Taller 2 - Introducción al Desarrollo Web Móvil

Grupo 1 — Equipo: SmartCoders

Integrantes

- Bastian Salinas — 21.848.994-K
- Benjamín Cuello — 21.682.135-1
- Benjamín Salas — 21.758.667-4
- Tomás Guerra — 21.664.344-5

Descripción General

InfoMóvil integra un ecosistema con **3 servicios backend propios** y un **frontend móvil empaquetado como aplicación Android (APK)** usando **Apache Cordova**.

La app mantiene las mismas funcionalidades del Taller 1 (Pokémon, países, clima y feriados), pero ahora **todos los datos provienen de nuestras propias APIs y bases de datos**.

APIs propias

API 1 — Pokémon (NestJS + PostgreSQL)

- **Endpoints**
  - `GET /pokemon?limit=&offset=` → lista de pokémon
  - `GET /pokemon/:idOrName` → detalle de un pokémon
- **Datos expuestos (contrato similar a PokeAPI):**
  - Listado:

```
{
  "count": 10,
  "results": [{ "name": "bulbasaur" }, { "name": "ivysaur" }]
}
```

  - Detalle:

```
{
  "id": 25,
  "name": "pikachu",
  "height": 4,
  "weight": 60,
  "types": [{ "slot": 1, "type": { "name": "electric" } }],
```

```
    "sprites": {
      "front_default": "/img/pokemon/25.png",
      "back_default": null,
      "other": {
        "official-artwork": {
          "front_default": "/img/pokemon/25_artwork.png"
        }
      }
    },
    "stats": [
      { "base_stat": 35, "stat": { "name": "hp" } },
      { "base_stat": 55, "stat": { "name": "attack" } }
    ]
  }
```

- **Puerto:** `http://localhost:3000`

API 2 — Países (Express + PostgreSQL)

- **Endpoints**
  - `GET /countries` → lista de todos los países
  - `GET /countries/search?name=Chile` → búsqueda por nombre
- **Datos expuestos (contrato similar a RestCountries):**

```
[
  {
    "name": { "common": "Chile", "official": "Republic of Chile" },
    "flags": { "png": "https://flagcdn.com/w320/cl.png", "svg":
"https://flagcdn.com/cl.svg" },
    "region": "Americas",
    "capital": ["Santiago"],
    "population": 19116201,
    "cca2": "CL"
  }
]
```

- **Puerto:** `http://localhost:4000`

API 3 — Clima y Feriados (FastAPI + MongoDB)

- **Endpoints**
  - `GET /weather?city=La%20Serena` → devuelve temperatura y viento actuales
  - `GET /holidays/{countryCode}/{year}` → devuelve feriados del país y año indicado
- **Datos expuestos (contrato similar a Open-Meteo y NagerDate):**
  - Clima:

```
{
  "coordenadas": {
    "name": "Coquimbo",
    "country_code": "CL",
    "latitude": -29.95,
    "longitude": -71.34
  },
  "clima": {
    "current": {
      "temperature_2m": 16.4,
      "wind_speed_10m": 5.1
    }
  }
}
```


- Feriados:


```
[
  { "date": "2025-01-01", "localName": "Año Nuevo", "name": "Año
Nuevo" },
  {
    "date": "2025-09-18",
    "localName": "Independencia Nacional",
    "name": "Independencia Nacional"
  }
]
```


- **Puerto:** `http://localhost:8000`

Frontend (Cordova + HTML + JS + Tailwind)

- Código fuente ubicado en `frontend/www/`, estructurado para reutilizarse dentro de Cordova (`www/`).
- Mantiene el diseño y la lógica del Taller 1, consumiendo **solo** nuestras APIs propias.
- **Configuración:** `frontend/www/js/config.js` define las URLs base de las APIs:

```
var BASE_URL_POKEMON = 'http://localhost:3000'
var BASE_URL_COUNTRIES = 'http://localhost:4000'
var BASE_URL_FASTAPI = 'http://localhost:8000'
```

Tecnologías a utilizar

Frontend

- HTML5
- CSS3 + Tailwind CSS (CDN)
- JavaScript ES6
- Apache Cordova

Backend

- NestJS (TypeScript) + PostgreSQL
- Express (Node.js) + PostgreSQL
- FastAPI (Python) + MongoDB

Ejecución con Docker Compose

1. Desde la raíz del repositorio:

```
docker compose up --build
```

2. Servicios expuestos (por defecto):
    - Frontend: `http://localhost:8080`

- API Pokémon (NestJS): `http://localhost:3000`
- API Países (Express): `http://localhost:4000`
- API Clima y Feriados (FastAPI): `http://localhost:8000`

3. Bases de datos:
    - NestJS y Express se conectan al contenedor PostgreSQL `postgres-db` definido en `docker-compose.yml` (se inicializa con `backend/express/database.sql`).
    - FastAPI se conecta a una base de datos MongoDB Atlas usando la cadena de conexión definida en la variable de entorno `MONGODB_URI` (con un valor por defecto embebido en `backend/fastapi/main.py`). El archivo `docker-compose.yml` ya incluye un `MONGODB_URI` de ejemplo para el entorno Docker.

