Faculdade de Ciências da Universidade de Lisboa

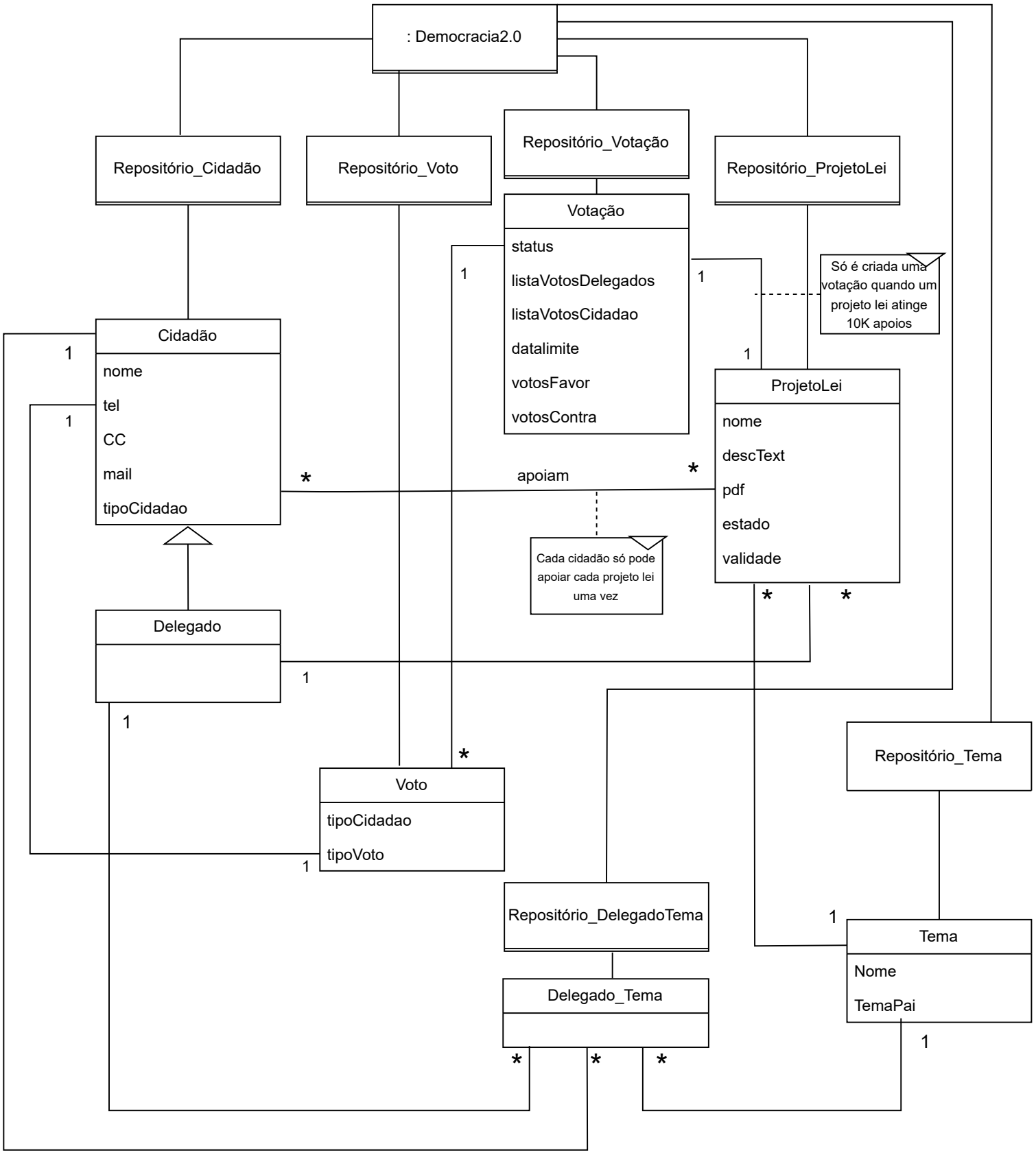
Construção de Sistemas de Software

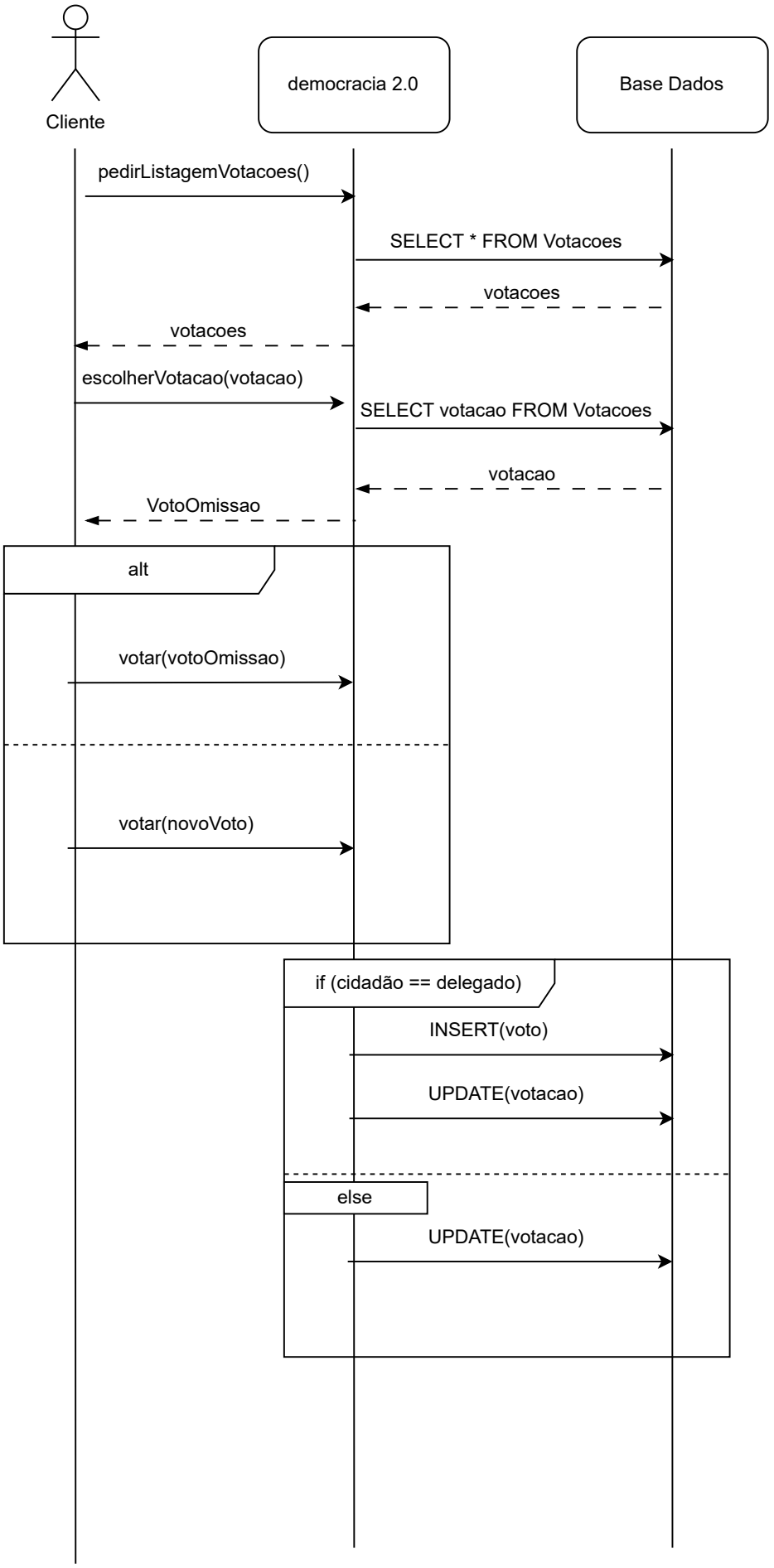
Relatório do Projeto Democracia2.0

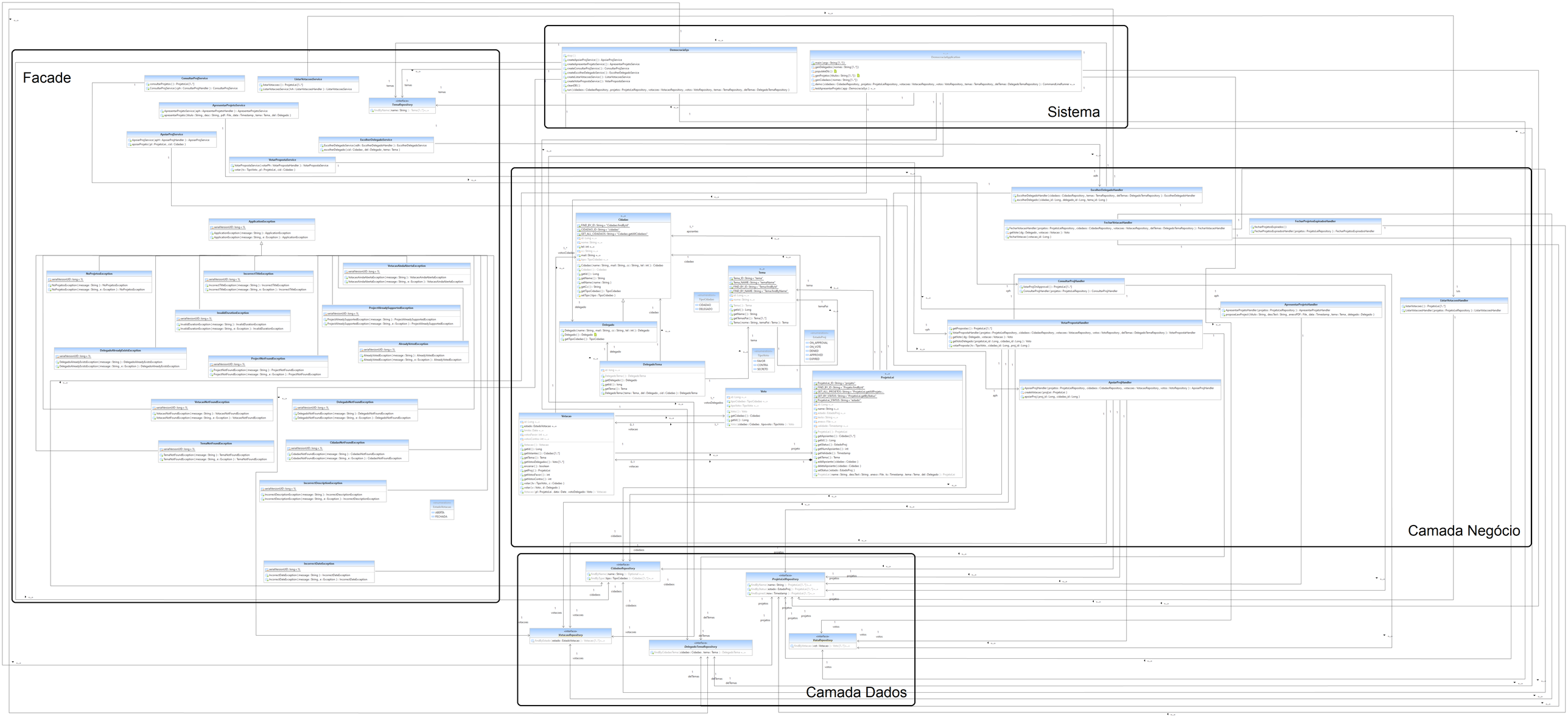
Alexandre Müller fc56343

Diogo Ramos fc56308

Francisco Henriques fc56348


: Democracia2.0
Repositório_Cidadão
Repositório_Voto
Repositório_Votação
Repositório_ProjetoLei
Votação
status
listaVotosDelegados
listaVotosCidadao
datalimite
votosFavor
votosContra
Só é criada uma votação quando um projeto lei atinge 10K apoios
Cidadão
nome
tel
CC
mail
tipoCidadao
apoiam
ProjetoLei
nome
descText
pdf
estado
validade
Cada cidadão só pode apoiar cada projeto lei uma vez
Delegado
Voto
tipoCidadao
tipoVoto
Repositório_Tema
Repositório_DelegadoTema
Delegado_Tema
Tema
Nome
TemaPai
1
*


Cliente
democracia 2.0
Base Dados
pedirListagemVotacoes()
SELECT * FROM Votacoes
votacoes
votacoes
escolherVotacao(votacao)
SELECT votacao FROM Votacoes
votacao
VotoOmissao
alt
votar(votoOmissao)
votar(novoVoto)
if (cidadão == delegado)
INSERT(voto)
UPDATE(votacao)
else
UPDATE(votacao)


Facade
Sistema
Camada Negócio
Camada Dados

Aplicação WEB

Para concretizar uma aplicação WEB capaz de fornecer o *front-end* para aplicação Democracia2.0 foi necessário criar um controlador capaz de processar os pedidos, enviá-los ao *back-end* e receber as respostas.

