



Instituto Politécnico de Tomar

COMPUTAÇÃO DISTRIBUIDA

ENGENHARIA INFORMÁTICA

2022/2023

Sistema Eleitoral



Setembro / 2022



Índice

1	<i>Grupo de Trabalho</i>	1
2	<i>Introdução</i>	2
2.1	Estado da arte	2
2.2	Aplicações e projetos baseados em blockchain.....	2
3	<i>Descrição do projeto</i>	3
4	<i>Trabalho 1 - Blockchain</i>	4
4.1	Arquitetura da aplicação	4
4.2	SECode	5
4.2.1	Classe Candidato	5
4.2.2	Classe Eleitor	6
4.2.3	Classe Eleicao	7
4.2.4	Classe Voto.....	8
4.2.5	Classe SecurityConsole.....	9
4.2.6	Classe SEGUI.....	9
4.3	Aplicação	12
4.3.1	Interfaces e Manual do Utilizador.....	12
	<i>Trabalho 2 – Computação multitarefa</i>	16
4.4	Arquitetura da aplicação	16
4.5	SECode	17
4.5.1	Classe ThreadMineira	17
4.6	Aplicação	18
5	<i>Trabalho 3 – Computação distribuída</i>	19
5.1	Arquitetura da aplicação	19
5.2	SEGUI	21
5.2.1	Classe SEGUI.....	21
5.3	Aplicação	24
6	<i>Limitações e Desenvolvimentos Futuros</i>	27
7	<i>Conclusão</i>	28
8	<i>Referências:</i>	29
8.1	Fonte Externa.....	29
8.2	Referência web	29

Índice de Figuras

Figura 1 Guilherme.....	1
Figura 2 Ruben.....	1
Figura 3 - Blockchain aplicada a um sistema eleitoral	3
Figura 4 - Diagrama de Classes.....	4
Figura 5 - Interface inicial do programa Sistema Eleitoral.....	12
Figura 6 - Seleção de uma eleição na lista de eleições	13
Figura 7 - Formulário de votação	14
Figura 8 - Aviso da existência de uma votação ativa, na tentativa de abertura de outra	15
Figura 9 - Aviso da necessidade de login	15
Figura 10 - Aba "Resultados"	15
Figura 11 - Interface Lista de Eleições com destaque nas abas "Minerar" e "Admin"	24
Figura 12 - Interface "Minerar"	25
Figura 13 - Interface "Admin"	26

1 Grupo de Trabalho

 <i>Figura 1 Guilherme</i>	Número : 23053 Nome : Guilherme Curso : Engenharia Informática Turma : A Email : aluno23053@ipt.pt
 <i>Figura 2 Ruben</i>	Número : 20965 Nome : Ruben Anastácio Curso : Engenharia Informática Turma : A Email : aluno20965@ipt.pt

Declaração:

Os alunos declaram sob compromisso de honra que o projeto descrito neste relatório é original e da sua autoria com exceção dos seguintes elementos:

Descrição	Fonte

Assinaturas:

2 Introdução

Este relatório tem por objetivo dar a conhecer todo o trabalho desempenhado no desenvolvimento do programa Sistema Eleitoral, assim como o seu funcionamento numa perspetiva do utilizador. Visa também dar a conhecer um pouco a tecnologia usada, blockchain, e a importância de mecanismos descentralizados em alguns sistemas atualmente centralizados.

O projeto em si, nada mais é que um Sistema Eleitoral, permitindo, portanto, que determinados eleitores votem num candidato, presente numa determinada eleição enquanto esta decorre, guardando dados como a quantidade de votos adquiridos pelo candidato e a votação do eleitor. O Sistema Eleitoral faz uso das tecnologias já descritas anteriormente, ou seja, do sistema blockchain, como forma descentralizada de guardar os dados referentes a uma determinada eleição, tornando todo o processo de votos confiável e incorruptível até à data. É também utilizado um sistema de chaves assimétricas que encripta os dados referentes ao utilizador de um eleitor na aplicação.

2.1 Estado da arte

A blockchain é um sistema compartilhado e imutável que facilita o processo de registo de transações e rastreamento de ativos numa rede, esta armazena os dados em blocos de informação que são conectados uns aos outros, criando assim uma base de dados descentralizada e extremamente segura.

O primeiro uso de uma cadeia de blocos criptograficamente segura foi descrito em 1991, para implementar um sistema em que os registos de data e hora dos documentos não pudessem ser violados. No entanto a primeira rede blockchain foi definida pela primeira vez no código fonte da bitcoin.

Sendo a base tecnológica das cripto moedas, a blockchain tem recebido o interesse de bancos, empresas e organizações governamentais. Desde então, têm sido feitas várias modificações a partir da versão original.

Em 2014, surgiu o termo “Blockchain 2.0”, o Ethereum, um novo conceito que permite que os utilizadores escrevam contratos inteligentes mais sofisticados.

