Construção de Sistemas de Software 2022/23

Relatório Projeto

- Fase 2-

-Tomás Piteira, 56303
-Daniel Lopes, 56357
-Miguel Ramos, 56377

Índice

Modelo de Domínio

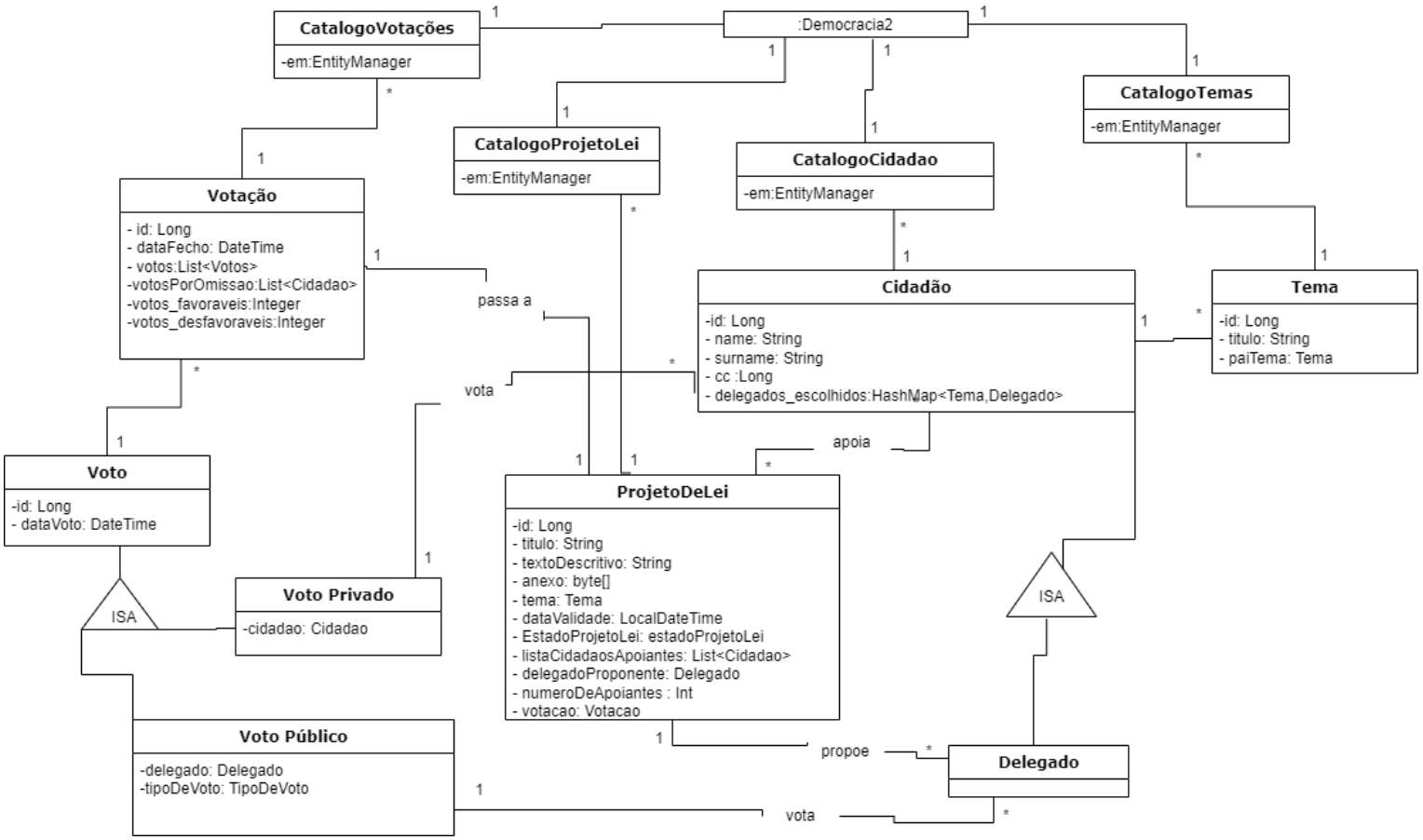
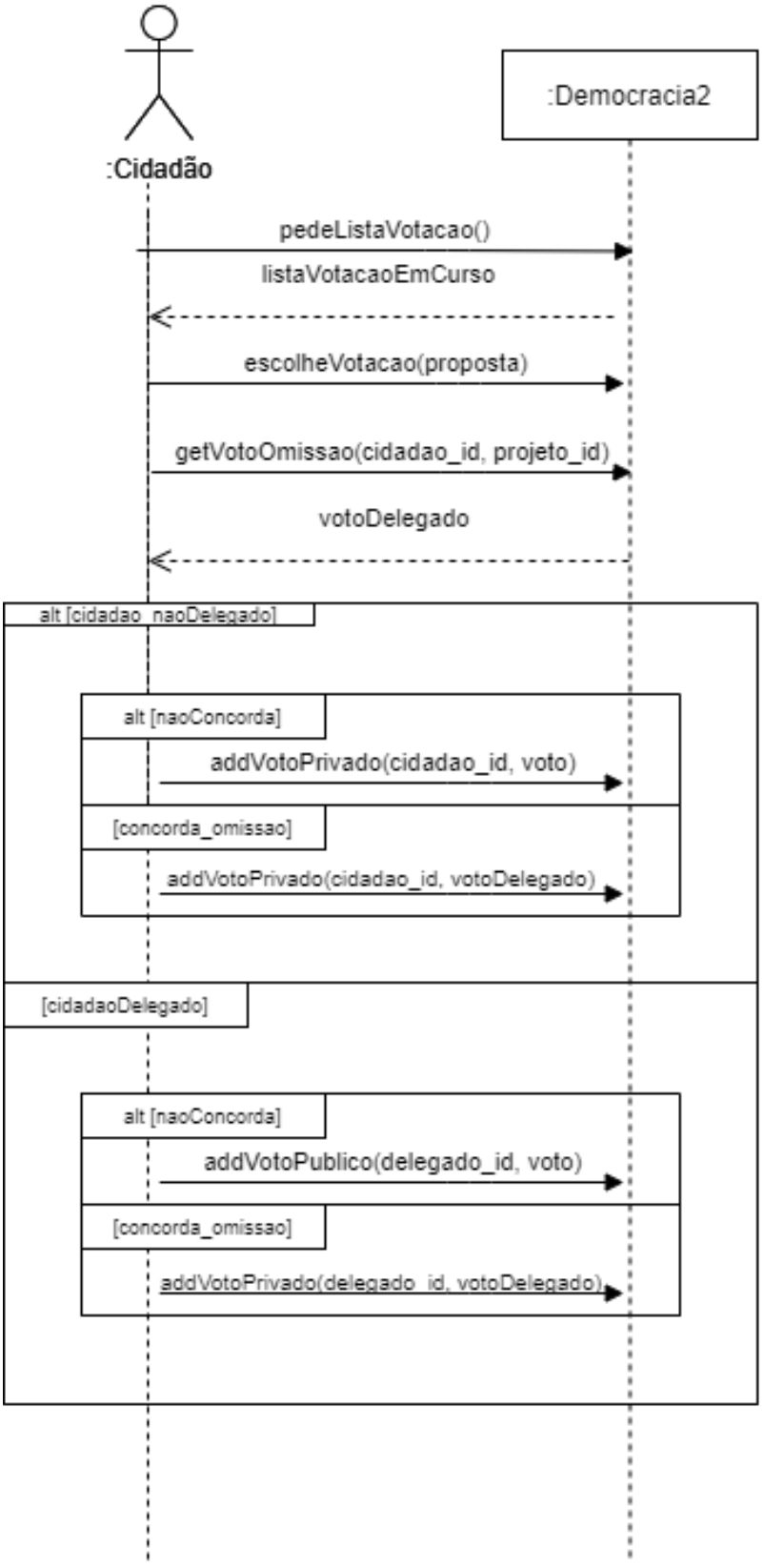


