# Engenharia de Aplicações

*Licenciatura em Engenharia Informática*

## Pensar "Object Oriented" na abordagem ao projeto DEMO_ORM e Implementação da camada de Persistência em JPA

### Objetivos da aula PL da semana 2

- Aplicar boas práticas de desenvolvimento de software
- Compreender e aplicar o processo ORM – processo de conversão bidirecional entre objetos e tabelas de uma base de dados relacional
- Compreender e aplicar o padrão DDD Repository
- Compreender e aplicar os conceitos associados a JPA:
    - Entity and annotations ,
    - EntityManager,
    - Persistence Unit – configura o EntityManager
        - META-INF
            - persistence.xml – define os dados de conexão à BD: driver, user, password
- Aplicar os conceitos no desenvolvimento de uma aplicação (DEMO_ORM) que envolve persistência de informação usando JPA2 implementada por EclipseLink acedendo a uma base de dados relacional H2,
- Rever conceitos associados à análise, conceção e implementação de software orientado a objetos
- Aceder a informação na base de dados
- Compreender e aplicar o padrão DDD Repository
- Compreender e aplicar os conceitos associados a JPA:
    - Entity and annotations ,
    - EntityManager,
    - Persistence Unit – configura o EntityManager
        - META-INF
            - persistence.xml – define os dados de conexão à BD: driver, user, password

**Pensar OO - "Object Oriented" - Breve Revisão**

1. **Importância da descrição dos Casos de Uso**

   O desenvolvimento do software vai ser dirigido pelos Casos de Uso, por isso na primeira fase fazer a descrição dos Casos de Uso é relevante. O modelo de casos de uso não é uma especificidade do desenvolvimento orientado a objetos.

   Considere a descrição em formato breve do **Caso de Uso "Registar Grupo Automóvel"**

   *O utilizador acede ao sistema para registar um novo grupo automóvel iniciando-se o processo de "Registar Grupo Automóvel".*
   *O sistema solicita os dados do novo grupo automóvel que são: nome, nº de portas e classe. Relativamente à classe o sistema apresenta a lista de classes para seleção.*
   *O utilizador submete os dados para serem registados pelo sistema.*

2. **Análise OO –** Identificar as entidades do domínio e suas relações – Modelo de Domínio (MD)

   A fase prévia de especificação dos requisitos é relevante para a elaboração do MD. O MD é um artefacto muito relevante para o desenvolvimento de software orientado a objetos, é inspirador para a criação das classes do domínio.

   *Considere o MD já apresentado*

3. **Design OO –** Conceber a solução de software em termos de objetos colaborantes com responsabilidades. Organizar adequadamente as peças de software (classes) a criar.

   **Realização de um caso de uso(UC)** – Pensar na concretização do Sistema para realizar este UC. Elaborar Diagrama de sequência do UC e Diagrama de classes.

   UC Registar Grupo Automóvel – Elabore o Diagrama de Sequência

   **A arquitetura lógica do Sistema -** organizada em 4 camadas (packages)

   - Apresentacao
   - Aplicacao
   - Dominio
   - Persistencia

   Definir a arquitetura lógica do projeto DEMO_ORM. Qual a vantagem deste tipo de arquitetura.

   Algumas considerações gerais:

   - Atribuição de responsabilidades devem estar de acordo com princípios **Information Expert**, **Information Hiding** e **"Tell, don't ask"**

4. **Aceder a informação na Base de Dados - Camada de persistência**
   a. Utilização do padrão Repository ( com ou sem RepositoryFactory )
   b. Implementar a camada de persistência
5. **Codificação OO com implementação de Testes Unitários**

Implemente em Java a funcionalidade e os respetivos Testes Unitários

---

## Preparação do Projeto para Implementação da camada de Persistência em JPA

*ATENÇÃO: Para a realização das seguintes tarefas de modo colaborativo deve:*

- *Consultar o documento no moodle " EAPLI_PL_modo_funcionamento"*
- *Para criar a lista de "issues" associadas às seguintes tarefas deve consultar a proposta definida na parte final deste documente fazendo os ajustes considerados adequados inclusive numeração*

---

1. Analisar o Diagrama de Sequência do caso de uso "Registar Grupo Automóvel" *(consultar RentACar-UC RegistarGrupoAutomovel.png)*
    a. Uso do padrão *Repository* para fazer a intermediação entre a camada da aplicação e a persistência de dados

2. Preparar o projeto com as bibliotecas necessárias para fazer persistência usando JPA2 com EclipseLink para ligar a uma base de dados H2.

Acrescentar no file pom.xml

```
<dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>eclipselink</artifactId>
        <version>2.5.2</version>
</dependency>
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>1.4.194</version>
</dependency>
<dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>org.eclipse.persistence.jpa.modelgen.processor</artifactId>
        <version>2.5.2</version>
        <scope>provided</scope>
</dependency>
```

3. Criar a Persistence Unit necessária para criar um gestor de persistência Entity Manager encarregado de persistir qualquer alteração do estado dos objetos para a base de dados.

    a. Definir a conexão com a base de dados ( driver, localização, user e password)
    A base de dados a criar, de nome **rentacar**, será localizada na pasta **DB** do projeto.

