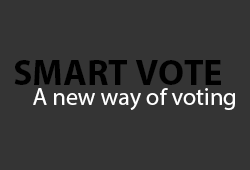


**Programação**

**Orientada a Objetos**

**ELEIÇÕES**



**Índice**

Conteúdo

[1 Grupo de Trabalho 3](#_Toc106302751)

[2 Descrição do projeto 4](#_Toc106302752)

[3 Arquitetura da aplicação 5](#_Toc106302753)

[3.1 beanbuilder 6](#_Toc106302754)

[3.2 candidate 7](#_Toc106302755)

[3.3 election 9](#_Toc106302756)

[3.4 eleitor 11](#_Toc106302757)

[3.5 gui 14](#_Toc106302758)

[3.6 lib 17](#_Toc106302759)

[3.7 multimedia 18](#_Toc106302760)

[3.8 utils 19](#_Toc106302761)

[3.9 enums 22](#_Toc106302762)

[3.10 interface 23](#_Toc106302763)

[3.11 Diagrama de classes 24](#_Toc106302764)

[4 Manual do Utilizador 25](#_Toc106302765)

[5 Limitações e Desenvolvimentos Futuros 31](#_Toc106302766)

[6 Conclusão 32](#_Toc106302767)

[7 Referências 33](#_Toc106302768)

# 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 : tiagoouriquecardoso@gmail.com |

Declaração:

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

|  |  |
| --- | --- |
| **Descrição** | **Fonte** |
| Código das aulas do professor | Aulas de POO |
|  |  |
|  |  |

# Descrição do projeto

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 ferramentos de controlo de versão, como por exemplo, o git.

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 guardar estas em ficheiros ou ler do ficheiro para a memória.

# Arquitetura da aplicação

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

1. beanbuilder
2. beans
   1. candidate
   2. election
   3. elector
3. gui
4. libs
5. multimedia
6. utils
   1. enums
   2. interfaces

Foi estruturada da seguinte forma de modo a conseguir dar uma fácil compreensão ao programador de onde tudo se situa.

## beanbuilder

O package “beanbuilder” é destinado a construir objetos de determinadas classes. De momento só é utilizado para construir um Eleitor de forma aleatória.

Métodos

|  |  |
| --- | --- |
| Métodos | Descrição |
| ElectorBean buildRandomElectorBean | pega nas funções da classe Generate Utils e gera um eleitor de forma aleatória |

## candidate

O package “candidate” é onde se encontra toda a informação dos candidatos em si.

CandidateBean

Está presente a classe “CandidateBean” que implementa *Serializable*.

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| *String name* | Guarda o nome |
| *String initials* | Guarda a sigla |
| *Int votes* | Guarda a quantidade de votos numa determinada eleição |
| Byte[] *photo* | Guarda a fotografia |

Métodos:

|  |
| --- |
| Métodos |
| Construtores |
| Getters & Setters |

CandidateList

A classe “CandidateList” que implementa *FileManager e Serializable*.

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| ArrayList<CandidateBean> candidateList | Guarda a lista de candidatos |

Métodos:

|  |  |
| --- | --- |
| Métodos | Descrição |
| 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 GU |
| 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 |

## election

O package “election” é onde se encontra toda a informação das eleições.

ElectionBean

A classe ElectionBean implementa *Serializable*.

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| 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 |

Métodos:

|  |
| --- |
| Métodos |
| Construtores |
| Getters & Setters |

ElectionManager

A classe ElectionManager que implementa *FileManager e Serializable*.

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| ElectionBean election | Guarda a eleição |

Métodos:

|  |  |
| --- | --- |
| Métodos | Descrição |
| 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 |

## eleitor

O package “elector” é onde se encontra toda a informação dos eleitores.

ElectorBean

A classe ElectorBean implementa *Serializable*.

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| 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 |
| Byte[] photo | guarda a foto |

Métodos:

|  |
| --- |
| Métodos |
| Construtores |
| Getters & Setters |

ElectorList

A classe ElectorList implementa *FileManager e Serializable*.

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| ArrayList<ElectorBean> electorList | Guarda a lista de eleitores |

Métodos:

|  |  |
| --- | --- |
| Métodos | Descrição |
| 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 texto) | 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 |

## gui

O package “gui” é onde se encontra todos os menus (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;

## lib

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.

## multimedia

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

## utils

O package “utils” é onde se encontra todas as funções úteis ao funcionamento da aplicação.

GenerateUtils

Métodos:

|  |  |
| --- | --- |
| Métodos | Descrição |
| 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 |

MainUtils

Atributos:

|  |  |
| --- | --- |
| Atributos | Descrição |
| 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 |

Métodos:

|  |  |
| --- | --- |
| Métodos | Descrição |
| 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) | converte uma imagem de um URL para array de bytes |
| byte[] iconToByteArray(ImageIcon icon) | converte um icon para array de bytes |
| ImageIcon byteArrayToIcon(byte[] byteArray) | Converte array de bytes em icon |

Constants

* **maxCCNumber:** define o número máximo que um nº de CC pode conter;
* **minCCNumber:** define o número mínimo que um nº de CC pode conter;
* **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;
* **userSystemDir:** diretório aberto por defeito no JFileChooser;
* **String blankCandidateName**: nome do candidato “Voto em Branco”;
* **gerarEleitorPopUpText:** texto do popup de quantos eleitores se pretende gerar aleatoriamente;
* **gerarEleitorPopUpTitle:** título do popup de quantos eleitores se pretende gerar aleatoriamente;
* **eliminarEleitorPopUpText:** texto do popup de confirmação de eliminação de eleitor;
* **eliminarEleitorPopUpTitle:** título do popup de confirmação de eliminação de eleitor;
* **eliminarCandidatoPopUpText**: texto do popup de confirmação de eliminação de eleitor;
* **eliminarCandidatoPopUpTitle:** título do popup de confirmação de eliminação de candidato;
* **exceptionDialogPopUpTitle:** título de popup de exception;
* **maxSizeForTextBox:** tamanho máximo de caracteres para as textbox;
* **maxSizeForSigla:** tamanho máximo de caracteres para as siglas;
* **maxSizeForCC:** tamanho máximo de caracteres para CC;
* **personResource:** caminho da imagem “person.png”;
* **blankResource:** caminho da imagem “blank.png”;
* **manuCandidatosResource:** caminho da imagem “menu\_candidates.png”;

## enums

O package “enums” é onde se encontra todas as enumerações necessárias para o código ficar mais nítido.