Diagrama de Classes

NOTA: Diagrama colocado na pasta Image

Caso de Uso J: Votar numa Proposta



Decisões de Implementação

Estrutura:

O nosso grupo decidiu dividir a aplicação Democracia2 em 4 camadas: Camada de Apresentação, Camada de Serviço/Aplicação, Camada de Negócio e Camada de Dados.
Como nesta aplicação o processamento aos casos de uso envolvem múltiplos recursos transacionais preferimos criar a Camada de Aplicação. Esta tem duas secções a Democracia2Service que é a Camada de Aplicação mais “acima” enquanto que as Camadas CidadaoService, DelegadoService e ProjetoDeLeiService estão mais “abaixo”.
Foram usados Catálogos e Handlers devido a nossa utilização anterior em DCO, o que nos permitiu um maior à vontade e experiência. Foram usados EntityManager ao invés de repositórios pois conseguimos ter uma maior flexibilidade e controle sobre as operações de persistência. Embora acha desvantagem em termos de escrever o código, o que no caso dos repositórios não aconteceria, conseguimos realizar operações complexas diretamente.
Na implementação da 2º fase do projeto foram criadas classes DTOs (Data Transfer Objects) para facilitar a transferência de dados entre o cliente e o servidor. Os DTOs desempenham um papel importante na separação das entidades de domínio do sistema das estruturas de dados usadas para comunicação com o mundo exterior, como as requisições e respostas HTTP. Eles são usados para encapsular os dados relevantes de uma entidade num objeto simples e serializável, que pode ser transmitido pela rede ou usado como modelo para a renderização de páginas.