Para tal, criámos a classe WebController com os seguintes métodos:

- getIndex – realiza o mapeamento para pedidos GET realizados ao URL "/" e redireciona para a página inicial;
- login – realiza o mapeamento para pedidos POST ao URL "/login", ao receber um nº de CC, realizar o pedido de login ao *back-end* e processar a resposta do mesmo. No fim, redireciona para a página inicial após adicionar o *user* corrente ao modelo;
- logout – realiza o mapeamento para pedidos POST ao URL "/logout" realizando o logout do *user* no *back-end*;
- escolherDelegado – realiza o mapeamento para pedidos do tipo GET ao URL "/escolherDel", redirecionando para a página de escolha de delegado após adicionar ao modelo os temas disponíveis no *back-end*;
- escolherDel – método que complementa o anterior ao mapear pedidos POST para a escolha de um delegado, através da receção de um nº de CC e de um tema e da realização de um pedido ao *back-end* para fazer o devido registo do delegado na BD;
- apresentar – método que mapeia pedidos GET realizados ao URL "/apresentar" e que redireciona para a página onde é possível apresentar um novo projeto de lei;
- apresentarProj – método complementar ao anterior ao mapear pedidos POST realizados na página do método anterior. Para isto, recebe todos os detalhes necessários à criação de um projeto de lei e envia um pedido ao *back-end* para que este possa efetuar as verificações necessárias e registar o novo projeto na sua BD;
- listVotes – método que mapeia pedidos GET ao URL "/votes" e que adiciona ao modelo a lista de votações existentes antes de redirecionar para a página que mostra essa mesma lista;
- getProjetoById – realiza o mapeamento de pedidos GET ao URL "/projeto/id" para ir buscar o projeto que corresponde ao id fornecido e mostrá-lo no *front-end*;
- apoiarProjeto – realiza o mapeamento a pedidos POST ao URL "/projeto/apoiar" para efetuar um pedido de registo de um apoio ao projeto fornecido no *body* do pedido POST;
- votarProjeto – método similar ao anterior, mas em vez de enviar um pedido de registo de um novo apoio, envia um pedido de registo de um novo voto num dado projeto em votação ao receber todos os campos necessários e enviá-los ao *back-end*;
- getListaProjetos – realiza o mapeamento a pedidos GET ao URL "/projeto" enviando a lista de todos os projetos existentes na BD.

Para estes efeitos foi necessário usar os services implementados na fase anterior usando a anotação Autowired que injeta dependências permitindo assim inicializar os serviços automaticamente.

API REST

Para concretizar uma API REST capaz de responder a pedidos HTTP feitos pelos utilizadores, para isso foi necessário criar um controlador REST capaz de processar os pedidos, enviá-los ao *back-end*, receber as respostas e enviar uma resposta HTTP com um determinado código.

Este controlador é bastante semelhante ao controlador WEB nas suas funcionalidades nucleares com a grande diferença a ser o tipo de resposta, pois o controlador WEB retorna o nome de um ficheiro HTML e o controlador REST processa a resposta do servidor e envia uma resposta HTTP com códigos referentes a um estado HTTP e informação útil referente à resposta do pedido no corpo desta mesma resposta.

Ao contrário da aplicação WEB onde foi usada a anotação @Controller, na API REST foi usada a anotação @RestController pois esta anotação é a usada para especificar um controlador de uma API REST. Adicionalmente também foi utilizada a anotação @RequestMapping("api") pois todos os URLs referentes a pedidos REST começam com o mesmo prefixo (http://localhost:8080/api/).

Aplicação Desktop

Para criar uma aplicação Desktop nativa para o projeto Democracia2.0 foi necessário criar diversos controladores referentes às várias funcionalidades implementadas e vários ficheiros FXML através do SceneBuilder para representar a GUI (*Graphic User Interface*) da aplicação.

Cada um dos controladores é ativado pela sua respetiva página na GUI e é responsável por receber e tratar os pedidos GET e POST, enviá-los à aplicação que corre o *back-end*, receber as respetivas respostas e ativar mensagens na *app* consoante as respostas recebidas.

De referir que para esta aplicação foi usada a API REST criada anteriormente como forma de tratamento dos pedidos e comunicação com a camada de negócio da aplicação Democracia2.0 criada na fase anterior.