Relatório do Projeto 1 - CSS

Trabalho realizado por:

Diogo Pinto fc55179

Ivo Estrela fc51051

Thiago Duarte fc53636

Mapeamento usado no projeto (ORM)

Neste projeto tomámos as seguintes decisões nas anotações JPA para o mapeamento:

Entidades (@Entity): Bill, Citizen, Delegate, DelegateTheme, Poll e Theme.

@Entity: essa anotação marca uma determinada classe como uma entidade JPA, o que significa que ela será persistida em uma tabela de banco de dados correspondente.

1. Explicação das anotações relativas a entidade **Bill**:

```
@Id @GeneratedValue private Long id;
```

@Id : Anotação utilizada para indicar que o campo id é a chave primária da entidade.

@GeneratedValue: Anotação utilizada, em conjunto com o @Id, para definir a estratégia de geração dos valores para a chave primária. Neste caso, como não está especificada a estratégia, o provedor de persistência (banco de dados utilizado) escolhe a estratégia de geração.

```
@NonNull private String title;

@NonNull private String description;
```

@NonNull: Anotação utilizada nas colunas onde o valor não pode ser nulo.

```
@Lob @NonNull private byte[] fileData;
```

@Lob : Anotação utilizada para persistir dados binários, neste caso, persistir dados binários do ficheiro pdf com o conteúdo principal do projecto de lei.

@NonNull: Explicação trivial já abordada anteriormente.

```
@ManyToMany
@Cascade(CascadeType.ALL)
private final List<Citizen> supporters = new ArrayList<>();
```

@ManyToMany: Anotação que indica que há um relacionamento de muitos-para-muitos entre as entidades Bill e Citizen. Ou seja, um objeto Bill pode ter muitos objetos Citizen associados a ele (muitos cidadões podem apoiar um projeto de lei), e a um objeto Citizen pode ter muitos objetos Bill associados a ele (muitos projetos de lei podem ser apoiados por um cidadão).

@Cascade(CascadeType.ALL): Indica que todas as operações de persistência (inserção, atualização e exclusão) devem ser em cascata, para as entidades associadas (os Citizens apoiadores do projeto). Portanto, ao inserir/atualizar/excluir um objeto Citizen, este objeto também será inserido/atualizado/excluído da lista de supporters.

```
@DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME)
@NonNull
private LocalDate expirationDate;
```

@DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME): Anotação utilizada para especificar o formato da data e hora. Neste caso, a anotação está sendo usada para especificar que o formato da data e hora deve ser ISO_DATE_TIME.

```
@Enumerated(EnumType.STRING)
private BillStatus status = BillStatus.OPEN;
```

@Enumerated(EnumType.STRING): Esta anotação é usada para mapear um campo que representa um tipo enumerado (enum) em uma entidade JPA. A opção EnumType.STRING indica que os valores do enum serão armazenados como Strings no banco de dados.

```
@OneToOne
@Cascade(CascadeType.ALL)
@NonNull
private Theme theme;
```

@OneToOne: A anotação indica que há uma relação de um-para-um entre as entidades Bill e Theme. Ou seja, um objeto Bill tem um tema associado a ele (um projeto tem um tema).

@Cascade(CascadeType.ALL) : Indica que todas as operações de persistência (inserção, atualização e exclusão) devem ser em cascata, para a entidade Theme associada. Portanto, ao inserir/atualizar/excluir um objeto Theme, a associação com este objeto também será inserida/atualizada/excluida do banco de dados.

@NonNull: Explicação trivial já abordada anteriormente.

```
@OneToOne
@Cascade(CascadeType.ALL)
@NonNull
private Delegate delegate;
```

Exatamente o mesmo caso que o anterior, mas a relação é entre a entidade Bill e a entidade Delegate.

```
@OneToOne(mappedBy = "associatedBill")
@Cascade(CascadeType.ALL)
private Poll associatedPoll;
```

@OneToOne(mappedBy = "associatedBill"): A anotação indica que há uma relação de um-para-um, bidirecional, entre as entidades Bill e Poll. Ou seja, um objeto Bill tem uma poll associado a ele (um projeto tem uma votação) e vice versa (há uma votação para um projeto).

O atributo mappedBy indica que a entidade Bill é a dona da associação, pois a criação da poll depende de uma condição relacionada a bill (a bill ter mais do que 10000 apoiadores).

