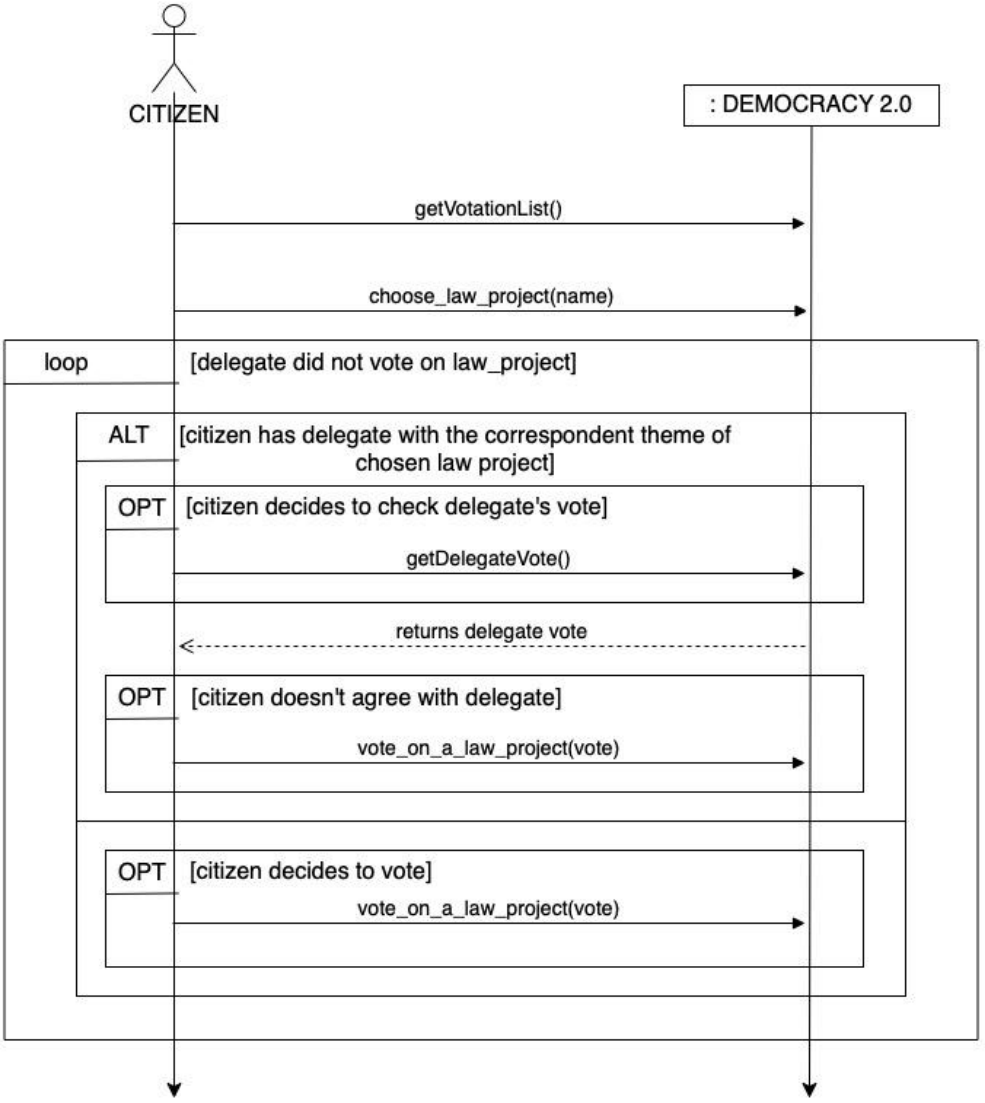
Relatório CSS - 2023
Democracia2

Ana Luís - Fc53563
