Programación B

UNIVERSIDAD FASTA | FACULTAD DE INGENIERÍA

TRABAJO PRÁCTICO

CENTRO DE MONITOREO URBANO

Messina, Matias - Fontanals, Facundo - Miglioli Flores, Joaquin - Henríquez, Diego

INTRODUCCIÓN +

Simulador de una Central de Monitoreo para una ciudad
-Control y muestra de estado de dispositivos que controle y muestre estados de los distintos dispositivos.
-Generacion de informes.

El objetivo lo logramos aplicando:
-Programación visual.
-Integrando técnicas de programación orientada a objetos
-Programación multihilo.
-Manejo de excepciones.
-Persistencia.
-Conexión a base de datos.

01

Semáforos

Ciclo de colores e intersecciones de calles que se manejan por un controlador.
Estado intermitente o estado de error.

02

Camaras de seguridad

Imágenes en tiempo real.
El operador puede detectar una anomalía, llamando a los servicios policiales, médicos o de bomberos.

03

Camaras de estacionamiento

Vehículos estacionados de manera indebida.
Se envia la informacion de la infraccion a la central de monitoreo

04

Radares de velocidad

Multas por exceso de velocidad conociendo el límite de velocidad a controlar.
Se envia la informacion de la infraccion a la central de monitoreo

CÓMO SE HIZO

Spring Boot

El proyecto tiene un backend en Java con Spring Boot que corre como un servidor local en localhost:8080

API REST

Ese backend expone una API REST y es quien maneja toda la lógica: ciclos de semáforos, detección de infracciones, guardado en base de datos y generación de informes.

index.html

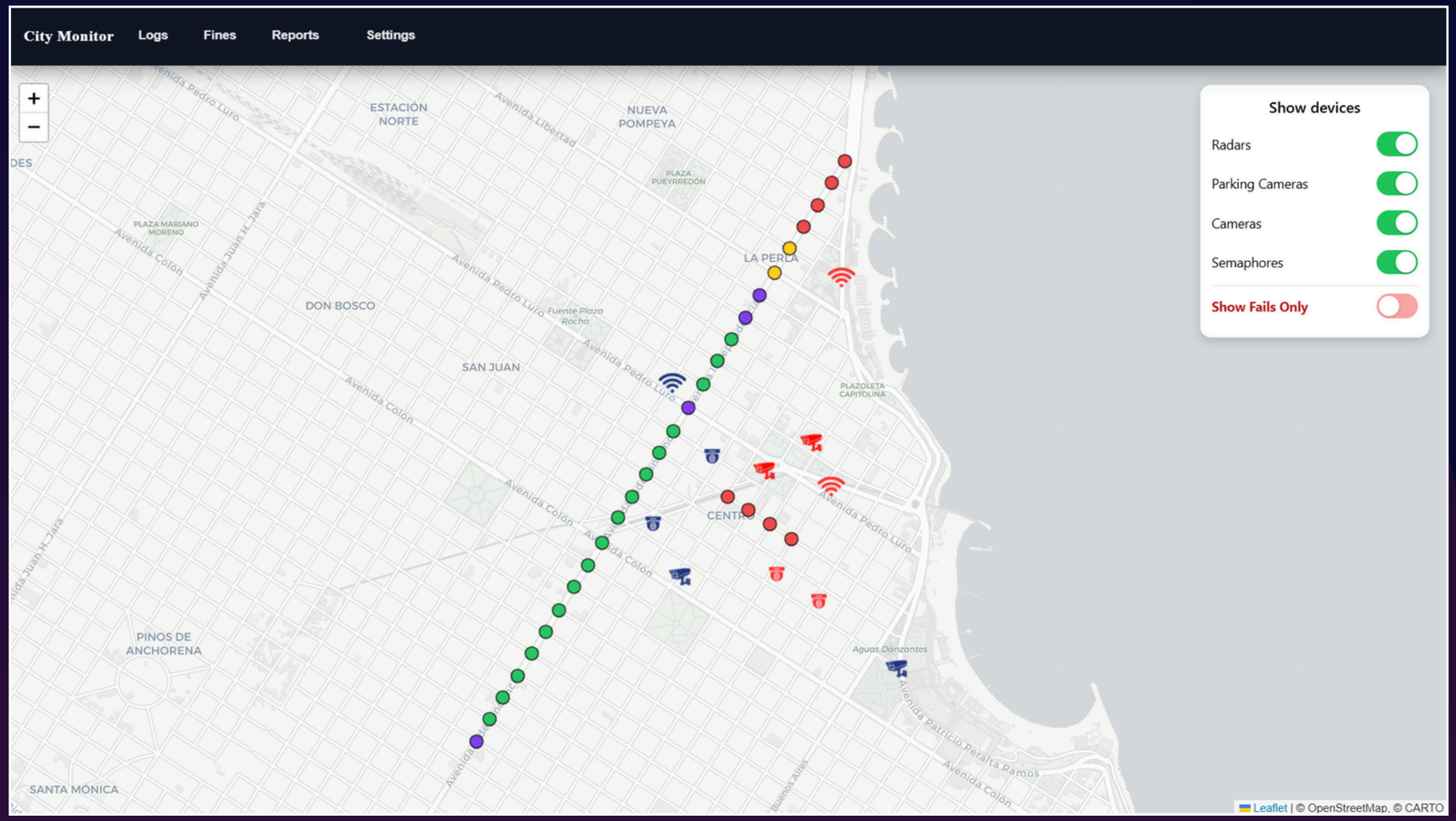
El frontend es un index.html con Leaflet que se abre en el navegador y le pide datos al backend con **fetch**. El backend responde JSON y el mapa se actualiza.