Aplicação Web (F2W):

Aqui vamos comentar o desenvolvimento da aplicação Web para os casos de uso identificados por F2W. Para este fim foram criadas 3 diferentes classes: WebController, WebEscolheDelegadoController e WebProjetoLeiController. Todas elas têm a anotação @Controller, indicando que é são controladores de requisições web. Também possuem a dependência Autowired para a classe "Democracia2Service", fornecendo assim os serviços necessários para manipulação dos dados do projeto.

1. **Classe WebController**

-Esta classe é responsável por lidar com as requisições HTTP relacionadas à interface web;

-Realçar a existência de um método **init** que é executado quando a URL "/init" é acedida. Esse método é responsável por realizar algumas ações iniciais no projeto, como criar temas, delegados, cidadãos e projetos de lei, e realizar algumas operações relacionadas a eles;
NOTA : Só se pode correr uma vez pois a segunda irá dar erro por estar a criar entidades a mais com chaves únicas

-Temos métodos **getLogin** e **getIndex** que são acionados com as URLs "/login" e "/index", respetivamente, e que retornam a página de login e a página inicial do projeto;
-Existem métodos **newCustomer** e **newCustomerAction**. O primeiro retorna a página para criar um novo utilizador (Cidadão), enquanto o segundo trata a ação de criar um novo utilizador, verificando a função (Cidadão ou Delegado) e realizando as operações necessárias;

-Criamos um método **login** que é acionado quando a URL "/login" é acedida com o parâmetro "cc". Esse método realiza o login do utilizador correspondente ao número de identificação.

2. **Classe WebEscolheDelegadoController**

-Esta classe é responsável por lidar com as requisições HTTP relacionadas à escolha de delegados pelos cidadãos ou pelos próprios;

-Temos métodos fundamentais como **escolherDelegado** e **escolheDelegado** para tratar as requisições POST e GET relacionadas à escolha de delegados, respetivamente. O método escolherDelegado trata a ação de escolher um delegado para um determinado tema, verificando se já existe um delegado associado a esse tema para o utilizador atual. Já o escolheDelegado retorna a página para escolher um delegado, exibindo a lista de delegados e temas disponíveis;

3. **Classe WebProjetoDeLeiController**

-Esta classe é responsável por lidar com as requisições HTTP relacionadas com os projetos de lei existentes. Segue-se a breve explicação de cada método:

-O método apoiarProjeto é chamado quando um utilizador apoia um projeto de lei. Ele adiciona uma mensagem de sucesso ao modelo e redireciona para a página de projetos de lei não expirados;

-O método votacoesEmCurso lista as votações em curso e adiciona a lista ao modelo, retornando a página "votacoes_curso_list";

-O método consultaListaProjetosNaoExpirados consulta a lista de projetos de lei não expirados e adiciona a lista ao modelo, retornando a página "ProjetosDeLeiNaoExpirados".

-O método consultaProjetoNaoExpirados consulta um projeto de lei não expirado com base no ID fornecido e adiciona o projeto ao modelo, retornando a página "projetoDeLei_detail".

-O método consultaVotacao consulta uma votação com base no ID fornecido e adiciona a votação ao modelo, retornando a página "votacao_detail";

-O método apresentarProjetoDeLei é chamado quando é apresentado um novo projeto de

lei. Ele realiza várias etapas, incluindo o armazenamento de um anexo num arquivo temporário, e em seguida, redireciona para a página de projetos de lei não expirados;
-O método novoProjeto adiciona os temas disponíveis ao modelo e retorna a página "apresentarProjetoLei";

-O método votar consulta um projeto de lei com base no ID fornecido e adiciona informações relacionadas ao voto ao modelo, retornando a página "votar";

-O método votarProjetoDeLei é chamado quando um utilizador vota em um projeto de lei. Ele realiza a votação e redireciona para a página de votações em curso.

-Todas estas páginas para o qual são redirecionadas encontram-se na pasta “templates” com o respetivo estilo associado com base no código css também fornecido.