Ana Teixeira - Fc56336
Vasco Barros - Fc54986

SSD - CASO DE USO J



Modelo de Domínio

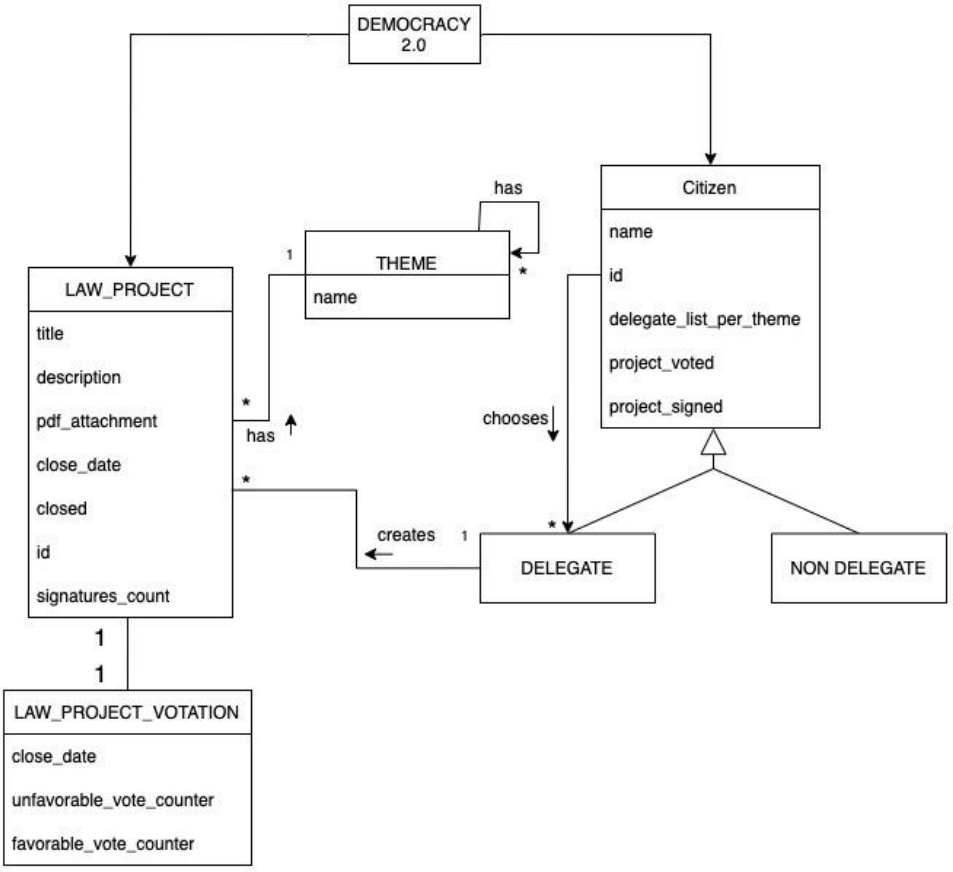
