CSS Phase 3 – UrbanWheels
Relatório Técnico

Guilherme Soares (fc62372), Vitória Correia (fc62211), Duarte Soares (fc62371)

15/12/2025

1 Contexto

A **ConcreteCast Solutions (CCS)** implementou o **UrbanWheels**, um sistema de bicicletas partilhadas. O objetivo desta fase foi construir interfaces gráficas (web com Thymeleaf e desktop com JavaFX) e integrar o sistema com o **WeatherWise** para previsões meteorológicas.

- Fases 1 e 2: backend funcional com API REST e persistência de dados.
- Fase 3: interfaces gráficas + integração com WeatherWise + vídeo de demonstração.

2 Decisões Técnicas

2.1 Docker

- **docker-compose.yml** atualizado para construir o container PostgreSQL a partir de um Dockerfile customizado:

```
pgserver:
  build:
    context: .
    dockerfile: postgres.dockerfile
  container_name: pgserver
  volumes:
    - postgres-data:/var/lib/postgresql/data
```

- *Motivo: garantir consistência de dados entre reinícios de container.*

2.2 Spring Boot

- **application.properties**: alterado `spring.jpa.hibernate.ddl-auto=validate` para validar a consistência do esquema de dados ao iniciar a aplicação.

2.3 DTOs e Integração WeatherWise

- `StationDTO` modificado para incluir previsão meteorológica:

```
Long id;
String name;
double lat;
double lon;
int maxDocks;
List<BikeDTO> bikes;
WeatherConditionDTO weather; // novo par metro
```

- Endpoint REST no `WeatherWiseController` para obter previsão:

```
@GetMapping("/forecast")
public ResponseEntity<?> getForecast(
        @RequestParam double lat,
        @RequestParam double lon,
        @RequestParam String date) {
    try {
        return ResponseEntity.ok(weatherService.getForecast(lat, lon, date));
```

```
    } catch (RuntimeException e) {
        return ResponseEntity.status(404).body(e.getMessage());
    }
}
```

- *Motivo: permitir associar a previsão meteorológica a cada estação.*

2.4 Interface Desktop – JavaFX

Para o cliente desktop, a aplicação foi estruturada em múltiplos `Controllers` associados a FXML. As principais decisões técnicas incluem:

- **Organização do layout:** Cada caso de uso (listar estações, reservar bicicleta, cancelar reserva, levantar/entregar bicicleta, consultar detalhes do utilizador) possui um `VBox` ou `HBox` dedicado, com espaçamento e alinhamento central.
- **Componentes utilizados:** `Button` para ações, `ComboBox` para seleção de estações e bicicletas, `Label` para status e mensagens de erro.
- **Lista de estações:** Ao clicar numa estação, abre uma nova cena mostrando todas as bicicletas associadas a essa estação, incluindo estado e previsão meteorológica.
- **Bindings e propriedades:** Utilização de `Property` (`StringProperty`, `ObjectProperty`) nos modelos (`Bike`) para permitir ligação reativa entre dados e interface. Por exemplo:
  - `Bike.modelProperty()` e `Bike.stateProperty()` para exibir nome e estado.
  - `Bike.weatherProperty()` para associar informação meteorológica e calcular cores.
- **Lógica de cores:** A classe `Bike` contém método `getColor()` que determina a cor a exibir com base em:
  - Condição meteorológica (`WeatherConditionDTO.condition`)
  - Temperaturas mínima e máxima (`minTemp`, `maxTemp`)

  Isto permite feedback visual imediato sobre condições adversas ou favoráveis.
- **Integração com WeatherWise:**
  - Endpoint REST chamado via cliente HTTP para obter previsão para a estação selecionada.
  - Dados retornados populam `WeatherConditionDTO` e são associados a cada bicicleta.
  - Informações meteorológicas usadas em labels ou na coloração das bicicletas.
- **Gestão de estados:** Estados das bicicletas (disponível, reservada, em uso) armazenados em `StateDTO` e ligados a cada `Bike` para atualização dinâmica da interface.

2.5 Interface Web – Thymeleaf

Para o cliente web, as decisões técnicas foram:

- **Cabeçalhos e menus:** Implementados com fragmentos reutilizáveis (`th:fragment`) para consistência e facilidade de manutenção.
- **Lista de estações:** As estações são exibidas em tabela ou lista. Cada estação é clicável e, ao ser clicada, mostra todas as bicicletas associadas a essa estação, incluindo estado e previsão meteorológica.
- **Filtros e estados:** Implementados filtros para estados de bicicletas (disponível, em uso, reservada) usando `th:if` e dropdowns.
- **Integração REST:** Chamadas HTTP ao backend UrbanWheels para operações CRUD e consumo da API WeatherWise para exibir previsão do tempo na página da estação.

3 Considerações Finais

Todas as alterações mantêm a arquitetura em camadas (Controller, Service, Repository) e comunicação via API REST. O ambiente completo é contentorizado via Docker e reproduzível conforme exigido. Vídeo de demonstração preparado para mostrar interações e integração com WeatherWise.