y en memoria (**`DEX_DB=memory`**, que se suma en este paso), y la web con el servidor de desarrollo de Vite. **`url`** es la dirección que Playwright espera que responda antes de empezar.
- Las variables de entorno van en el archivo y no en el comando: así el mismo **`pnpm e2e`** funciona en macOS, Linux y Windows.
- **`workers: 1`** corre las pruebas de a una, porque comparten la misma API.
- **`reuseExistingServer: false`** y **`--strictPort`** hacen que falle si los puertos 3001 o 5173 están ocupados: antes de **`pnpm e2e`**, detén el **`pnpm dev`** con **`Ctrl+C`**.

e2e/tsconfig.json

```
{
  "extends": "../tsconfig.base.json",
  "compilerOptions": {
    "lib": ["ES2023", "DOM"],
    "types": ["node"],
    "module": "nodenext",
```

```
    "moduleResolution": "nodenext"
  },
  "include": [".", "../playwright.config.ts"]
}
```

La API suma el modo en memoria, con la variable **`DEX_DB`**: con **`memory`**, los datos de los entrenadores no se guardan en el archivo y se pierden al apagarla; sin la variable, o con **`file`**, la API funciona como hasta ahora:

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
```

Las pruebas de punta a punta

El recorrido completo: entrar, buscar, abrir una ficha, armar un equipo y ganar un gimnasio. También el login que vuelve a la página protegida y la caída simulada de la API.

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` intercepta los pedidos del navegador: la última prueba agrega **`fail=1`** al pedido de la lista, y la Pokédex muestra el error.

La View Transition se comprueba contando las llamadas a **`document.startViewTransition`**, la API del navegador que usa React:

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 corre un script en la página antes que el de la app, y aquí envuelve la función del navegador para contarla.

La auditoría de accesibilidad

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([])
  })
}
```

Recorre siete pantallas en los dos temas y junta las violaciones de todas antes de comparar: así un error en la primera no esconde los demás. Cada pantalla está lista cuando su título principal está en pantalla y no queda nada «Cargando». No se espera a que la red quede quieta: Playwright lo desaconseja y recomienda comprobar lo que se ve (Playwright, *Page: waitForLoadState*).

La primera vez que corrió, la auditoría encontró tres problemas, y este paso los corrige. El primero: el coral de los errores no llegaba al contraste de 4.5:1 en el tema oscuro. El tema oscuro pasa a usar un coral más claro (**accent-400**), y el claro, uno más oscuro (**accent-700**). En el tema claro, el coral original (**accent-600**) ya pasaba la auditoría; con el **accent-700** el texto de error se lee mejor. Los cuatro diffs que siguen aplican ese cambio en los textos y bordes de error:

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>
       )}
```

El segundo: la silueta de una imagen sin **`alt`** era un **`role="img"`** sin nombre. Ahora, si la imagen es decorativa, se oculta a los lectores de pantalla:

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"
```

El tercero: un **`<select>`** envuelto en su **`<label>`** sumaba la opción elegida a su nombre («Movimiento 1 Elige un movimiento»). Los selects de idioma y de tipo ya tienen la etiqueta separada desde sus pasos; los de movimientos la separan aquí, unida con **`htmlFor`** y un id de **`useId`**, como en «Refs y el DOM», porque hay uno por movimiento. El mismo prefijo va en los ids de los mensajes de error, y el borde rojo del tema oscuro queda solo para los campos con error:

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>
       )}
```

La medición de rendimiento

React Compiler se activa en la web, como en «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` corre Babel dentro de Vite, y **`reactCompilerPreset`** es la configuración del compilador que trae el plugin de React.

Para medir el caso más caro, la Pokédex suma una vista con los 151 en una página:

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}`,
```

La medición es otra configuración de Playwright: el build de producción servido con **`vite preview`** (un servidor local que muestra el build tal como quedaría en producción) y la API con los 151 Pokémon de PokéAPI.

playwright.perf.config.ts

```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`** es una orden de Chrome DevTools, que Playwright manda por una sesión **CDP** (*Chrome DevTools Protocol*): hace la CPU cuatro veces más lenta, como un teléfono modesto.
- Las imágenes se bloquean, para que la red no meta ruido.
- Por cada tecla, el script mide desde que se presiona hasta el cuadro siguiente: el trabajo de React más el dibujo.

El método y los resultados que obtuve, que son los de la tabla de la parte 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.
```

El zip trae también los tres archivos de resultados, en **perf/resultados/**. Para repetir la medición:

```
pnpm --filter @dex506/api seed
PERF_VARIANTE=compilador pnpm perf
```

En Windows, en PowerShell: **`$env:PERF_VARIANTE="compilador"; pnpm perf`**.