Ao todos temos 3 enums:

1. Errors;
2. FirstNamesEnums;
3. LastNamesEnums.

O primeiro enum “Errors” tem como função guardar todas as mensagens de erro despoletadas aquando exceptions ou validações de input.

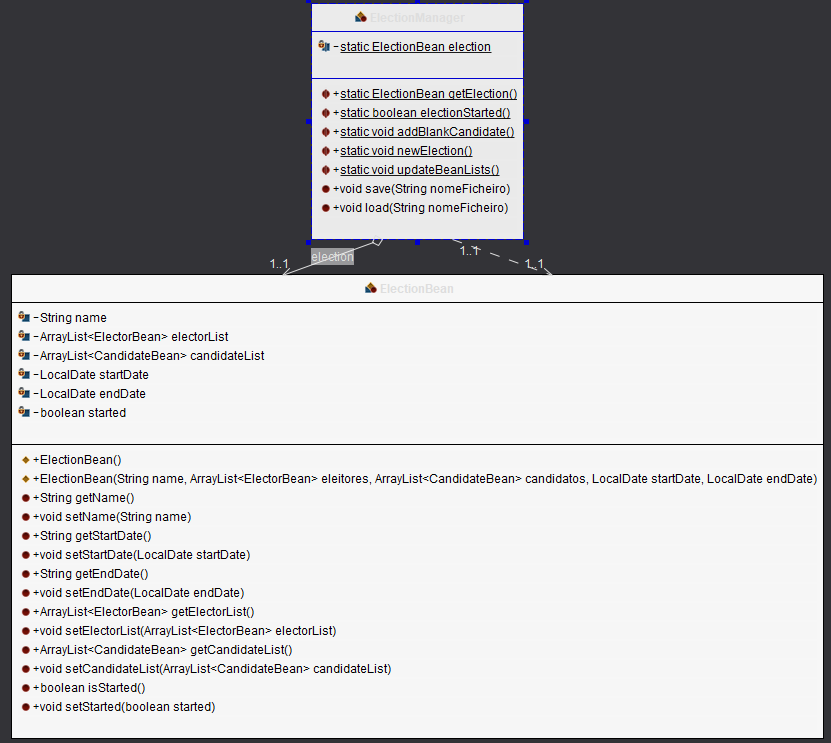
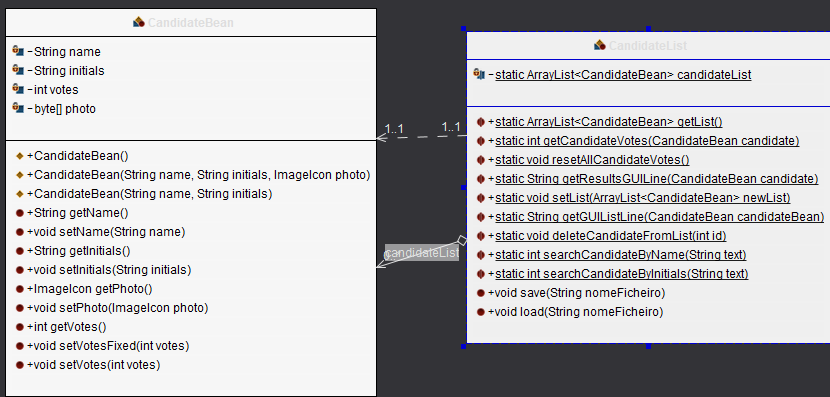
O segundo, FirstNamesEnums, tem como papel guardar uma lista de primeiros nomes de forma aquando geração de novos eleitores, ser randomizado e escolhido ao acaso a partir da lista.