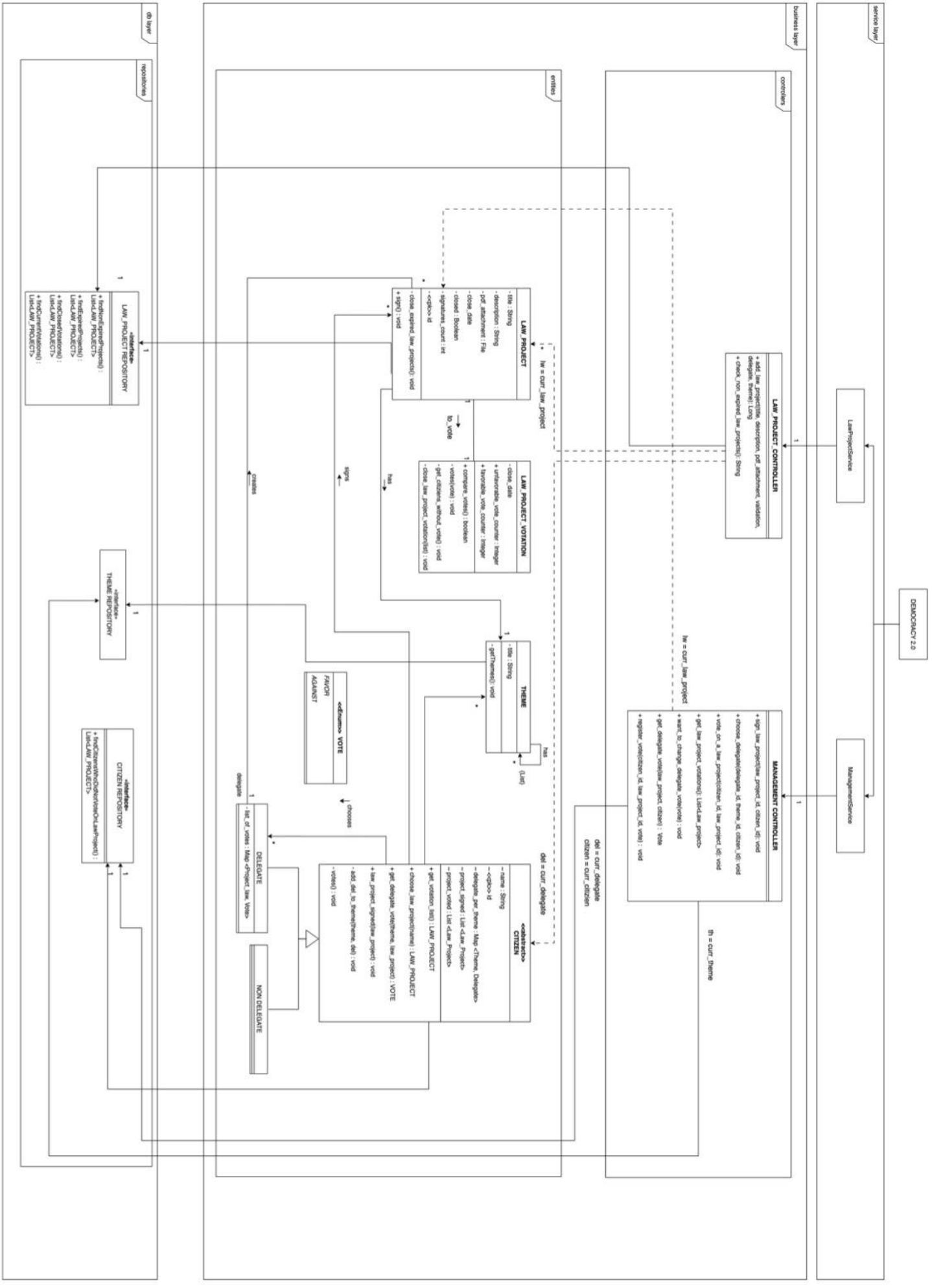
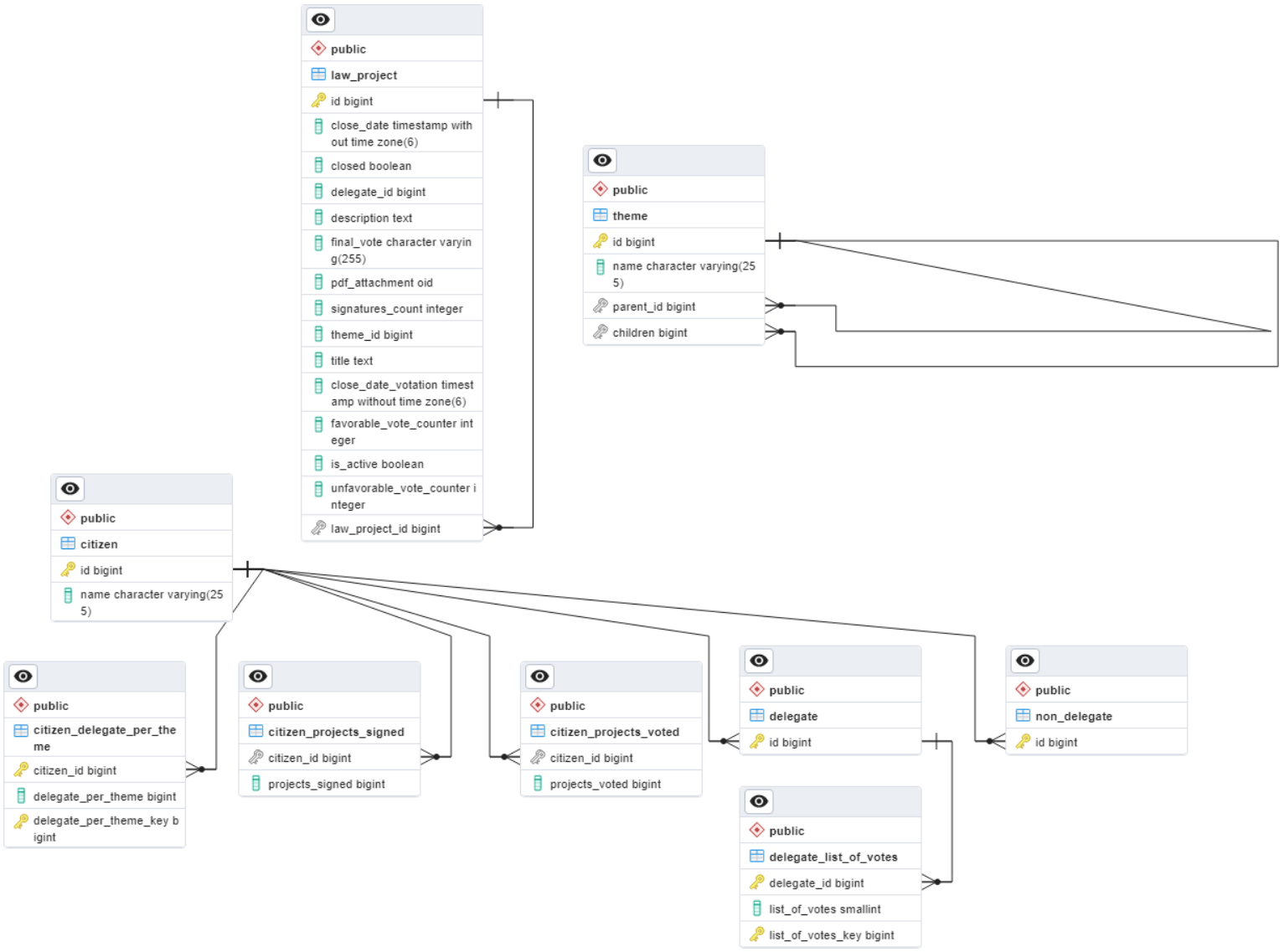


