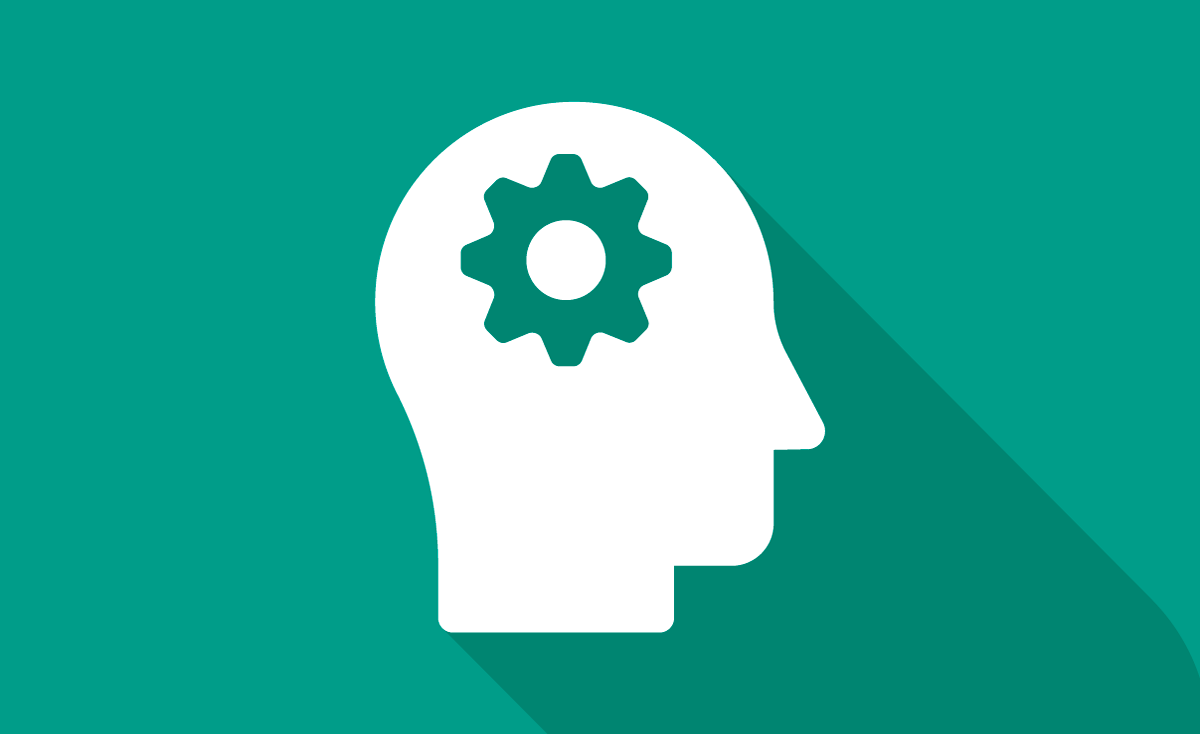


**Programação**

**Orientada a Objetos**

**ELEIÇÕES**

****

**logotipo do projeto**

**Índice**

1 Grupo de Trabalho 3

2 Descrição do projeto 4

3 Arquitetura da aplicação 4

4 Manual do Utilizador 4

5 Limitações e Desenvolvimentos Futuros 4

6 Conclusão 4

7 Referências 5

# Grupo de Trabalho

|  |  |
| --- | --- |
| FOTO | Número: 21606  Nome: Tiago Oliveira  Curso: LEI  Turma: C  Email : tiago66.oliveira@hotmail.com |
| FOTO | Número: 21607  Nome: Tiago Cardoso  Curso: LEI  Turma: C  Email : |

Declaração:

Os alunos identificados declaram sob compromisso de honra que o projeto final é da sua autoria com exceção dos seguintes elementos:

|  |  |
| --- | --- |
| **Descrição** | **Fonte** |
|  |  |
|  |  |
|  |  |

# Descrição do projeto

* Descrição detalhada do projeto,
* Projetos similares (apresentar hiperligações e imagens).
* Enquadramento e pertinência do projeto na disciplina