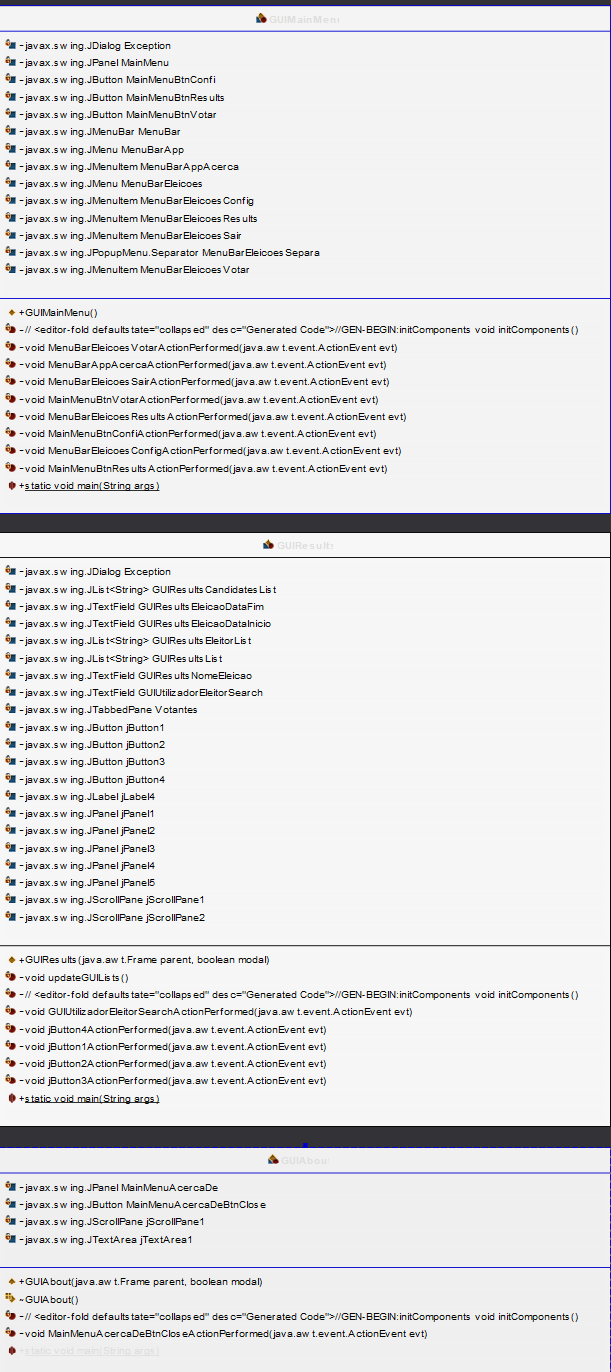
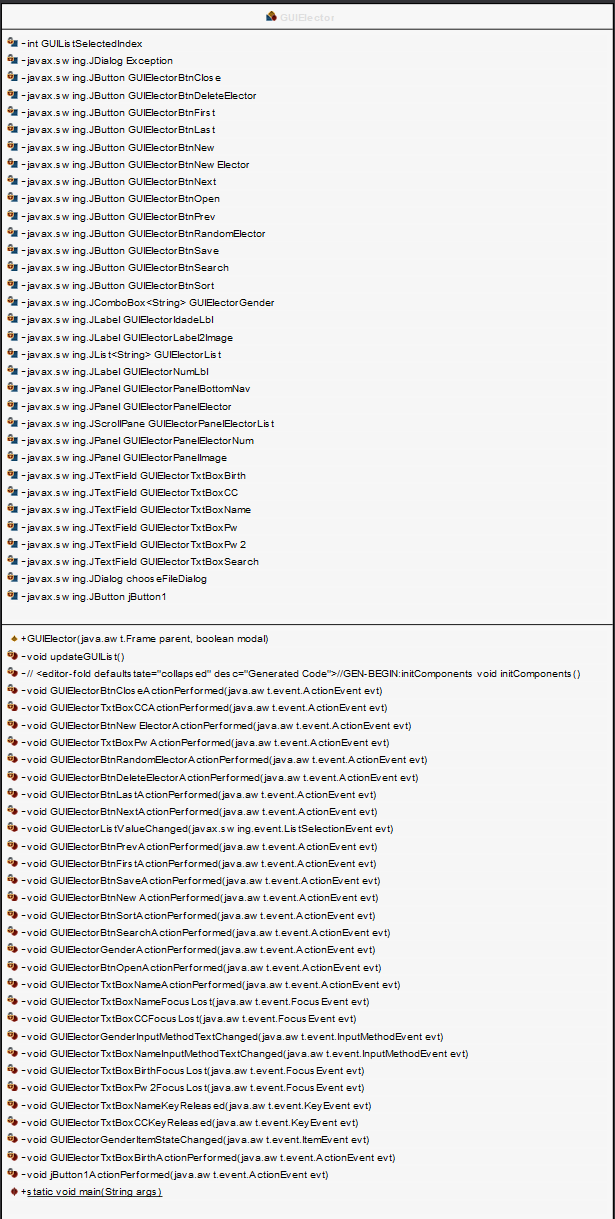
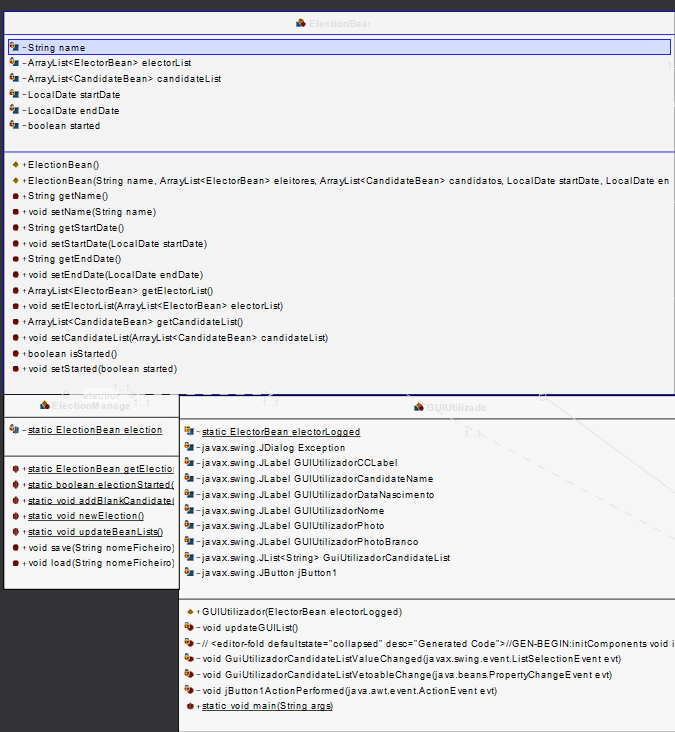
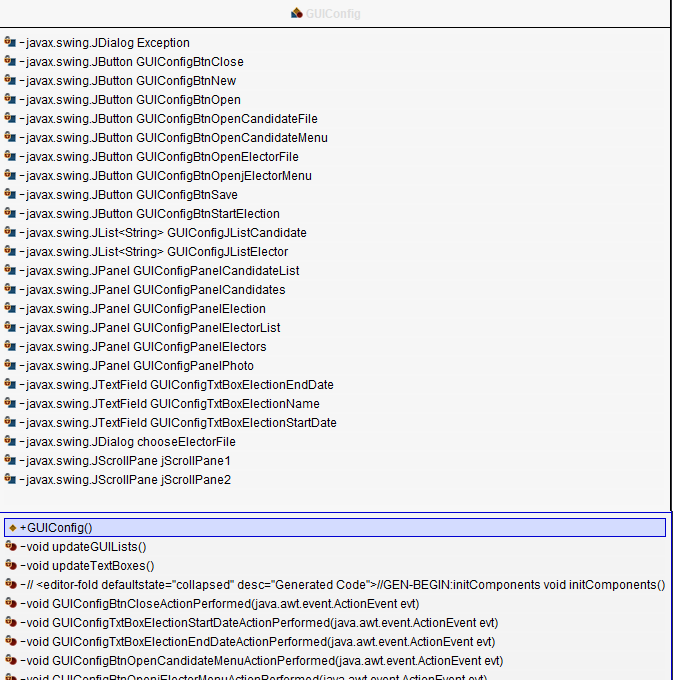
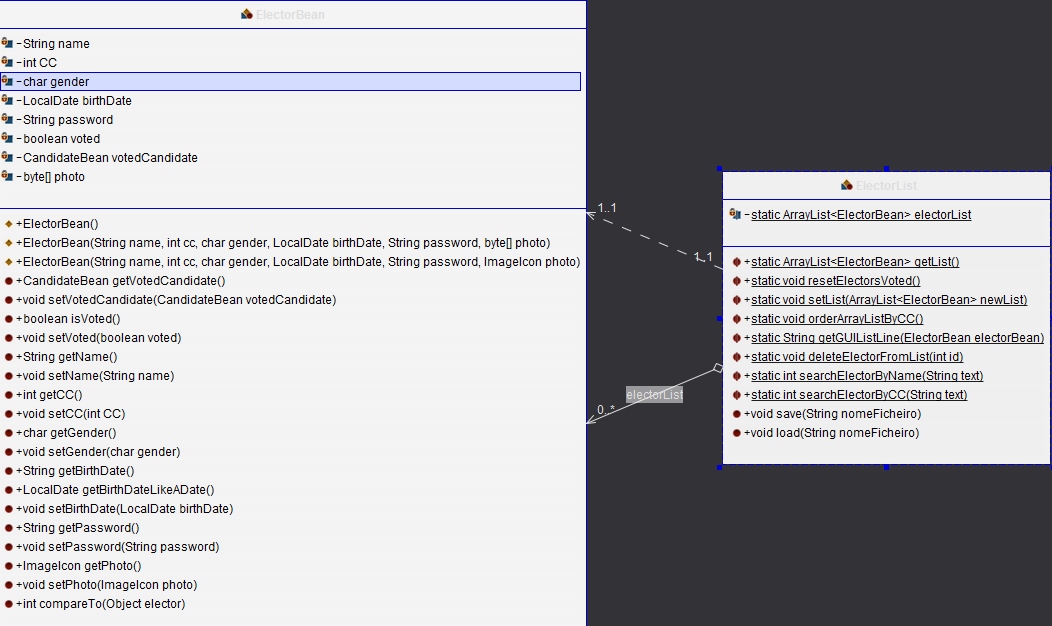
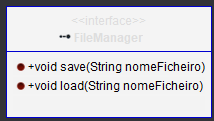
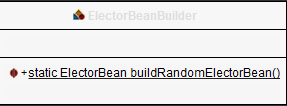
Por útlimo, o LastNamesEnums, tem a mesma explicação que o enumerador acima, a diferença, é o conteúdo que guarda, baseando-se assim em últimos nomes.