DIAGRAMA DE CLASSES



ERD produzido pela Base de Dados (Opcional)



Anotações JPA

Classe Citizen

```java
package pt.ul.fc.css.project.entities;

import static jakarta.persistence.InheritanceType.JOINED;

@Entity
@Inheritance(strategy = JOINED)
/** Class to creat objects of Citizens */
public abstract class Citizen {

  @Id
  @GeneratedValue(strategy = GenerationType.SEQUENCE)
  protected Long id; // primary key

  @NonNull protected String name;

  /**Map<THEME_ID, DELEGATE_ID> to keep the delegates of the citizen*/
  @ElementCollection(fetch = FetchType.EAGER)
  protected Map<Long, Long> delegate_per_theme = new HashMap<>();

  /**List<PROJECT_ID> list of projects voted*/
  @ElementCollection(fetch = FetchType.EAGER)
  protected List<Long> projects_voted = new ArrayList<>();

  /**List<PROJECT_ID> list of projects signed*/
  @ElementCollection(fetch = FetchType.EAGER)
  protected List<Long> projects_signed = new ArrayList<>();
```

Anotações da classe Citizen:

- @Entity - Define Citizen como uma classe que pode ser persistida na Base de Dados.
- @Inheritance(strategy = JOINED) - As subclasses de Citizen (classe Delegate e NonDelegate) terão as suas próprias tabelas na Base de Dados.
- @Id - Define id como a chave primária da tabela Citizen.
- @GeneratedValue(strategy = GenerationType.SEQUENCE) - Indica que os valores do id serão gerados automática e sequencialmente.
- @ElementCollection(fetch = FetchType.EAGER) - Mapeia as coleções projects_voted e projects_signed que é um mapeamento de Long. Esta anotação inclui fetch = FetchType.EAGER de modo a indicar que os elementos são recuperados imediatamente mesmo que não estejam a ser geridos.

Classe Delegate

```java
package pt.ul.fc.css.project.entities;

import jakarta.persistence.ElementCollection;

/**
 * The Delegate class represents a citizen that can create law projects and have a public vote.
 */
@Entity
public class Delegate extends Citizen {

  @ElementCollection(fetch = FetchType.EAGER)
  private Map<Long, Vote> list_of_votes = new HashMap<>();
```

Anotações da classe Delegate:

- @Entity - Define Delegate como uma classe que pode ser persistida na Base de Dados.
- @ElementCollection(fetch = FetchType.EAGER) - Mapeia a coleção list_of_votes que é um mapeamento de Long e Vote. Esta anotação inclui fetch = FetchType.EAGER de modo a indicar que os elementos são recuperados imediatamente mesmo que não estejam a ser geridos.