Programación B

EL SIMULADOR

El simulador muestra gráficamente el estado actual de cada uno de los dispositivos presentes , además de un menú amplio de opciones para el operador, como ver y filtrar las multas creadas o reportes de los dispositivos.

Permite también al operador poder ingresar marcas y modelos de autos desde la propia aplicación que serán guardados en la base de datos del programa, o autos particulares con patente y dueño.



Programación B

DESARROLLO DE SOFTWARE

01 Herramientas para programar

02 Estructura del código

Trabajar con Git fue fundamental para que todos pudiéramos programar sobre el mismo proyecto en nuestras casas y juntar cada parte para ejecutar el código.

Configuramos un archivo .gitignore, que le indica a Git qué archivos no debe seguir, como los de compilación o el estado de la simulación (state.bin). De esta manera evitamos conflictos innecesarios.

Trabajamos con un repositorio central en GitHub, usando ramas separadas para cada módulo. Por ejemplo, una rama para Reportes y otra para UI. Esto nos permitió desarrollar en paralelo y luego integrar los cambios de forma ordenada y segura

Programación B

ORGANIZACIÓN DEL CÓDIGO Y ARQUITECTURA

Nuestro código en src/main/java está separado por paquetes, donde cada uno tiene una responsabilidad única

01

Controllers

Recibe las peticiones HTTP que llegan desde el index.html y decide qué servicio debe encargarse de resolverlas.