La revisión de seguridad

El último archivo del paso deja por escrito qué se revisó, dónde y qué queda afuera:

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                                                
```

```
hilos de Node, y eso lo frena un límite de intentos (fuera de alcance)
| `apps/api/src/store.ts`                                                    |
| Las ayudas para practicar en producción          | `?delay` y `?fail` no hacen nada
con `NODE_ENV=production`
| `apps/api/src/middleware/network.ts`                                       |
| Fugas en los errores                             | Un error inesperado responde 500
con un mensaje genérico; el detalle va solo al log
| `apps/api/src/middleware/errors.ts`                                        |
| 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.
```

Probar

```
pnpm install
pnpm lint
pnpm format:check
pnpm typecheck
pnpm test
pnpm e2e
```

`pnpm install` va primero porque este paso suma dependencias en dos **`package.json`** (el aviso de las «agent skills» del paso 10 puede volver a aparecer). **`pnpm format:check`** revisa que todo tenga el formato de Prettier; si algo no lo tiene, **`pnpm format`** lo corrige. Vitest tiene que terminar con **`Tests 66 passed (66)`**, y Playwright con **`6 passed`**. Si una prueba e2e falla, Playwright deja una traza en **`test-results/`**: ábrela con **`pnpm exec playwright show-trace`** y la ruta de su archivo **`trace.zip`**, y vas a ver cada paso de la prueba con la pantalla de ese momento.

```
git add .
git commit -m "paso 11: calidad"
```

Descarga el código del paso 11.

Paso 12: integración continua

En este paso vas a hacer que cada cambio que subas pase por todas las comprobaciones, en una máquina que no es la tuya.

Qué es

La **integración continua** (*CI*) es correr el lint, los tipos, las pruebas y el build automáticamente en cada cambio que subes, en un servidor limpio. Si algo falla ahí, el cambio no está listo, aunque en tu máquina funcione. **GitHub Actions** es el servicio de integración continua de GitHub: lee **workflows**, archivos YAML en **`.github/workflows/`** que dicen qué correr y cuándo. Cada trabajo corre en un **runner**, una máquina limpia que GitHub presta para eso. En un repositorio público, los runners estándar son gratis; en uno privado, el plan gratuito incluye un monto de uso por mes (GitHub, *GitHub Actions billing*).

El 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** dice cuándo corre: en cada **push** a **main** y en cada **pull request**, el pedido de unir los cambios de una rama a otra, que se revisa antes de aceptarlo.
- **jobs** son los trabajos. **calidad** corre el lint, el formato, los tipos, las pruebas y el build; **e2e**, las pruebas de punta a punta. **needs: calidad** hace que **e2e** espere a que **calidad** pase: no tiene sentido abrir un navegador si las pruebas unitarias fallan.
- **runs-on: ubuntu-latest** elige el runner: la versión de Ubuntu que GitHub marca como la actual, que no siempre es la más nueva.
- **steps** son los pasos, en orden. **uses** corre una **acción**, un paso que publicó otro: bajar el código (**actions/checkout**), instalar pnpm (**pnpm/action-setup**, que lee la versión de **packageManager**) e instalar Node (**actions/setup-node**, que lee **.nvmrc** y guarda el caché de pnpm). **run** corre un comando.
- **--frozen-lockfile** hace que la instalación falle si el lockfile no coincide con los **package.json**, en vez de cambiarlo: en CI se instala exactamente lo que se probó.
- Las pruebas usan solo las criaturas de **fixtures/**: el CI nunca llama a PokéAPI.
- **if: failure()** corre el último paso solo si algo falló, y sube las trazas de Playwright como un **artefacto**, un archivo que se descarga desde la página del workflow.

Subirlo

Antes de subir, corre lo mismo que va a correr el CI, para no verlo fallar por algo que se ve en tu máquina:

```
pnpm lint
pnpm format:check
pnpm typecheck
pnpm test
pnpm build
```

Después, en GitHub, crea un repositorio nuevo llamado **`dex-506`**, público o privado (en uno público, los minutos del CI son gratis), y **vacío**: sin README, sin licencia y sin **`.gitignore`**, porque ya los tienes y un repositorio con archivos rechazaría el primer envío.

GitHub no acepta tu contraseña para subir por HTTPS. Antes del primer envío, inicia sesión desde la terminal con GitHub CLI, el programa oficial de GitHub para la terminal: **`gh auth login`**, y elige HTTPS. La otra opción es un **token de acceso personal**, una contraseña larga que generas en la configuración de GitHub y que pegas cuando git te pide la contraseña.

Conecta tu carpeta y sube el código:

```
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
```

- Una **rama** es una línea de commits con nombre. **`git branch -M main`** le pone **`main`** a la tuya, el nombre que espera el workflow.
- **`git remote add origin …`** le dice a git dónde está el repositorio de GitHub, con el nombre **`origin`**. Reemplaza **`<tu-usuario>`** por tu usuario.
- **`git push -u origin main`** sube la rama y recuerda el destino, para que los próximos envíos sean solo **`git push`**.

En la pestaña «Actions» del repositorio vas a ver el workflow correr, con sus dos trabajos en verde. Desde ahí, cada **`push`** lo vuelve a correr.

Descarga el código del paso 12.

Paso 13: el build de producción

En este paso vas a generar el build de producción y a servir la app entera desde un solo proceso de Node.

El build

`vite build` arma la web para producción: junta el código en pocos archivos, lo minifica (quita espacios y acorta nombres) y pone en el nombre de cada archivo una huella de su contenido (un hash, como el de las contraseñas, pero rápido y sin sal): si el contenido cambia, cambia el nombre. Así el navegador puede guardarlos en caché sin miedo a usar uno viejo. El resultado queda en **`apps/web/dist/`**. La ficha queda en un archivo aparte, por la carga diferida del paso 08.

La API sirve la web

En desarrollo hay dos servidores, la API y el de Vite, y el proxy los une. En producción no hay servidor de Vite: la API sirve también los archivos de la web, desde el mismo origen.

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`** responde los archivos del build tal cual: el HTML, el JavaScript, el CSS. Los de **`assets/`**, que llevan la huella en el nombre, salen con **`immutable`** y un año de caché: el navegador no los vuelve a pedir.
- La web es una SPA: una dirección como **`/pokedex/25`** no es un archivo, es una pantalla que decide React Router. Por eso toda dirección que no es de la API ni un archivo responde **`index.html`**. Sin eso, recargar en **`/pokedex/25`** daría un 404 del servidor. La expresión regular dice «cualquier dirección que no empiece con **`/api/`** ni termine en una extensión»: un archivo que no existe, como un **`/assets/viejo.js`** de un build anterior, es un 404 y no un **`index.html`** disfrazado de 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.
```

Con **`NODE_ENV=production`**, el servidor además sirve el build de la web.

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')
```

Es el punto de entrada de producción. Pone **`NODE_ENV`** antes de cargar el servidor, porque la forma de la línea de comandos (**`NODE_ENV=production node …`**) no funciona en 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` arma el build y levanta la API en modo producción.

La prueba

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')
  })
})
```

La prueba arma un build de mentira en una carpeta temporal y comprueba las tres reglas: una dirección de la SPA devuelve **`index.html`**, un archivo sale tal cual, y **`/api`** sigue siendo de la API, con su 404 de forma **`ApiError`**.