## interface

O package “interface” é onde se encontra todas as interfaces da aplicação. Neste package existe apenas uma interface com o nome de “FileManager”, cuja função é ter os métodos de save e load de ficheiros. A razão pela qual foi criada, foi devido à enorme repetição destes métodos que estava a acontecer.

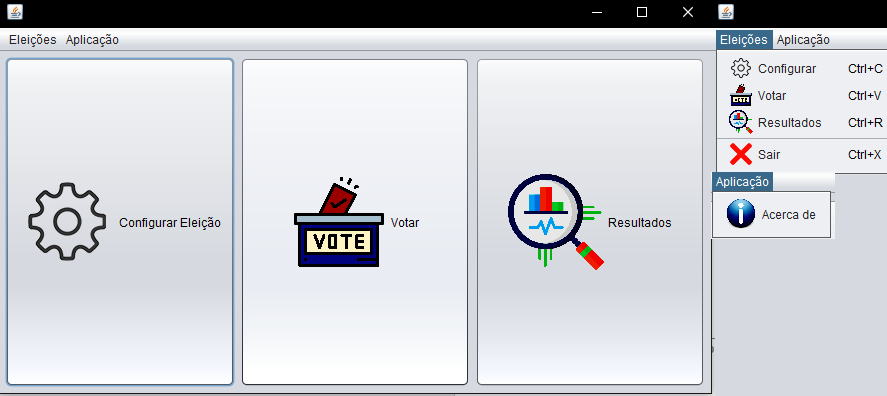
## Diagrama de classes





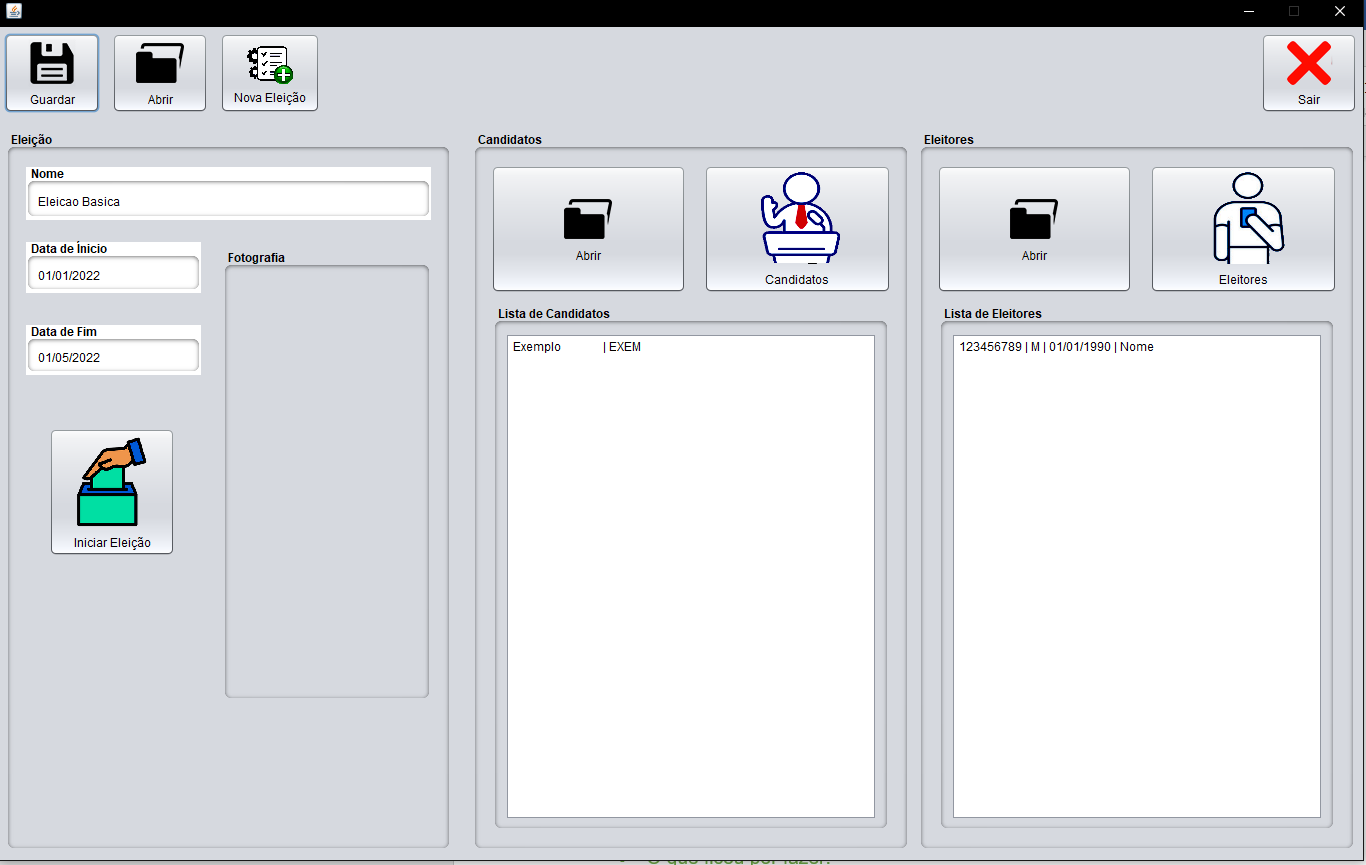
# Manual do Utilizador

GUIMainMenu



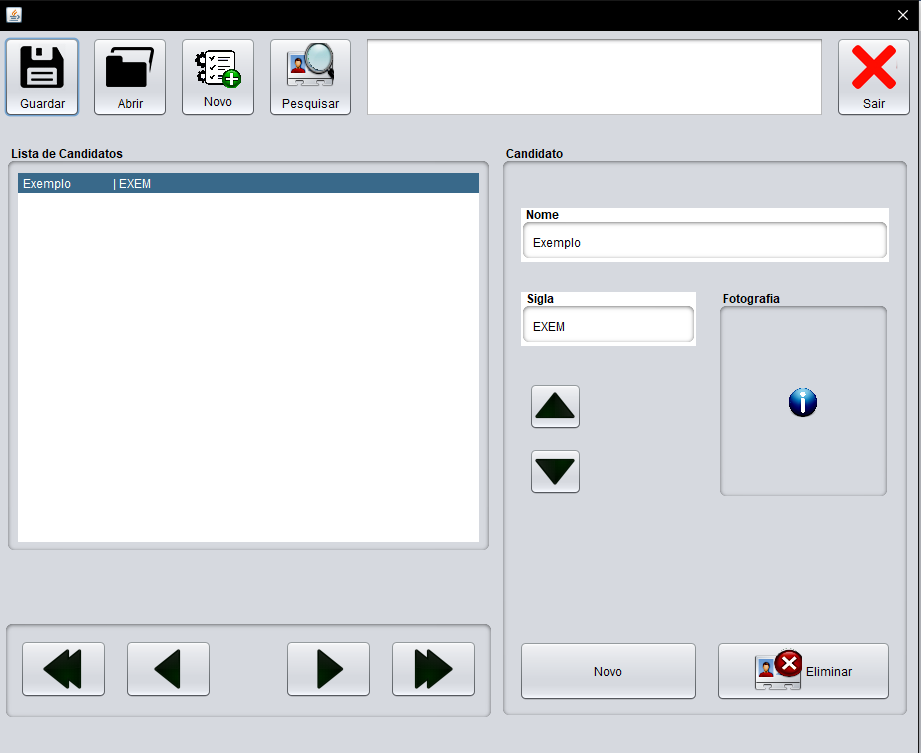
Esta é a mainclass da aplicação e é o ecrã inicial onde permite aceder a todas as restantes janelas