---

*Consultar o documento* ***CriarAplicaçãoJPA-EclipseLink-H2*****.docx** mas
***Atenção:***

- *Este documento cria uma aplicação Java que usa o builder Ant e não o Maven. Com Ant têm que se incluir explicitamente as bibliotecas a necessárias ao projeto, com Maven declara-se essas bibliotecas como dependências, no ficheiro pom.xml. Por isso não deve seguir os pontos 1 e 2 do documento.*
- *Todo o documento deve ser analisado pois não se refere exatamente ao projeto DEMO_ORM. Ao criar a Persistence Unit o nome sugerido pode ser alterado, no código disponibilizado é DEMO_ORMPU.*
- *"H2 database engine" pode ser usado em modo embutido ("embedded") na aplicação Java, ou em modo servidor. Neste documento está a ser usado embutido na aplicação. Para mais informação consultar*
  *http://www.h2database.com/html/main.html*

---

4. Fazer as anotações JPA para a classe do domínio **GrupoAutomovel** tornando-a numa entidade a persistir na base de dados. Garantir a existência de um atributo Id e um construtor sem parâmetros.
   Atualizar o ficheiro de configuração "persistence.xml" registando as "Entity Classes" – **GrupoAutomovel**

5. Analisar e completar a implementação da funcionalidade **Registar um novo Grupo Automóvel**
   a. Analisar a interface GrupoAutomóvelRepositório e implementar a classe GrupoAutomóvelRepositórioJPAImpl (*consultar GARepo.zip*) de modo a permitir guardar e aceder a informação na base de dados

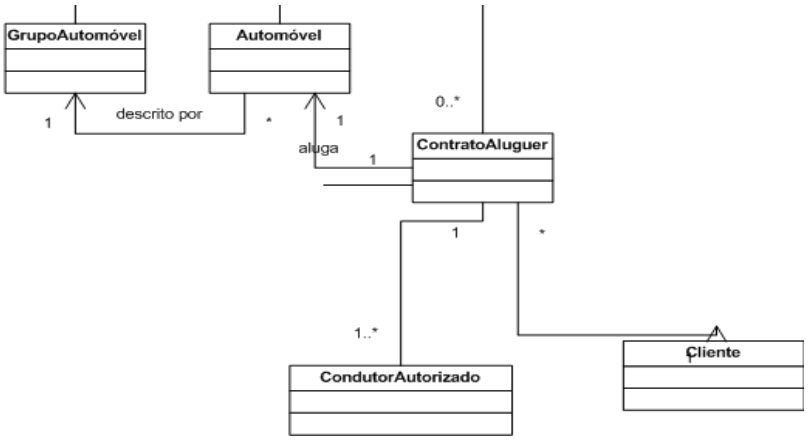
5. No separador Services do Netbeans aceda à conexão com a base de dados e confirme as atualizações produzidas. (*consultar documento CriarLigacaoNetBeans_BaseDadosH2.docx* )

6. Analisar e completar a implementação das funcionalidades **Listar todos os Grupos Automóveis** e **Procurar um Grupo Automóvel por id.**

7. **Refactoring** do código para evitar duplicação
   **a.** Código repetido na implementação das classes GrupoAutomovelRepositorioJPAImpl e AutomovelRepositórioJPAImpl é um "code smell" que aconselha fazer o Refactoring do código. Utilize a classe utilitária **JpaRepository.java** também disponível no moodle associada a este exercício ORM. (*consulte JpaRepository.zip*)
   **b.** Analisar cuidadosamente a classe JpaRepository.java
   **c.** Fazer o refactoring do código desenvolvido passando a utilizar a classe genérica **JpaRepository.java** de modo a eliminar a duplicação de código.

## FUNCIONALIDADES EXTRA

**RegistarContratoAluguer ->CondutorAutorizado  associação one-to-many**




1. Adicionar ao projeto uma nova entidade cliente que pode ser do tipo cliente empresa ou cliente particular O contrato aluguer está associado com cliente numa relação unidirecional de “many to one”. Utilize herança que deve ser implementada com uma única tabela
2. Registar Contrato Aluguer de um Automóvel já existente na BD mas cujos condutores autorizados não existem e serão registados na altura do registo do contrato.
3. Listar todos os Contratos Aluguer
4. Procurar Contratos Aluguer por id
5. Alterar Registar Contrato Aluguer de modo a incluir o pagamento. Para tal adicionar ao projeto uma nova entidade **Pagamento** que pode ser do tipo **Pagamento a Crédito** ou **Pagamento por Cheque.**  Utilize herança que deve ser implementada com uma tabela por classe. O Contrato Aluguer está associado com Pagamento numa relação unidirecional de