2. Explicação das anotações relativas a entidade **Citizen**:

```
@Entity
@Inheritance(strategy = SINGLE_TABLE)
public class Citizen {
```

@Inheritance(strategy = SINGLE_TABLE): Anotação usada para definir a estratégia de herança a ser usada em uma hierarquia de classes no JPA. Neste caso, a estratégia SINGLE_TABLE é definida, o que significa que todas as classes na hierarquia de herança serão mapeadas para a mesma tabela no banco de dados. Esta estratégia faz sentido pois a subclasse Delegate não possui nenhum atributo diferente da superclasse Citizen, o que significa que não temos o problema de ter uma tabela esparsa, pois não existirá campos com valores null (pelo menos nesta fase do projeto). Além disso, esta estratégia é extremamente eficiente em termos de performance, e como a tabela Citizen terá muitas linhas (pois segundo o enunciado temos mais do que 10 milhões de habitantes), é necessário este tipo de propriedade para a aplicação.

```
@Column(unique = true)
@NonNull
private Integer cc;
```

@Column(unique = true): Anotação que indica que a coluna correspondente ao atributo cc na tabela do banco de dados deve ter uma restrição de unicidade, ou seja, nenhum outro registro poderá ter o mesmo valor nessa coluna. Isso faz sentido, pois cada cidadão português possui possui um número do cartão de cidadão único, e como a autenticação da aplicação será feita com o cartão de cidadão, é importante garantir na base de dados a unicidade desta coluna.

```
@ManyToMany(mappedBy = "voters")
@Cascade(CascadeType.ALL)
private final List<DelegateTheme> delegateThemes = new ArrayList<>();
```

@ManyToMany(mappedBy = "voters"): A anotação indica que existe uma relação de muitos-para-muitos, bidirecional, entre as entidades Citizen e DelegateTheme. Ou seja, vários objetos Citizen podem ter muitos objetos DelegateTheme associados a eles (vários cidadões podem ter vários delegados (um para cada tema – lógica presente na entidade DelegateTheme)) e vice-versa (vários delegados (com um determinado tema - lógica presente na entidade DelegateTheme) podem representar vários cidadões).

3. Explicação das anotações relativas a entidade **Delegate**:

Sem anotações (além do @Entity).

4. Explicação das anotações relativas a entidade **DelegateTheme**:

```
@ManyToOne private Theme theme;
```

@ManyToOne: A anotação indica que a classe DelegateTheme tem um relacionamento de muitos-para-um com a classe Theme. Isso significa que várias instâncias da classe DelegateTheme podem estar associadas a uma única instância da classe Theme (vários delegados-tema estão associados a um só tema), e que uma instância da classe Theme pode estar associada a várias instâncias da classe DelegateTheme (um tema pode ter vários DelegateTheme associados a ele, ou seja, podem existir vários objetos DelegateThemes associados ao tema Saúde, por exemplo).

```
@ManyToOne private Delegate delegate;
```

@ManyToOne: A anotação indica que a classe DelegateTheme tem uma relação de muitos-para-um com a classe Delegate. Isso significa que várias instâncias da classe DelegateTheme podem estar associadas a uma única instância da classe Delegate (vários delegados-tema estão associados a um só delegado), e que uma instância da classe Delegate pode estar associada a várias instâncias da classe DelegateTheme (um delegado pode ter vários DelegateTheme associados a ele, ou seja, podem existir várias relações delegado-tema associadas ao delegado Pedro, por exemplo).

```
@ManyToMany private List<Citizen> voters;
```

@ManyToMany: Anotação explicada nas anotações do Citizen (relação bidirecional).

5. Explicação das anotações relativas a entidade **Poll**:

```
@ElementCollection
@CollectionTable(name = "public_voters")
@MapKeyJoinColumn(name = "delegate_id")
@Column(name = "vote_type")
@Enumerated(EnumType.STRING)
@Cascade(CascadeType.ALL)
private final Map<Delegate, VoteType> publicVoters = new HashMap<>();
```

@ElementCollection: Indica que o atributo publicVoters é uma coleção de tipos simples ou incorporáveis (guarda VoteType que é guardado como String).

@CollectionTable(name = "public_voters"): Especifica que o nome da tabela que guarda a coleção de elementos tem nomenclatura "public_voters".