GUIConfig



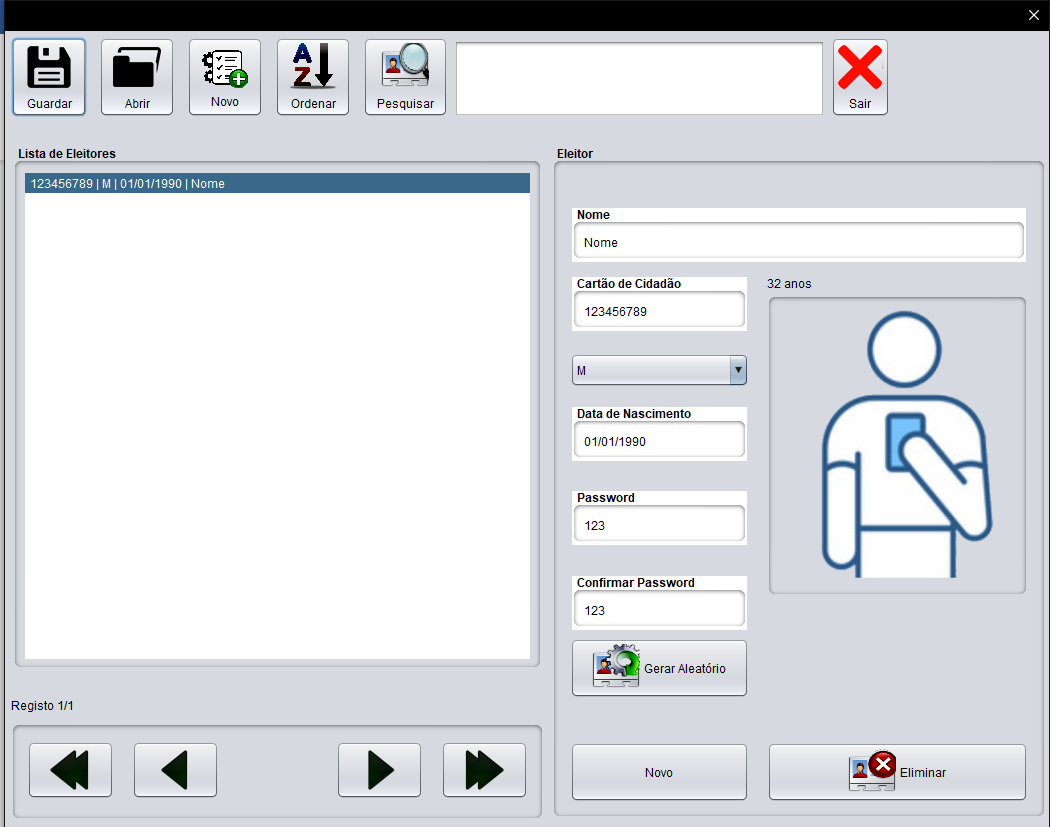
Nesta janela é onde se configura toda a eleição em si, onde se define o seu período, lista de candidatos e eleitores.

GUICandidate



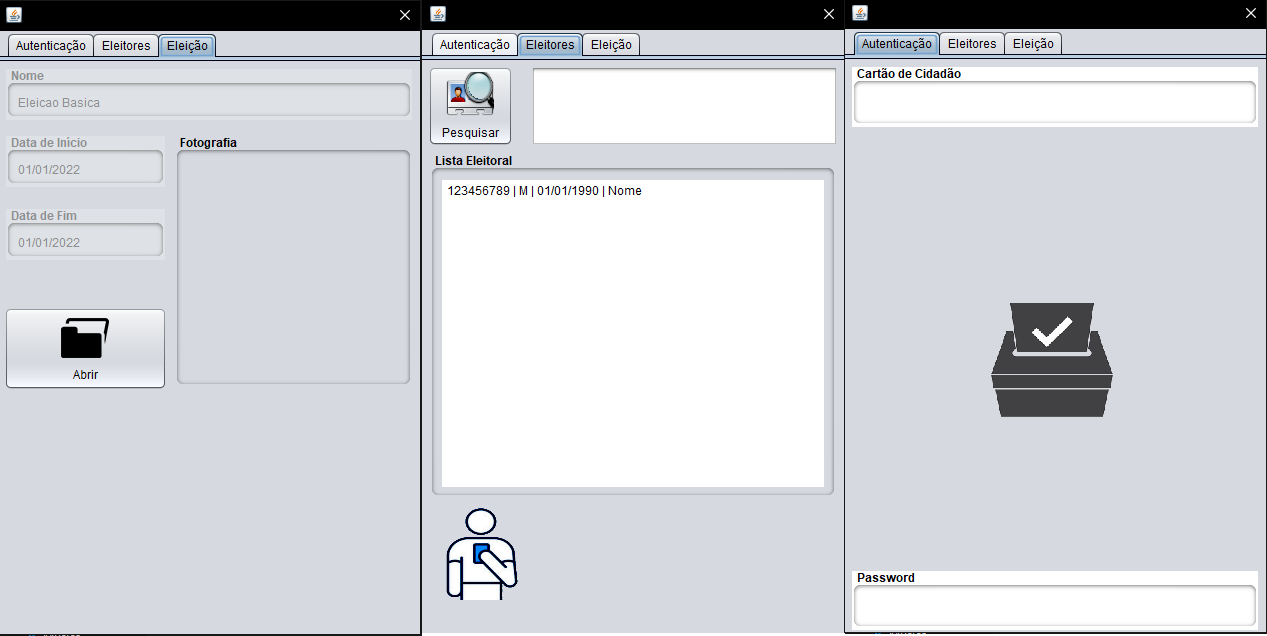
Nesta janela é onde se cria, guarda, apaga e se edita os candidatos.