O trabalho prático, realizado no âmbito da unidade curricular de Programação Orientada a Objetos, consiste no desenvolvimento de uma aplicação em Java em que a sua finalidade é servir como gestor de eleições, ou seja, permite definir uma eleição com os candidatos e eleitores à escolha do utilizador. O trabalho foi dividido em duas partes, a parte do GUI que engloba toda a parte gráfica (menus) e a parte referente à lógica em si. Para todo desenvolvimento foi utilizado git com o repositório no github.

Em relação à parte gráfica, a “Main Class” é uma janela (JFrame) composto por três botões (JButton) e um menu (JMenuBar). As opções disponíveis do menu (JMenuItem) permitem aceder às mesmas janelas que os três botões principais e além disso permite aceder ao “Acerca De”.

Os botões principais são:

- “Configurar Eleição”: abre uma nova janela (JFrame) que permite configurar uma eleição, onde se define o período da mesma e onde aparecem os candidatos e os eleitores com a possibilidade de editar os mesmos. Aqui é possível então criar novos, editar os que já existem, apagar, guardar em ficheiro e ler do ficheiro. No menu de eleitores existe uma funcionalidade de gerar eleitores de forma aleatória mas sempre com nomes, géneros, idade e mesmo fotos de forma coerente.

- “Votar”: abre uma nova janela (JFrame) que permite realizar o voto em si, escolhendo o eleitor e inserindo a password do mesmo. Aqui é feito verificação se o eleitor em questão já realizou ou não voto na eleição ativa.

- “Resultados”: abre uma nova janela (JFrame) que permite ver os resultados da eleição em forma de texto e gráficos.

Toda esta informação é guardada em memória, maioritariamente em ArrayLists com a possibilidade de guardas estas em ficheiros ou ler do ficheiro para a memória.

# Arquitetura da aplicação

Como se encontra estruturado o código da aplicação,

Descrição geral das bibliotecas de classes (packages)

Diagrama de classes

Descrição detalhada de cada classe ou interface e justificação da sua existência:

Justificação da sua existência e a forma como foi construída (herança e/ou composição)

Descrição dos atributos

Descrição dos métodos

O projeto está dividido em 9 packages de forma a toda a informação estar dividida e organizada.

1. beanbuilder
2. candidate
3. election
4. elector
5. gui
6. multimedia
7. utils
8. enums
9. interfaces

O package “beanbuilder” é destinado a construir objetos de determinadas classes de forma aleatória. De momento só é usado para construir um Eleitor de forma aleatória e é composto por um método:

* **ElectorBean buildRandomElectorBean():** pega nas funções da classe Generate Utils e gera um eleitor de forma aleatória.

O package “candidate” é onde se encontra toda a informação dos candidatos em si. Está presente a classe “CandidateBean” que implementa *Serializable* e é composta pelos atributos:

* **String name:** guarda o nome;
* **String initials:** guarda a sigla;
* **Int votes:** guarda a quantidade de votos numa determinada eleição;
* **ImageIcon photo**: guarda a fotografia;

e pelos métodos:

* construtores, getters & setters;

e a classe “CandidateList” que implementa *FileManager e Serializable* e é composta pelo atributo:

* **ArrayList<CandidateBean> candidateList**;

e pelos métodos:

* construtores, getters & setters;
* **void resetAllCandidateVotes():** aquando uma nova eleição os candidatos ficam com os seus votos a 0 permitindo serem reutilizados;
* **String getGUIListLine(CandidateBean candidateBean):** retorna uma String formatada para mostrar no GUI;
* **void deleteCandidateFromList(int id):** apaga o candidato com index = “id”
* **int searchCandidateByName(String text):** procura um candidato pelo nome e retorna o seu index no caso de encontrar;
* **int searchCandidateByInitials(String text):** procura um candidato pela sigla e retorna o seu index no caso de encontrar;
* **void save(String nomeFicheiro):** guarda o conteúdo da lista num ficheiro;
* **void load(String nomeFicheiro):** lê o conteúdo de um ficheiro e passa-o para a lista