02

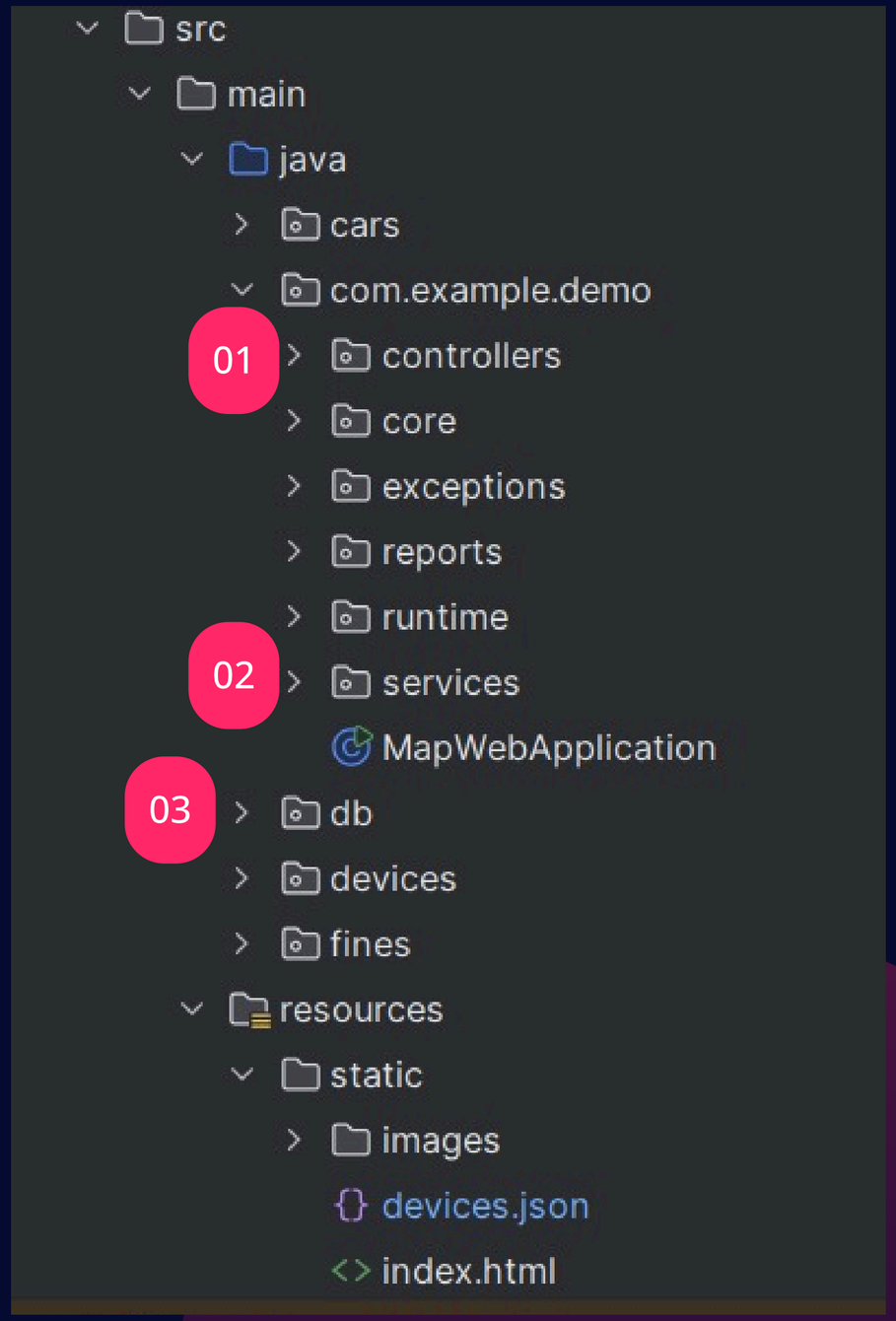
Services

Ocurren las operaciones complejas y se toman decisiones. Coordina el trabajo a realizar.

03

DB

Se conecta directamente con la base de datos PostgreSQL. Su tarea es transformar los datos de las tablas en objetos Java y viceversa



(index.html) → Controllers (API) → Services(Lógica) → DAOs (Datos)→ DB (PostgreSQL)