GUIElector



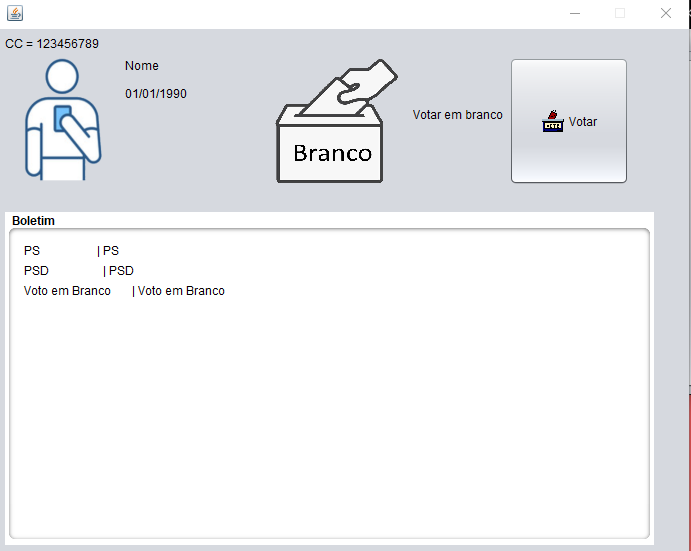
Nesta janela é onde se cria, guarda, apaga e se edita os eleitores.

GUIVote



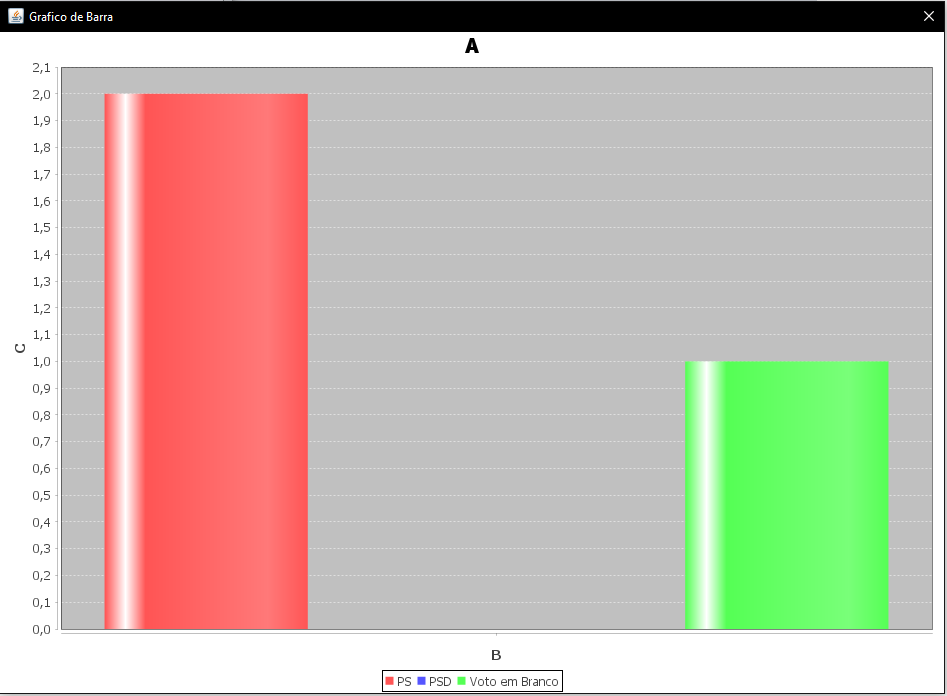
Nesta janela é possível verificar qual eleição ativa e permite abrir outra a partir de ficheiro, ver os eleitores da eleição e realizar o voto caso o eleitor em questão ainda não tenha votado.

GUIUtilizador



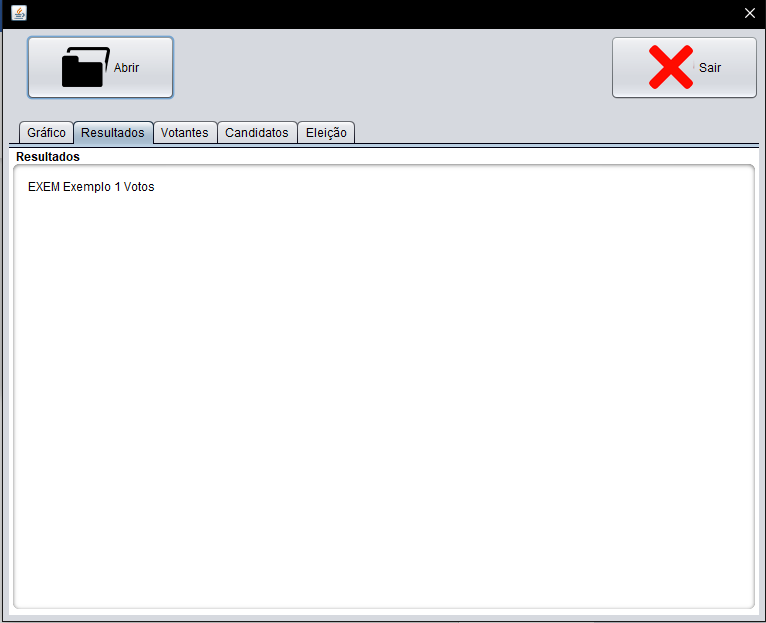
Nesta janela é listado os candidatos que é possível votar e onde é realizado o voto em si.