Classe NonDelegate

```java
package pt.ul.fc.css.project.entities;

import jakarta.persistence.Entity;

@Entity
/** Class that extends Citizens and creates Non delegates */
public class NonDelegate extends Citizen {

  public NonDelegate() {}

  public NonDelegate(@NonNull String name) {
    super(name);
  }
```

Anotações da classe NonDelegate:

- @Entity - Define NonDelegate como uma classe que pode ser persistida na Base de Dados.
- @NonNull - Indica que o parâmetro *name* não pode ser nulo.

Classe LawProject

```java
package pt.ul.fc.css.project.entities;

import jakarta.persistence.Column;

/** The LawProject class represents a law project that can be created by a delegate. */
@Entity
public class LawProject {

  @Id
  @GeneratedValue(strategy = GenerationType.SEQUENCE)
  private Long id;

  @Transient private ScheduledExecutorService executor;

  @Column(columnDefinition = "TEXT")
  private String title;

  @Column(columnDefinition = "TEXT")
  private String description;

  @Lob private byte[] pdf_attachment;

  @Column(name = "close_date")
  @Convert(converter = LocalDateTimeConverter.class)
  private LocalDateTime close_date;

  @Column(name = "closed")
  private boolean closed;

  @Column(name = "signatures_count")
  private int signatures_count;

  @Column(name = "delegate_id")
  private Long delegate_id;

  @Column(name = "theme_id")
  private Long theme_id;

  @Enumerated(EnumType.STRING)
  private Vote final_vote;

  @Embedded private LawProjectVotation votation;
```

Anotações JPA da classe LawProject:

- @Entity - Define LawProject como uma classe que pode ser persistida na Base de Dados.
- @Id - Define id como a chave primária da tabela LawProject.
- @GeneratedValue(strategy = GenerationType.SEQUENCE) - Indica que os valores do id serão gerados automática e sequencialmente.
- @Transient - Indica que *executor* não vai ser persistido na Base de Dados.
- @Column(name = “name”) - Define o nome da coluna com o nome *name*.
- @Lob - Persiste um ficheiro na base de dados

- @Convert(converter = LocalDateTimeConverter.class) - Converte a classe LocalDateTime na classe LocalDateTimeConverter de modo a ser possível persistir *close_date* na Base de Dados.
- @Enumerated(EnumType.STRING) - Persiste na base de dados um atributo de um enumerado.
- @Embedded - Mapeia uma classe @Embeddable num objeto de entidade.

Classe LawProjectVotation

```
package pt.ul.fc.css.project.entities;

import jakarta.persistence.Embeddable;

/** Class that creates Projects in votation */
@Embeddable
public class LawProjectVotation {

  // Close date
  private LocalDateTime close_date_votation;

  @SuppressWarnings("unused")
  private boolean isActive; // Used in a query

  @Transient private ScheduledExecutorService executor; // To use for the countdown

  // Counter for favorable votes
  private int favorable_vote_counter;
  // Counter for unfavorable votes
  private int unfavorable_vote_counter;

  // Law project that is associated with this votation
  @OneToOne private LawProject lawProject;
```

Anotações JPA da classe LawProjectVotation:

- @Embeddable - Indica que a classe LawProjectVotation pode ser incorporada noutra entidade (neste caso na entidade LawProject) como um componente embutido. Deste modo, os atributos desta classe serão mapeados para os campos da tabela da classe que incorpora LawProjectVotation, ou seja, para a tabela da entidade LawProject.
- @Transient - Indica que o atributo executor não vai ser mapeado para a tabela que incorpora LawProjectVotation.
- @OneToOne - Relação um-para-um que indica que um objeto LawProjectVotation tem/conhece um LawProject e vice-versa.

