Ciências ULisboa | Informática

Construção de Sistemas de Software
Engenharia Informática
Semestre 2025/2026

**Projeto Prático #3:** O projeto deve ser desenvolvido por grupos de *no máximo* três alunos. É estritamente proibida a partilha de código entre alunos de grupos diferentes.

**Plágio:** Além de inspeção manual, será também executado um software de deteção de plágio no código-fonte do projeto. *Todos* os alunos que submeterem código obtido através de plágio terão os seus projetos anulados.

**Uso de Inteligência Artificial:** É permitido utilizar ferramentas de IA como o ChatGPT, Copilot, DeepSeek e similares. No entanto, todos os membros do grupo devem ser capazes de compreender e *explicar* o projeto *na sua totalidade* aos docentes quando pedido. Se houver código no projeto que o grupo não consiga explicar, o projeto será considerado como *plágio*.

**Submissão:** A data de submissão do projeto é **15/12/2025** às 23:59. A descrição do processo de submissão pode ser encontrada no enunciado.

# UrbanWheels: Gestão de Bicicletas Partilhadas

## 1 Contextualização

A empresa **ConcreteCast Solutions (CCS**[1]**)** pretende implementar um sistema de bicicletas partilhadas, o **UrbanWheels**, para promover a mobilidade sustentável. Foi-vos incumbida a tarefa de desenvolver o backend que irá gerir toda a operação, desde o registo de utilizadores até à gestão da frota de bicicletas e das suas viagens.

Nesta terceira e última fase do projeto, o foco será a construção de interfaces gráficas para interagir com o sistema **UrbanWheels** e a integração com um sistema externo de meteorologia, o **WeatherWise**. Assumimos que as fases 1 e 2 foram concluídas, resultando num backend funcional com uma API REST bem definida e persistência de dados.

## 2 Casos de Uso

Nesta fase do projeto, o vosso trabalho será dividido em três aspectos: a implementação de uma interface web com Thymeleaf, uma interface desktop com JavaFX e, finalmente, a atualização do **WeatherWise** e a inclusão de uma API REST, permitindo a integração entre os sistemas.

### 2.1 Interface Web (Thymeleaf)

Deverá ser criada uma interface web para as operações de gestão, tipicamente realizadas por administradores. Os seguintes casos de uso devem ser implementados usando **Spring Boot com Thymeleaf**:

A. Realizar login (Nota: vamos fazer um mock. Qualquer palavra passe será aceite mas o utilizador deve ser válido).

B. Registar uma nova estação.

[1] Entidade fictícia

C. Listar todas as estações.

D. Obter os detalhes de uma estação específica (e.g., lista de bicicletas).

E. Adicionar uma nova bicicleta à frota (associar a uma estação inicial).

F. Listar todas as bicicletas em todas as estações (pode-se filtrar pelo seu estado: `DISPONIVEL`, `EM_USO`, etc.).

G. Registar uma nova operação de manutenção para uma bicicleta.

H. Registar um novo utilizador.

I. Listar todos os utilizadores.

### 2.2 Interface Desktop (JavaFX)

Para as operações do dia-a-dia dos clientes, deverá ser desenvolvida uma *rich client application*. Os seguintes casos de uso devem ser implementados em **JavaFX**:

J. Realizar login (Nota: vamos fazer um mock. Qualquer palavra passe será aceite mas o utilizador deve ser válido).

K. Listar todas as estações.

L. Obter os detalhes de uma estação específica (e.g., lista de bicicletas)

M. Realizar uma reserva de bicicleta. Modificando o estado da bicicleta para `RESERVADA`.

N. Cancelar uma reserva de bicicleta. Modificando o estado da bicicleta para `DISPONIVEL`.

O. Levantar uma bicicleta.

P. Entregar uma bicicleta.

Q. Obter os detalhes do utilizador (e.g., histórico de viagens).

### 2.3 Integração de Sistemas (REST API)

De forma a integrar os dois projetos, o sistema UrbanWheels deverá consultar a previsão do tempo antes de uma viagem.

I. **Atualização do WeatherWise:** No projeto WeatherWise, incluir um caso de uso que permita previsões meteorológicas baseadas em localização (latitude e longitude) e data e torná-lo publico através de uma API REST. Isso deve incluir:

- Modificações ao modelo de dados para suportar consultas por localização e data.
- Um endpoint REST que aceite parâmetros de localização e data.

Opcionalmente, podem utilizar uma API pública de meteorologia (e.g., OpenWeatherMap, WeatherAPI) para obter dados reais.

J. **Novo Caso de Uso no UrbanWheels:** Ao reservar uma bicicleta (caso de uso M), a aplicação JavaFX deve:

(a) Chamar a nova API REST do WeatherWise para obter a previsão do tempo para a localização da estação de origem.