04

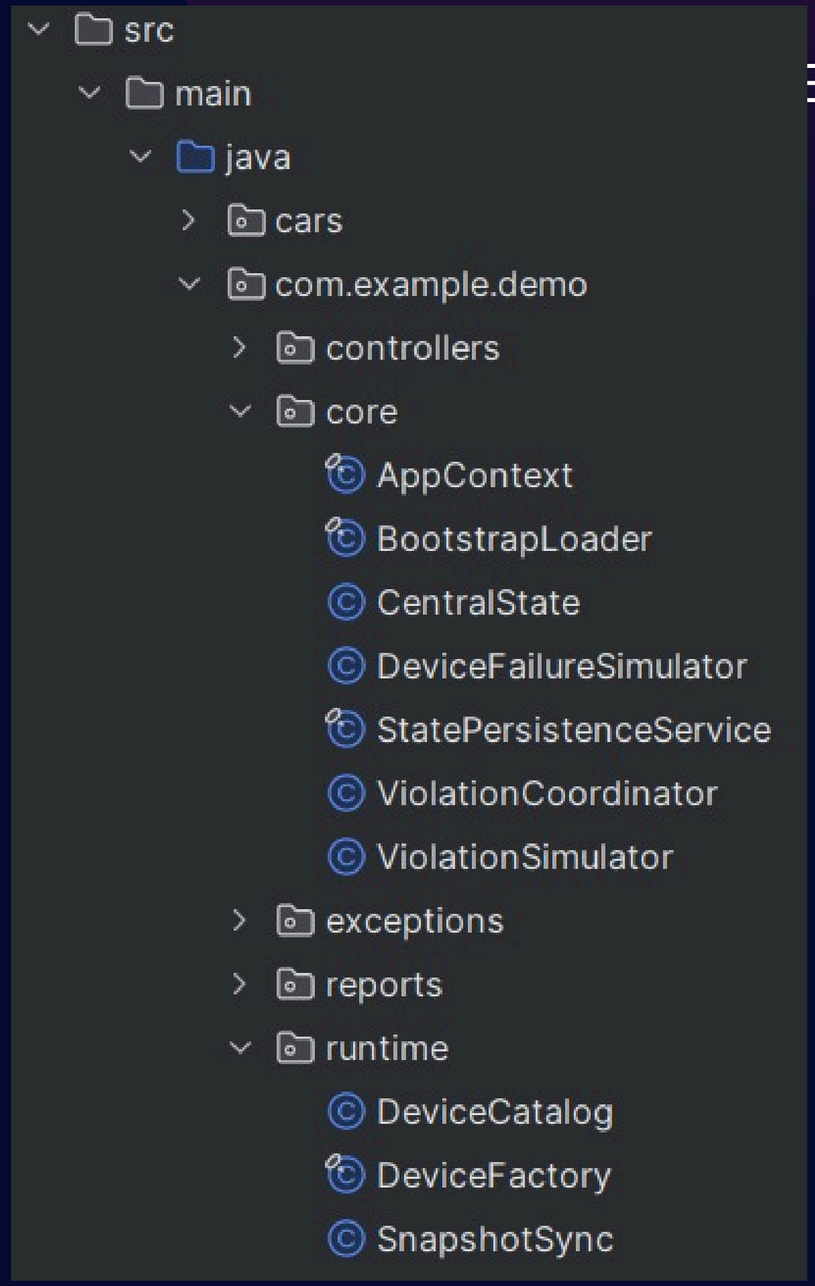
Core

Gestiona la lógica de ejecución en tiempo real, iniciando los hilos que simulan el comportamiento y las fallas de los dispositivos de forma automática.

05

Runtime

Son clases utilitarias que conectan la simulación (core) con los objetos de dominio



ABSTRACCIÓN

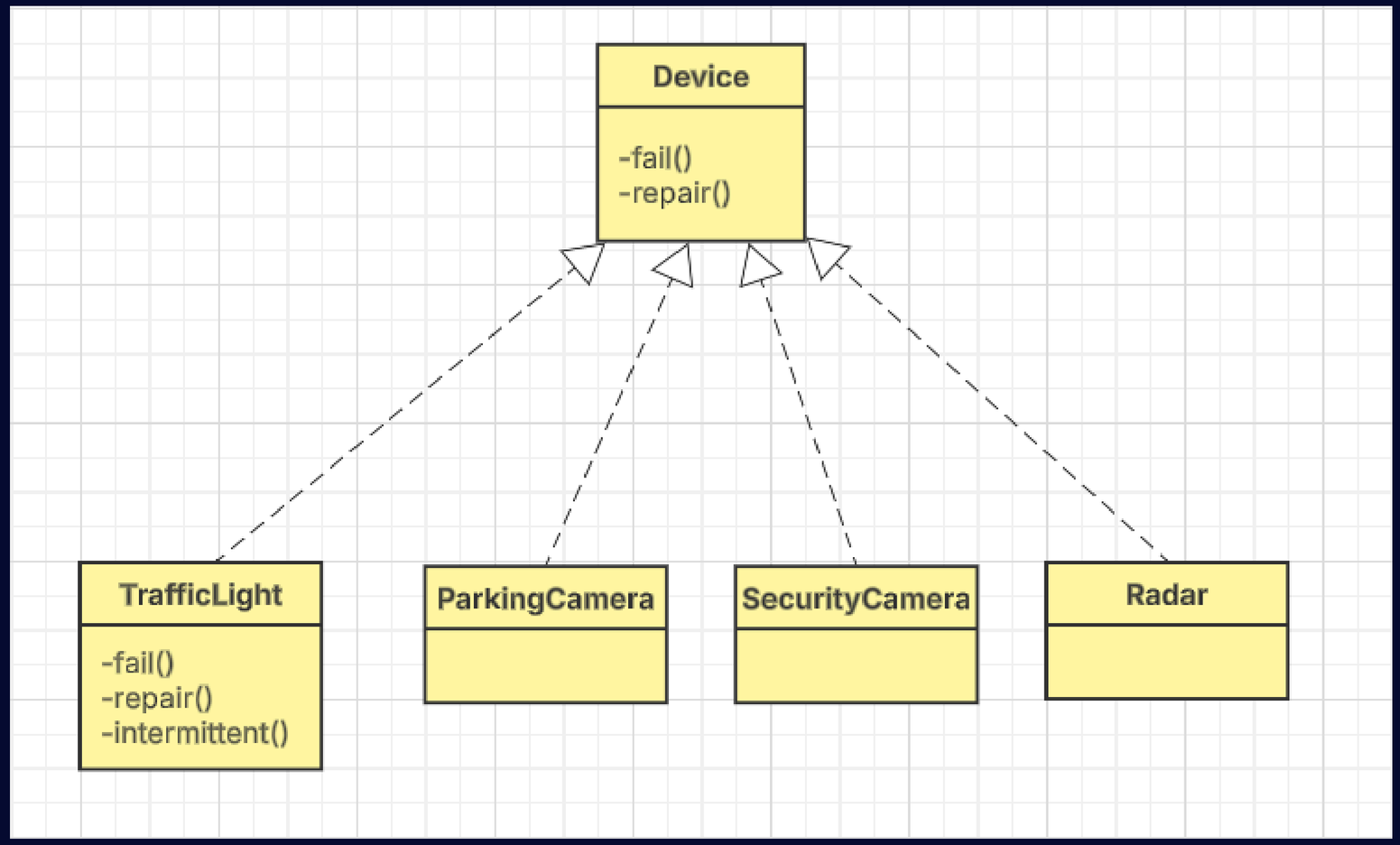
Se analizó el problema y se indentificaron las distintas clases de objetos junto con los estados que debería tener y las acciones a realizar dentro de la solución.