API REST (F2R):

Nesta secção vamos abordar a API REST desenvolvida para os casos de uso identificados por F2R. Utilizamos a anotação @RestController indicando que a classe RestProjetoLeiController é um controlador REST que lida com requisições HTTP e retorna respostas no formato JSON. Também a anotação @RequestMapping("/api") que define o caminho base para todas as rotas do controlador, ou seja, sabemos que todas as rotas serão acedidas a partir de /api. As dependências Democracia2Service e ProjetoDeLeiService são injetadas usando a anotação @Autowired, o que nos permitiu utilizá-las. Vamos agora analisar de forma breve cada método e etapa existente nesta classe:

Definição e configuração do controlador: O controlador RestProjetoLeiController é responsável por lidar com as solicitações da API REST. Ele possui várias rotas mapeadas para executar diferentes ações relacionadas com os projetos de lei;

Apoiar um projeto de lei: A rota POST /api/projetos-de-lei-n-expirados/apoio/{id}/{cc} permite que um utilizador (identificado pelo número do cartão de cidadão - {cc}) apoie um projeto de lei específico (identificado pelo ID - {id}). O método apoiarProjeto_ dentro do controlador verifica se o utilizador está autenticado e, em seguida, chama o serviço projetoDeLeiService para registar o apoio ao projeto de lei;

Listar votações em curso: A rota GET /api/votacoes-em-curso retorna uma lista de projetos de lei que estão em votação no momento. O método allVotacoesCurso dentro do controlador chama o serviço projetoDeLeiService para obter essa lista;

Consultar lista de projetos não expirados: A rota GET /api/projetos-de-lei-n-expirados retorna uma lista de todos os projetos de lei que ainda não expiraram. O método consultaListaProjetosNaoExpirados dentro do controlador chama o serviço projetoDeLeiService para obter essa lista;

Login: A rota POST /api/login/{cc} é usada para fazer login de um utilizador com base no número do cartão de cidadão {cc} fornecido. O método login dentro do controlador verifica se o utilizador existe e, em caso afirmativo, armazena o utilizador atual no serviço democracia2Service;

Consultar projeto de lei não expirado: A rota GET /api/projetos-de-lei-n-expirados/{id} retorna os detalhes de um projeto de lei não expirado com base no ID fornecido. O método consultaProjetoNaoExpirados dentro do controlador chama o serviço projetoDeLeiService para obter esses detalhes;

Votar num projeto de lei: A rota GET /api/votacao/{id}/votar/{cc} é usada para permitir que um utilizador (identificado pelo número do cartão de cidadão - {cc}) vote num projeto de lei específico (identificado pelo ID - {id}). O método votar dentro do controlador chama o serviço projetoDeLeiService para obter os detalhes do projeto de lei e retorna um objeto ModelMap contendo as informações necessárias para exibir a página de votação;

Registar o voto num projeto de lei: A rota POST /api/votacao/{id}/votar/{cc}/{voto} é usada para registar o voto de um utilizador (identificado pelo número do cartão de cidadão - {cc}) em um projeto de lei específico (identificado pelo ID - {id}). O método votarProjetoDeLei dentro do controlador chama o serviço projetoDeLeiService para registar o voto no projeto de lei;

Scheduled Tasks (C):

Nesta fase implementamos os casos de uso C à custa de Scheduled Tasks. Vamos ter duas classes: FechaProjetosLeiScheduled e FechaVotacoesService. Estas classes e os seus métodos agendados são responsáveis por automatizar o processo de fechamento de projetos de lei expirados e votações expiradas no sistema. Essas tarefas agendadas são executadas em intervalos regulares de 30 segundos para verificar se há projetos de lei ou votações que atingiram sua data de validade e, caso tenham, são fechados. Os logs utilizados são usados para registar informações sobre o sucesso ou falha dessas operações.

Democracia2_Desktop/JAVAFX (F2D):

A aplicação javafx foi feita através do exemplo dado pelos professores (javafxexample).
Tem como estrutura:

Application: Contém a classe main para que seja possível correr o projeto;
Control: package que contém os controllers que são usados para realizar os pedidos HTTP ao servidor e que interagem com as páginas fxml;
Models: Modelos de dados para que sejas possível

Páginas: Páginas fxml que contém tabelas para que seja possível mostrar as listas de Votações ou Projetos de Lei, botões para apoiar, votar e navegar entre páginas.

Para realizar o mapeamento dos dados que serão enviados pelos pedidos HTTP feitos ao servidor foi usado o GSON.

Testes

Os testes foram implementados tendo em consideração todos os casos de uso implementados no RestProjetoDeLeiController. Tentámos ao máximo testar todos os possíveis focos de erro no nosso código, tendo estes mesmos testes sido um objeto essencial no nosso projeto, na medida em que nos ajudou a alcançar alguns problemas que, até então, não tinham sido detetados por nenhum membro do grupo.