@MapKeyJoinColumn(name = "delegate_id"): Especifica que o nome da coluna relativa a chave estrangeira que referencia a entidade Delegate tem nomenclatura "delegate_id".

@Column(name ="vote_type"): Especifica que o nome da coluna que guarda o valor do enumerado VoteType tem nomenclatura "vote_type".

```
@OneToMany private final List<Citizen> privateVoters = new ArrayList<>();
```

@OneToMany: Indica que a classe Poll tem uma relação de um-para-muitos com a classe Citizen. Isso significa que uma poll pode ter votos de vários cidadões, e que vários cidadãos podem votar uma vez por Poll.

```
@OneToOne
@Cascade(CascadeType.ALL)
@NonNull
private Bill associatedBill;
```

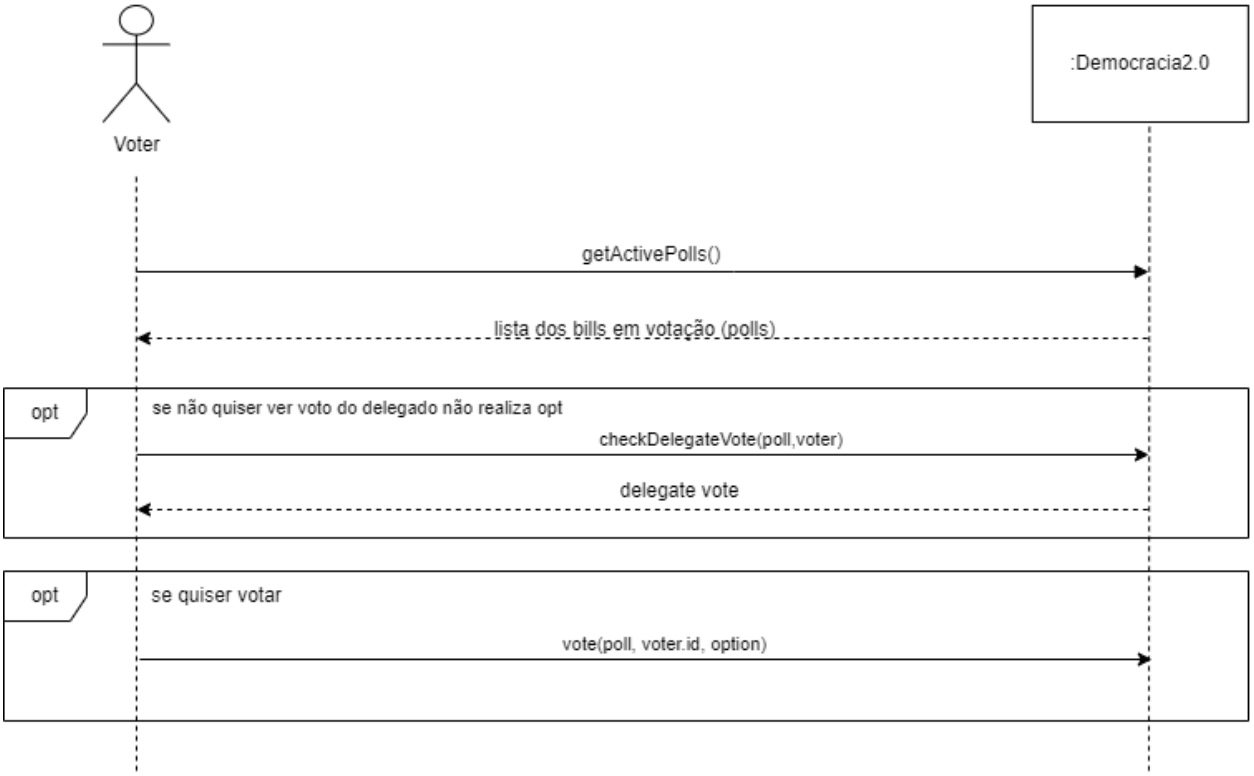
@OneToOne: Anotação explicada nas anotações do Bill (relação bidirecional).