O package “election” é onde se encontra toda a informação das eleições. Está presente a classe ElectionBean que implementa *Serializable* e é composta pelos atributos:

* **String name:** guarda o nome
* **ArrayList<ElectorBean> electorList**: guarda a lista de eleitores;
* **ArrayList<CandidateBean> candidateList**: guarda a lista de candidatos;
* **LocalDate startDate**: guarda a data de Inicio;
* **LocalDate endDate**: guarda a data de Fim;
* **boolean started**: guarda a verificação se já foi iniciada;

e pelos métodos:

* construtores, getters & setters;

e a classe ElectionManager que implementa *FileManager e Serializable* e é composta pelo atributo:

* ElectionBean election;

e pelos métodos:

* **void addBlankCandidate():** adiciona o candidato fictício “Votar em Branco”
* **newElection():** reinicia os votos dos candidatos e os eleitores para estes poderem ser reutilizados em novas eleições;
* **updateBeanLists():** atualiza a lista de candidatos e eleitores
* **void save(String nomeFicheiro):** guarda o conteúdo da lista num ficheiro;
* **void load(String nomeFicheiro):** lê o conteúdo de um ficheiro e passa-o para a lista

O package “elector” é onde se encontra toda a informação dos eleitores. Está presente a classe ElectorBean que implemente *Serializable* e é composta pelos atributos:

* **String name:** guarda o nome;
* **int CC:** guarda o CC;
* **char gender**: guarda o género (M ou F);
* **LocalDate birthDate**: guarda data de nascimento;
* **String password**: guarda password;
* **boolean voted**: guarda verificação se já votou numa determinada eleição;
* **CandidateBean votedCandidate**: guarda em que candidato votou numa determinada eleição;
* **ImageIcon photo**: guarda a foto;

e pelos métodos:

* construtores, getters & setters;

e a classe ElectorList que implementa *FileManager e Serializable* e é composta pelo atributo:

* ArrayList<ElectorBean> electorList;

e pelos métodos:

* **void resetElectorsVoted():** reinicia os eleitores de forma a poderem ser reutilizados numa nova eleição;
* **void orderArrayListByCC():** ordena eleitores pelo CC
* **String getGUIListLine(ElectorBean electorBean):** retorna uma String formatada do eleitor;
* **void deleteElectorFromList(int id):** apaga o eleitor com index “id”
* **int searchElectorByName(String text):** procura um eleitor pelo nome e retorna o seu index no caso de encontrar;
* **int searchElectorByCC(String text):** procura um eleitor pelo CC e retorna o seu index no caso de encontrar;
* **void save(String nomeFicheiro):** guarda o conteúdo da lista num ficheiro;
* **void load(String nomeFicheiro):** lê o conteúdo de um ficheiro e passa-o para a lista

O package “gui” é onde se encontra todos os menos (JFrames e JDialogs) da aplicação. Encontra-se a GUIAbout que é um pequeno JDialog composto por uma TextArea que contém uma pequena descrição da aplicação.

Encontra-se a GUICanditate que é JDialog composto alguns JButton’s, JList’s, TextField’s eJLabels e pelos seguintes métodos:

* **void updateGUIList():** atualiza a lista de candidatos;
* **ActionEvent Button Close:** faz o “dispose” da janela;
* **ActionEvent Button New:** cria um novo candidato por defeito e põe-no selecionado na lista;
* **ActionEvent Button Delete:** apaga o candidato selecionado da lista;
* **ListSelectionEvent:** verifica quando é mudado o candidato selecionado e preenche o formulário com a sua informação;
* **ActionEvent Button Select Last:** seleciona o último elemento da lista;
* **ActionEvent Button Select First:** seleciona o primeiro elemento da lista;
* **ActionEvent Button Select Prev:** seleciona o elemento anterior ao selecionado;
* **ActionEvent Button Select Next:** seleciona o próximo elemento;
* **ActionEvent Button Save:** guarda a lista de candidatos num ficheiro;
* **ActionEvent Button Open:** lê candidatos de um ficheiro;
* **ActionEvent Button New List:** apaga todos os elementos da lista;
* **ActionEvent Button Search:** procura candidatos por nome ou sigla;
* **KeyEvent Textfield Name e Initials:** quando é alterado o conteúdo das textfields, automaticamente edita o candidato.

Encontra-se o GUIConfig que éum JFrame composto alguns JButton’s, JList’s, TextField’s eJLabels e pelos seguintes métodos:

* **void updateGUIList():** atualiza a lista de candidatos e eleitores;
* **ActionEvent Button Close:** faz o “dispose” da janela;
* **ActionEvent Button New:** cria uma nova eleição por defeito;
* **ActionEvent Button Save:** guarda a eleição num ficheiro;
* **ActionEvent Button OpenElection:** lê eleição de um ficheiro;
* **ActionEvent Button OpenCandidate:** lê candidatos de um ficheiro;
* **ActionEvent Button OpenElector:** lê eleitores de um ficheiro;
* **ActionEvent Button StartElection:** começa uma eleição;
* **ActionEvent Button OpenMenuCandidate:** abre GUICandidate
* **ActionEvent Button OpenMenuElector:** abre GUIElector

Encontra-se a GUIElector que é JDialog composto alguns JButton’s, JList’s, TextField’s eJLabels e pelos seguintes métodos:

* **void updateGUIList():** atualiza a lista de eleitores;
* **ActionEvent Button Close:** faz o “dispose” da janela;
* **ActionEvent Button New:** cria um novo eleitor por defeito e põe-no selecionado na lista;
* **ActionEvent Button Delete:** apaga o eleitor selecionado da lista;
* **ListSelectionEvent:** verifica quando é mudado o eleitor selecionado e preenche o formulário com a sua informação;
* **ActionEvent Button Select Last:** seleciona o último elemento da lista;
* **ActionEvent Button Select First:** seleciona o primeiro elemento da lista;
* **ActionEvent Button Select Prev:** seleciona o elemento anterior ao selecionado;
* **ActionEvent Button Select Next:** seleciona o próximo elemento;
* **ActionEvent Button Save:** guarda a lista de eleitores num ficheiro;
* **ActionEvent Button Open:** lê eleitores de um ficheiro;
* **ActionEvent Button New List:** apaga todos os elementos da lista;
* **ActionEvent Button Search:** procura eleitores por nome ou sigla;
* **KeyEvent Textfield Name e Initials:** quando é alterado o conteúdo das textfields, automaticamente edita o eleitor.

Encontra-se a GUIMainMenu que é JFrame composto alguns JButton’s, MenuBar e pelos seguintes métodos:

* **ActionEvent Button Votar:** abre GUIVote;
* **ActionEvent Button Config:** abre GUIConfig;
* **ActionEvent Button Results:** abre GUIResults;
* **ActionEvent Button About:** abre GUIAbout;

Encontra-se a GUIResults que é JDialog composto alguns JButton’s, TabbedMenus e pelos seguintes métodos:

* **ActionEvent Button Graficos:** abre os resultos sob a forma de gráficos;
* **updateGUILists():** atualiza todas as listas;
* **ActionEvent Button Abrir:** permite ler uma eleição;
* **ActionEvent Button Search:** permite procurar eleitores;
* **ActionEvent Button Close:** faz dispose à janela;
* **GUIResults():** define todos os dados e campos referente à eleição atual;

Encontra-se a GUIResultsStatistics que é JDialog composto um gráfico JFreeChart de barras.