ENCAPSULAMIENTO

Aplicamos el encapsulamiento para proteger el estado interno de nuestros objetos. Esto evita que los datos sean modificados accidentalmente y nos asegura que se mantenga la consistencia.

```
private String deviceId;  3 usages
private String address;  3 usages
private DeviceStatus status = DeviceStatus.UNKNOWN;
```

Herencia



Usamos la herencia para reutilizar código y crear clases hijas a partir de una clase madre, cumpliendo con el requisito de la consigna.

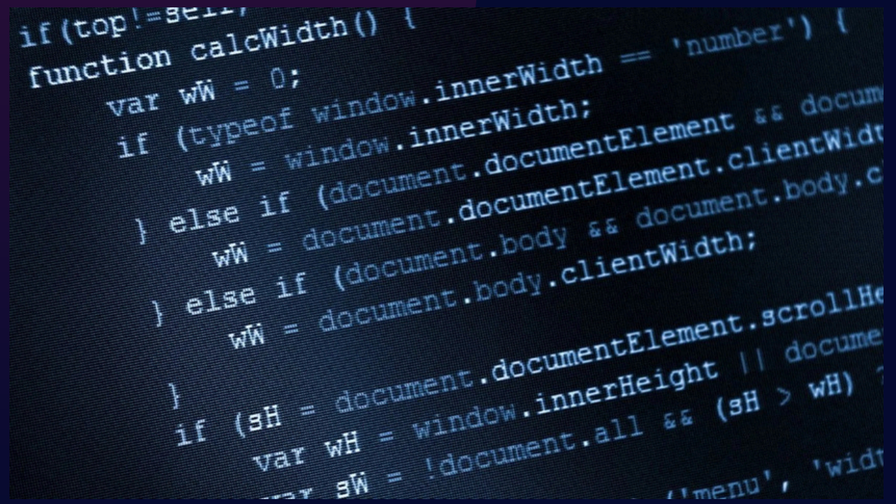
POLIMORFISMO

Usamos polimorfismo (que significa "muchas formas") para permitir que objetos de diferentes clases respondan de manera distinta a la misma llamada.

```
@Override
```

ABOUT PRESENTATION AND COMMUNICATION

- How did you prepare for this presentation in English?
- Why do you think English is important for programming students?
- How did this experience improve your English skills?



Programación B

SCRUM METHODOLOGY

- How did you implement the Scrum methodology in your project?
- What were the key tasks in your project backlog?

- How did you manage your sprints and track your progress?
- How did you document your Scrum process in English?