Classe Theme

```java
package pt.ul.fc.css.project.entities;

import jakarta.persistence.ElementCollection;

/** The class that creates objects for the theme */
@Entity
public class Theme {

  @NonNull private String name; // the name of the theme

  @Id
  @GeneratedValue(strategy = GenerationType.SEQUENCE)
  private Long id; // primary key

  @ManyToOne private Theme parent; // the parent of the current theme

  @ElementCollection
  @OneToMany(fetch = FetchType.EAGER)
  @JoinColumn(name = "children")
  private List<Theme> children = new ArrayList<Theme>(); // list of childrens of the theme

  public Theme() {}

  /**
   * Constructor
   *
   * @param name the name of the theme
   */
  public Theme(@NonNull String name) {
    this.name = name;
  }
```

Anotações JPA da classe Theme:

- @Entity - Define Theme como uma classe que pode ser persistida na Base de Dados.
- @Id - Define id como a chave primária da tabela Theme.
- @GeneratedValue(strategy = GenerationType.SEQUENCE) - Indica que os valores do id serão gerados automática e sequencialmente.
- @ManyToOne - Relação muitos-para-um que indica que vários Theme podem ter o mesmo *parent*, ou seja, que um *parent* pode ter vários Theme.
- @ElementCollection - Mapeia a coleção children que é um mapeamento de Theme.
- @OneToMany(fetch = FetchType.EAGER) - Relação um-para-muitos que indica que um Theme pode ter vários objetos Theme filhos. fetch = FetchType.EAGER especifica que os objetos Theme desta lista devem ser carregados imediatamente juntos com a classe Theme mãe, sem necessidade desta ir buscar os objetos filhos posteriormente.

- @JoinColumn(name = “children”) - Adiciona uma coluna à tabela da classe Theme para armazenar as foreign keys dos objetos filhos, sendo o nome dessa coluna “children”.
- @NonNull - Indica que o parâmetro *name* não pode ser nulo.

Classe LawProjectController

```
package pt.ul.fc.css.project.controllers;

import java.time.LocalDateTime;

/** Controller for the LawProject class */
@Component
public class LawProjectController {
```

Anotações JPA da classe LawProjectController:

- @Component - Marca a classe como estando disponível para outras classes (neste caso uma classe de serviço) para usá-la como uma dependência. Deste modo, evitamos Dependency Injections.

Classe LawProjectService

```
package pt.ul.fc.css.project.services;

import java.time.LocalDateTime;

/** Service class for LawProjectController */
@Service
public class LawProjectService {
```

Anotações JPA da classe LawProjectService:

- @Service - Marca a classe como estando disponível para outras classes que precisam de usar os seus recursos e funcionalidades. Neste projeto, as classes de testes interagem diretamente com os serviços criando uma camada (service layer) e não sendo necessário interagir diretamente com a camada de dados nem com o controlador/handler, criando assim uma melhor organização por camadas.

Limitações do Projeto:

- No teste do caso de uso F é necessário esperar 1 minuto para verificar se o método countdown() funciona corretamente.
- Inicialmente ao tentar fazer a conexão com a Base de Dados usando o pgAdmin 4 (visto que usamos o SpringBoot), obtivemos um erro. Conseguimos consertá-lo e fazer a conexão com Base de Dados corretamente alterando a linha 11 do ficheiro application.properties de jdbc:postgresql://postgres:5432/testedb para jdbc:postgresql://localhost:5432/democracia. Temos em conta que poderá ter sido um equívoco de configuração do nosso servidor, mas esta foi a solução que conseguimos encontrar.

application.properties

```
1  spring.jpa.database-platform = org.hibernate.dialect.PostgreSQLDialect
2  #spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation = true
3  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
4  #spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.H2Dialect
5  # Hibernate ddl auto (create, create-drop, validate, update)
6  #spring.jpa.hibernate.ddl-auto = update
7  # for production environment, this variable should be validate
8  spring.jpa.hibernate.ddl-auto=create-drop
9
10
11 spring.datasource.url = jdbc:postgresql://localhost:5432/democracia
12 spring.datasource.username = postgres
13 spring.datasource.password = admin
14
15 spring.jpa.show-sql=true
16
17
18 logging.level.org.springframework=INFO
19 logging.level.com.mkyong=INFO
20 logging.level.com.zaxxer=DEBUG
21 logging.level.root=ERROR
22
23
24 spring.datasource.hikari.connectionTimeout=20000
25 spring.datasource.hikari.maximumPoolSize=5
26
27 logging.pattern.console=%-5level %logger{36} - %msg%n
```