GUIResults

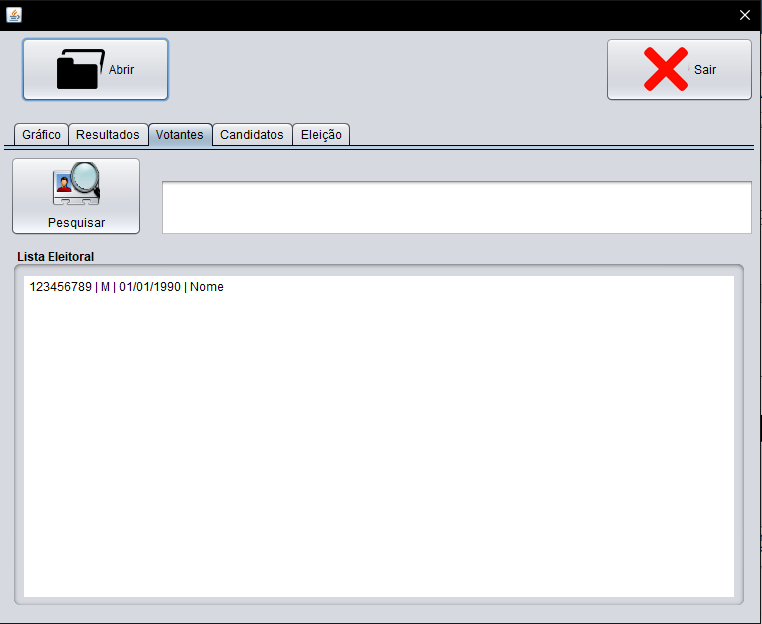


Aqui é onde apresentado os resultados da eleição atual ou permite abrir outra eleição a partir de ficheiro. No primeiro panel é possível abrir uma janela com os resultados sob a forma de gráficos de barras:

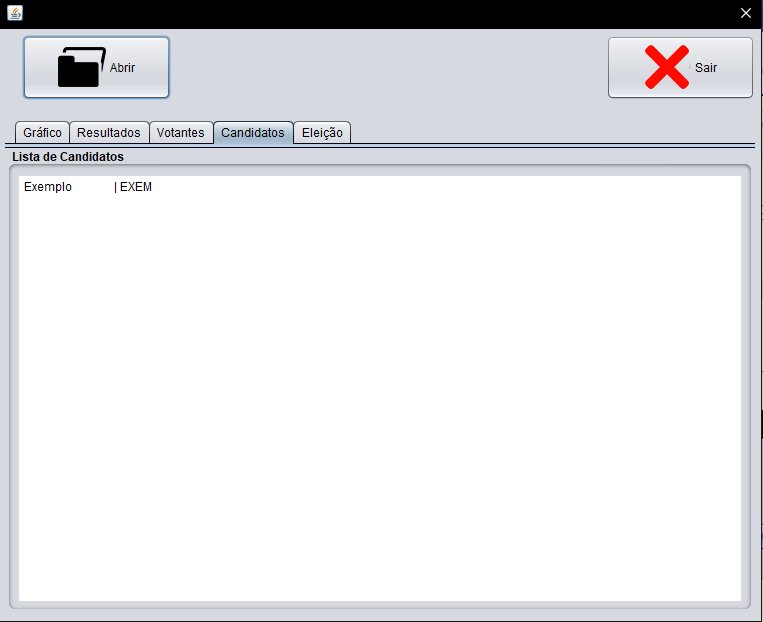
No segundo painel é listado os candidatos com o seus respetivos número de votos.



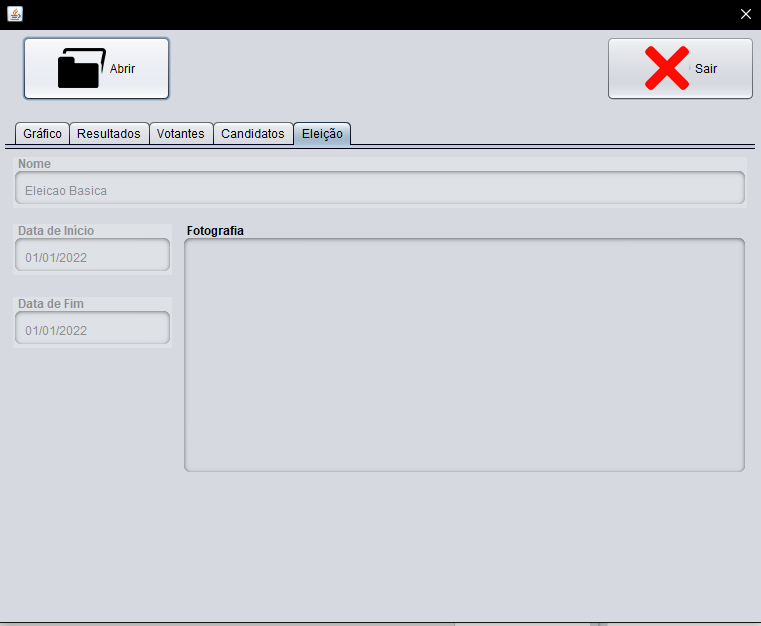
No 3º painel lista os eleitores da eleição.



No 4º painel lista os candidatos da eleição.



No 5º painel mostra informação da eleição ativa.



# Limitações e Desenvolvimentos Futuros

O programa tem um problema a gerar eleitores de forma aleatória pois a chamada à API consome muito tempo o que gera um “freeze” na aplicação enquanto o processo não termina por completo.

Além disso os ficheiros onde fica guardada toda a informação ficam demasiado grandes. Por exemplo um ficheiro de 500 eleitores pode ocupar cerca de 220MB no disco.

Em desenvolvimentos futuros estas 2 limitações deveriam ser corrigidas ou usar outras alternativas.

# Conclusão

O desenvolvimento desta aplicação permitiu pôr em prática todos os conhecimentos da disciplina e ainda originou necessidade de procurar na internet por ajuda em certas situações o que é muito bom.

Mais especificamente aprofundou-se o conhecimento em Java Swing, enums e das interfaces. Para além disso, a fazer chamadas a uma API externa para obter as fotos, e como usar bibliotecas externas e integrar as mesmas no projeto, tendo assim que ler documentação e aprender como implementar no código.

# Referências

Slides do Professor Manso;

<https://stackoverflow.com/>