Notas para Cordova / emulador

- En emulador Android: usar `http://10.0.2.2:<puerto>` como host para `BASE_URL_*`.
- En Docker: usar `http://host.docker.internal:<puerto>` como host para `BASE_URL_*`.

Cordova y APK Android

Esta sección resume cómo integrar el frontend de InfoMóvil en un proyecto Cordova (incluido en `cordova/`) y generar un APK Android en tu propio equipo.

Requisitos previos

- Node.js + npm instalados.

- Cordova CLI:

```
npm install -g cordova
```

- Java JDK y Android SDK (por ejemplo, mediante Android Studio), con variables de entorno configuradas según tu sistema operativo.

Preparar el proyecto Cordova (primera vez)

1. Desde la raíz del repositorio, sincroniza el frontend hacia Cordova:

```
npm run sync-www:cordova
```

   Esto copia `frontend/www` a `cordova/www`.
2. Entra a la carpeta Cordova:

```
cd cordova
```

3. Añadir la plataforma Android (solo la primera vez):

```
cordova platform add android
```

Sincronizar el frontend con Cordova (cada vez que cambies el frontend)

1. Volver a la raíz del repositorio (si estás dentro de `cordova/`):

```
cd ..
```

2. Ejecutar la sincronización:

```
npm run sync-www:cordova
```

   Esto limpiará y volverá a copiar el contenido de `frontend/www` dentro de `cordova/www`.

Generar el APK Android

1. Desde la carpeta `cordova/`:

```
cd cordova
cordova build android
```

2. Una vez completado el build, el APK de debug se encontrará (típicamente) en:

```
cordova/platforms/android/app/build/outputs/apk/debug/app-debug.apk
```

Ese archivo `app-debug.apk` es el que debes usar para pruebas en emulador/dispositivo y para la entrega del Taller (por ejemplo, subirlo al Campus).

Problemas comunes (APK)

- Conectividad HTTP bloqueada: ya se habilitó `usesCleartextTraffic` y `networkSecurityConfig` en `cordova/config.xml` para permitir HTTP en desarrollo.
- URLs de backend incorrectas: ajusta `cordova/www/js/config.js` según el entorno:
  - Emulador Android (AVD): `http://10.0.2.2:{puerto}`
  - Dispositivo físico: `http://{IP_PC_LAN}:{puerto}` (misma red WiFi)
- Whitelist/CSP: se añadió `cordova-plugin-whitelist` y una CSP permisiva en `index.html`.
- Backend no levantado: asegúrate de ejecutar primero los servicios con Docker Compose.

Comandos rápidos (Windows, cmd.exe):

```
docker compose up --build

cd cordova
cordova prepare android
cordova build android
```

Depuración útil (opcional)

- Overlay de depuración en pantalla: activar con

  ```
  localStorage.setItem('DEBUG_OVERLAY', '1')
  ```

  y desactivar con

  ```
  localStorage.removeItem('DEBUG_OVERLAY')
  ```

- Cambiar host de las APIs sin recompilar (usar IP LAN del PC):

```
localStorage.setItem('API_HOST', '192.168.X.Y')
```

Para volver al valor por defecto (10.0.2.2 en Cordova / localhost en navegador):

```
localStorage.removeItem('API_HOST')
```

Verificación rápida de la app

- En emulador (BlueStacks/AVD) con backend arriba, la app debe mostrar:
  - Tarjetas con lista de países más poblados (banderas visibles)
  - Pokémon destacados con sprites
  - Clima de ciudades de ejemplo y feriados de CL 2025

Cumplimiento de requisitos del taller

- Backend — 3 APIs independientes con base de datos:
  - NestJS (Pokémon) + PostgreSQL: `backend/nest/` (endpoints documentados en su README, datos servidos desde DB, seed automático).
  - Express (Países) + PostgreSQL: `backend/express/` (endpoints y esquema documentados, seed con `database.sql`/`countries-seed.json`).
  - FastAPI (Clima/Feriados) + MongoDB: `backend/fastapi/` (Swagger en `/docs`, seed automático en `startup`).
- Frontend: consume las 3 APIs, diseño responsivo Mobile First con Tailwind, JS puro en `frontend/www/js/*`.
- APK Android generado con Cordova (`cordova/`), probado en emulador; overlay de depuración desactivado por defecto.
- Documentación: este README + `docs/backend-contracts.md` + READMEs por API incluyen instrucciones de instalación/ejecución y detalles técnicos.