El README final

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`.
```

Probar y servir

```
pnpm test
pnpm start
```

Vitest tiene que terminar con **`Tests 67 passed (67)`**. **`pnpm start`** arma el build y, al armarlo, Vite avisa «Some chunks are larger than 500 kB after minification». No es un error: el archivo principal junta React, React Router, TanStack Query, Redux Toolkit y React Hook Form, y Vite sugiere dividirlo. Para esta app alcanza con la ficha en su archivo aparte; en una más grande, más rutas con **`lazy`** lo achican. Después, la API anuncia **`http://localhost:3001`**: abre esa dirección, sin el puerto de Vite, y vas a ver la app entera servida por un solo proceso. La cookie de la sesión ya es **`Secure`** (el atributo que la deja viajar solo por HTTPS), y aun así funciona sin HTTPS porque los navegadores tratan **`localhost`** como un origen seguro. Lo comprobé en Chromium. Si en otro navegador el login no te deja entrar (vuelves al formulario sin error), es que no guardó la cookie: prueba con Chrome.

> *"El error frecuente: la app abre, pero al recargar en una ficha aparece «No existe la ruta GET /pokedex/25». Falta la regla que devuelve **`index.html`** para las direcciones de la SPA, o está después de **`notFound`**: el orden de los middlewares importa."*

Qué haría falta para desplegarla

Esta guía termina en el build de producción en tu máquina. Para ponerla en internet, hace falta:

- **un servidor que corra Node 26**, con un proceso que lo vuelva a levantar si se cae;
- **HTTPS**, porque la cookie de la sesión es **`Secure`** en producción;
- **las variables de entorno** del **`.env.example`** (**`NODE_ENV=production`** lo pone **`production.ts`**);
- **quitar las cuentas de prueba**, Ana y Leo, cuya contraseña está en el repositorio;
- **los datos de PokéAPI**: correr **`pnpm --filter @dex506/api seed`** una vez en el servidor;
- **un lugar que no se borre** para **`apps/api/.data/`**, o mejor, una base de datos de verdad (ver el cierre).

```
git add .
git commit -m "paso 13: el build de producción"
```

Descarga el código del paso 13.

Paso 14: cierre

En este paso vas a ver qué le falta a Dex 506 para ser una app con usuarios reales, y qué aprendiste en toda la serie.

Lo que dejé afuera a propósito

La revisión de seguridad del paso 11 nombra tres cosas que una app real necesita y que esta guía no construye:

- **Un límite de intentos de login.** Hoy alguien puede probar contraseñas sin parar. Un límite por dirección IP y por cuenta lo frena.
- **Cabeceras de seguridad.** `Content-Security-Policy` le dice al navegador de dónde puede cargar scripts, lo que corta muchos XSS aunque haya un error; `Strict-Transport-Security` le pide que use siempre HTTPS.
- **Una base de datos de verdad.** El archivo JSON escribe de forma atómica y en orden, pero guarda todo en memoria y lo escribe entero en cada cambio: con muchos usuarios a la vez, hace falta una base de datos como PostgreSQL o SQLite.

También quedan afuera los mensajes de error de la API, que llegan en español aunque la interfaz esté en inglés: una API con usuarios de varios idiomas devolvería solo el código, y la web traduciría el mensaje.

Lo que aprendiste

En la parte 1 viste qué problema resuelve React y por qué nace TypeScript, y escribiste componentes con props, estado y estilos. En la parte 2 tu app habló con un servidor, navegó entre pantallas, validó formularios y eligió dónde vive cada dato. En la parte 3 viste cómo renderiza React por dentro, mediste antes de optimizar, usaste React 19 a fondo y probaste en tres niveles. Y en esta parte juntaste todo en una app completa: un contrato compartido, una API con sesión y pruebas, cinco pantallas, accesibilidad y seguridad revisadas, integración continua y un build de producción.

Bibliografía

- Node.js, Modules: TypeScript. Consultado el 18 de noviembre de 2026.
- pnpm, Installation, Dependency resolution settings, Build settings y CLI settings. Consultados el 18 de noviembre de 2026.
- Express, Error handling. Consultado el 18 de noviembre de 2026.
- PokéAPI, documentación v2, con su política de uso justo. Consultada el 18 de noviembre de 2026.
- Playwright, Page: waitForLoadState. Consultado el 18 de noviembre de 2026.
- GitHub, GitHub Actions billing. Consultado el 18 de noviembre de 2026.
- Lauren Tan, Joe Savona y Mofei Zhang, React Compiler v1.0, 7 de octubre de 2025. Consultado el 18 de noviembre de 2026.