6. Explicação das anotações relativas a entidade **Theme**:

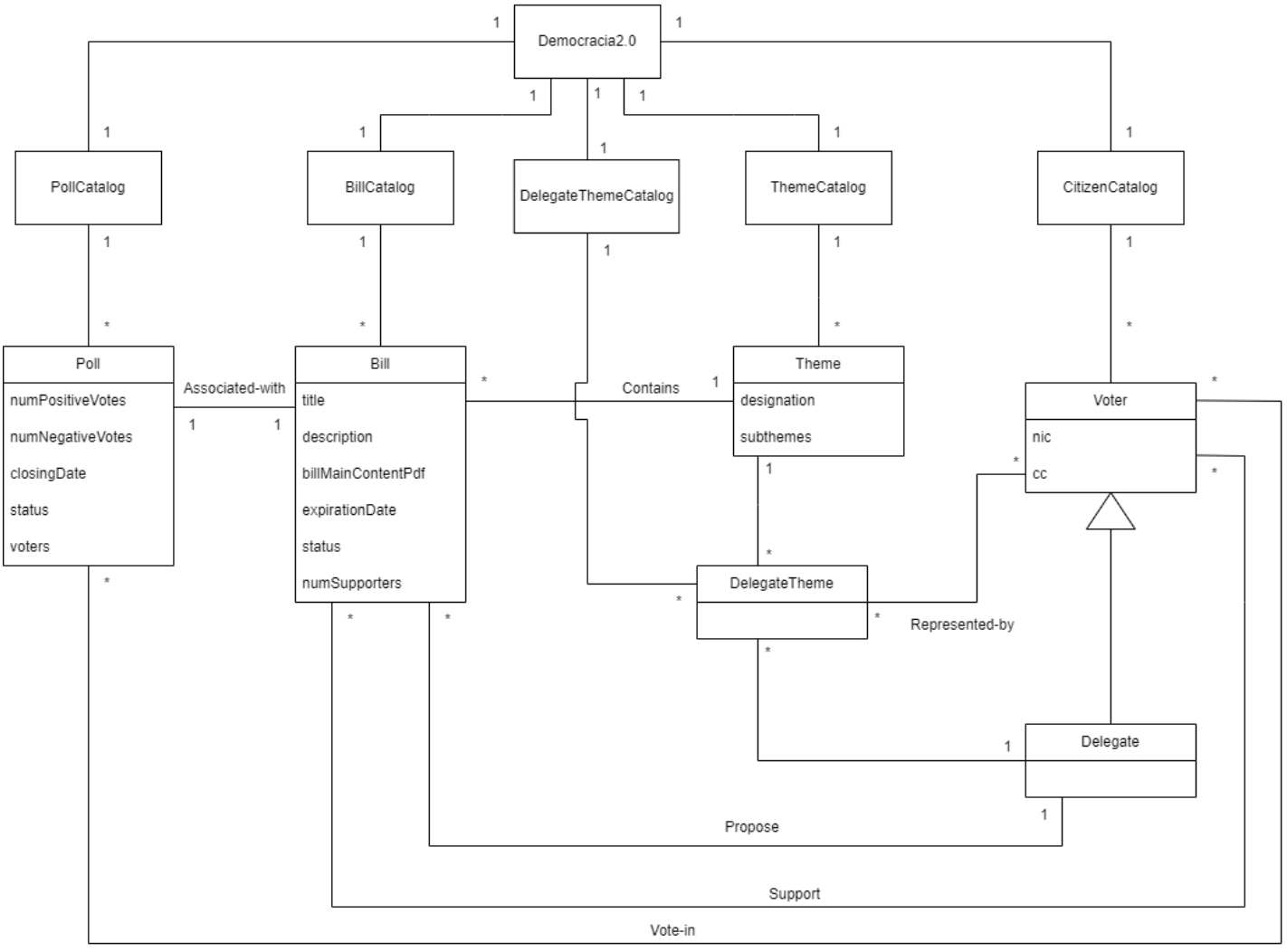
```
@OneToMany(mappedBy = "parentTheme")
@Cascade(CascadeType.ALL)
private List<Theme> subThemes;

@ManyToOne private Theme parentTheme;
```

Ambas as relações indicam uma relação de um-para-muitos (do ponto de vista do tema pai), ou seja, um tema pode possuir vários subtemas; e vários subtemas possuem um tema pai - parentTheme (do ponto de vista dos subtemas).

SSD - Use Case J: Votar numa proposta



Domain Model



Class Diagram