Encontra-se a GUIUtilizador que é JFrame composto alguns JButton’s, JList e pelos seguintes métodos:

* **GUIUtilizar():** define o eleitor atual e lista os candidatos para realizar o voto;
* **ListSelectionEvent:** verifica quando é mudado o candidato selecionado e preenche o formulário com a sua informação;
* **updateGUIList():** atualiza a lista de candidatos

Encontra-se a GUIVote que é JDialog composto alguns JButton’s, JList e pelos seguintes métodos:

* **updateGUIList() :** atualiza a lista de eleitores;
* **GUIVote():** define todos os campos referente à eleição atual;
* **ActionEvent Button Search:** procura eleitores;
* **ActionEvent Button Abrir:** permite ler uma eleição;
* **ActionEvent Button ENTER:** verifica se a password está correta e abre a janela de Voto em si;

O package “lib” é onde se encontra todas as bibliotecas externas usadas pela aplicação. Somente tem uma chamada “JFreeChart” que permite a construção de vários tipos de gráficos para estatísticas.

O package “multimedia” é onde se encontra todas as imagens usadas comi ImageIcon usadas na aplicação.

O package “utils” é onde se encontra todas as funções úteis ao funcionamento da aplicação. Está presente a classe GenerateUtils e é composta pelos atributos:

* **maxCCNumber:** define o número máximo que um nº de CC pode conter;
* **minCCNumber:** define o número mínimo que um nº de CC pode conter;

e pelos métodos:

* **FirstNamesEnum getRandomFirstName(Random rd):** retorna um nome próprio aleatório;
* **LastNamesEnum getRandomLastName(Random rd):** retorna um apelido aleatório;
* **Int getRandom8DigitNumber(Random rd):** retorna um numero aleatório de 8 dígitos;
* **LocalDate getRandomBirthDate():** retorna uma data aleatória
* **int randBetween(int start, int end):** gera um número dentro do intervalo passado por argumento;
* **ImageIcon getRandomPhoto(char gender, int idade):** obtém uma foto aleatória a partir da API fakeface de acordo com os argumentos passados;

e a classe MainUtils que é composta pelos atributos:

* **String electorFilePath:** nome por defeito do ficheiro de eleitores;
* **String candidateFilePath:** nome por defeito do ficheiro de candidatos;
* **String electionFilePath**: nome por defeito do ficheiro de eleição;
* **String blankCandidateName**: nome do candidato “Voto em Branco”;
* **DateTimeFormatter formatter:** define o formato de data utilizada em toda aplicação;
* **DefaultListModel<String> listaGUIElector:** define o modelo a utilizar nas JList de eleitores
* **DefaultListModel<String> listaGUICandidate:** define o modelo a utilizar nas JList de candidatos
* **DefaultListModel<String> listaGUIResults:** define o modelo a utilizar nas JList dos resultados

E pelos métodos:

* **getPersonAge(String date):** calcula a idade a partir de uma data;
* **ImageIcon resizeIcon(ImageIcon icon, int width, int height):** redimensiona uma imagem para o tamanho passado por argumento;
* **byte[] imageFromURLToByteArray(URL url):** obtem uma image de URL e guarda-a em array de bytes.

O package “enums” é onde se encontra todas as enums constituídos por nomes próprios, apelidos e género que são usados para gerar eleitores de forma aleatória.

O package “interface” é onde se encontra todas as interfaces da aplicação. Existe somente a FileManager que define os métodos de Save e Load.

# Manual do Utilizador

* Como se utiliza a aplicação desenvolvida.
  + Imagem e descrição de cada janela da aplicação

# Limitações e Desenvolvimentos Futuros

* Quais as limitações que a aplicação tem.
* O que ficou por fazer.

# Conclusão

* O que aprendeu a desenvolver este trabalho.
* O que aprendeu a fazer por si autonomamente.
* Situações em que precisou de ajuda e a sua resolução (agradeça aqui a quem o ajudou).

# Referências

* Livros
* Documentos
* Referência web