(b) Exibir a previsão (e.g., temperatura e descrição do tempo) ao utilizador.

## 3 Requisitos Não Funcionais

O projeto deve cumprir os seguintes requisitos:

- As interfaces gráficas (Thymeleaf e JavaFX) devem comunicar com o backend do UrbanWheels através da API REST desenvolvida na fase 2.
- A nova API REST no projeto WeatherWise deve ser implementada em Spring Boot, seguindo as mesmas boas práticas do projeto UrbanWheels. Não é necessário re-implementar o projecto WeatherWise. Devem apenas incluir a camada de apresentação com o REST API.
- O projeto todo deve ser contentorizado usando Docker, para facilitar a sua execução e integração.
- Todas as novas funcionalidades devem ser acompanhadas de um vídeo de demonstração. **A ausência do vídeo para um determinado caso de uso resultará na anulação da nota correspondente a esse caso de uso.**
- A arquitetura deve manter-se dividida em camadas (Controller, Service, Repository) em ambos os projetos.
- O repositório deve aceitar apenas código com o nível de qualidade aceite pela equipa (ex.: *pre-commit*, testes).
- O projeto deve ser contentorizado usando Docker.
- O controlo da participação de cada membro do grupo será feito via atividade no repositório git, com base no número e qualidade dos *commits*.
- As decisões técnicas devem ser justificadas num relatório técnico.

## 4 Tarefas da Fase 3

As tarefas da fase 3 presumem que as tarefas da fase 1 e 2 foram realizadas com sucesso. Caso alguma das tarefas ainda esteja pendente, é necessária a sua conclusão antes de iniciar a fase 3.

- Implementar a interface web com Thymeleaf.
- Implementar a interface desktop com JavaFX.
- No projeto WeatherWise, atualizar o projeto e implementar o endpoint REST.
- No projeto UrbanWheels, implementar a lógica de cliente HTTP para consumir a API do WeatherWise e integrar no caso de uso M.
- Produzir um (ou dois) vídeo(s) que demonstre o funcionamento de **todos** os casos de uso implementados. O vídeo deve mostrar a interação com as interfaces gráficas e a integração entre os sistemas.
- Se não for possível fazer o upload do vídeo para o repositório, devem atualizar o ficheiro 'README.md' para incluir um link para o vídeo de demonstração (e.g., YouTube, Google Drive).

# 5 Como Entregar

Para entregar o trabalho, basta criar uma tag chamada *fase3* e enviá-la para o repositório. O repositório deverá conter o código-fonte de ambos os projetos (UrbanWheels e WeatherWise), os ficheiros Docker necessários, e o 'README.md' atualizado com as instruções de execução e o link para o vídeo.

```
git tag fase3
git push origin fase3
```

Certifiquem-se de que o vosso projeto está acessível à conta CSS000 na tag fase3. É fundamental que o ambiente de execução de **todos os componentes** (backend UrbanWheels, backend WeatherWise, e clientes Thymeleaf e JavaFX) possa ser reproduzido pela equipa docente. Falhas na execução de qualquer parte do projeto resultarão em **grave penalização**.

# 6 Critérios de Avaliação

Esta entrega corresponde a 40% da nota final do projeto. Os critérios de avaliação são os seguintes:

## 6.1 Processo de Desenvolvimento

- Granularidade e frequência dos *commits*. Todos os membros do grupo devem contribuir de forma equitativa. *Disparidades significativas serão penalizadas.*
- Qualidade das mensagens de *commit*.
- Qualidade do código presente no repositório.
- Funcionalidade dos casos de uso.

## 6.2 Interfaces Gráficas (Thymeleaf e JavaFX)

- Implementação correta e completa dos casos de uso designados para cada tecnologia.
- Qualidade da interface e da experiência de utilizador (clareza, usabilidade e robustez).
- Comunicação correta com a API REST do backend.

## 6.3 Integração de Sistemas

- Implementação correta e funcional da API REST no projeto WeatherWise.
- Integração bem-sucedida do sistema UrbanWheels com o WeatherWise no caso de uso de reservar a bicicleta.

## 6.4 Vídeo de Demonstração

- Clareza e objetividade na apresentação das funcionalidades.
- Demonstração completa de todos os casos de uso implementados.
- Qualidade geral do vídeo.

### 6.5 Relatório

- Descrição clara das decisões técnicas tomadas.
- O relatório deve ser conciso e bem estruturado. Máximo de 2 páginas.

---

**Nota Final:** Em caso de dúvidas, utilizem o fórum de discussão no Moodle. Bom trabalho com o desenvolvimento do *UrbanWheels*!