2.2 Aplicações e projetos baseados em blockchain

- Moedas digitais
- Arte e colecionáveis (NFTS)
- DeFi

3 Descrição do projeto

- O projeto consiste num sistema de votações, que permite ao utilizador (eleitor) votar num dos candidatos disponíveis, numa eleição válida. O voto será processado a partir da blockchain.
- O eleitor tem de criar uma conta de utilizador e estar conectado à mesma para proceder ao voto. Todo o sistema de gerenciamento de contas utiliza criptografia de chaves assimétricas, de forma a tornar os dados de cada utilizador seguros.
- O programa é composto por um sistema de registo de eleições, com uma data de início, uma data de fim, uma lista de candidatos e uma lista de eleitores.
- Um eleitor só pode votar num dos candidatos se a eleição estiver ativa e este ainda não tiver procedido ao voto. No final é possível observar o número total de votos que cada candidato obteve, acompanhado por o número total de votos nessa eleição e mostrar a cada eleitor em quem votou.
- O tema deste projeto, apesar de ser um assunto amplamente falado, não está ainda implementado, pelo menos de forma publica. No entanto existem várias propostas e artigos defensores deste sistema como uma tecnologia indispensável no futuro.

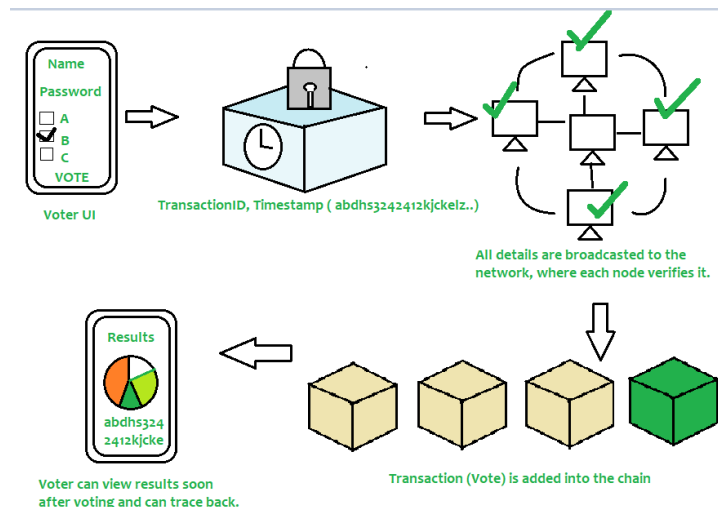


Figura 3 - Blockchain aplicada a um sistema eleitoral

4 Trabalho 1 - Blockchain

4.1 Arquitetura da aplicação

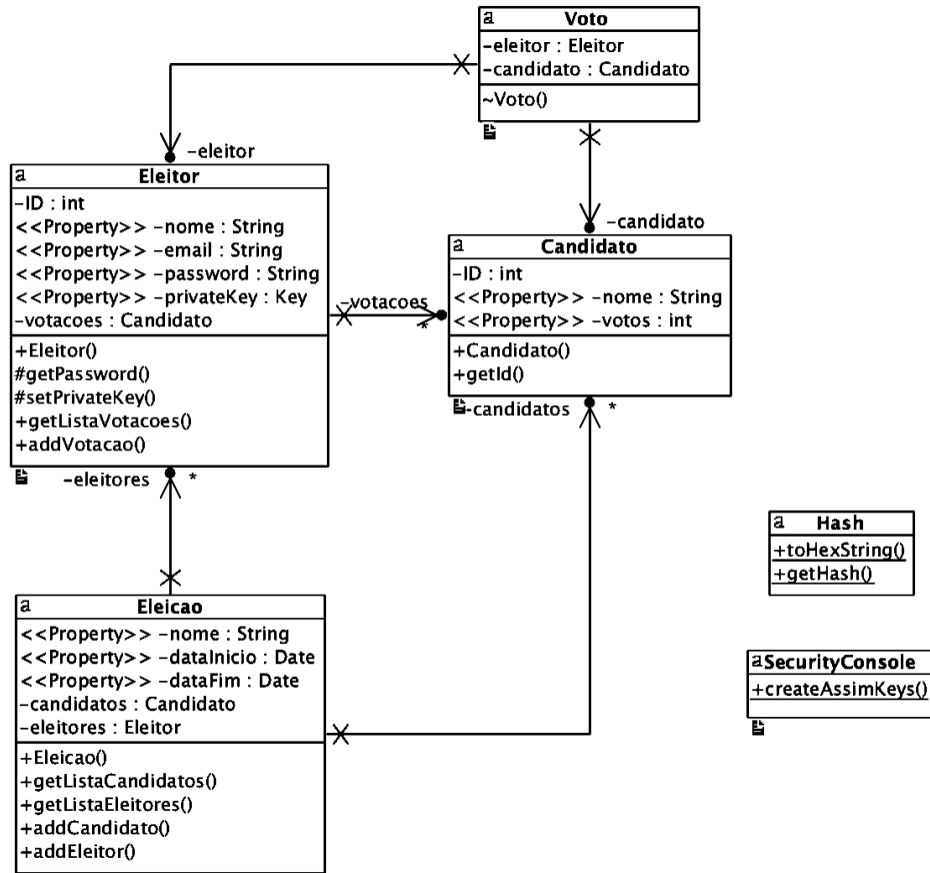


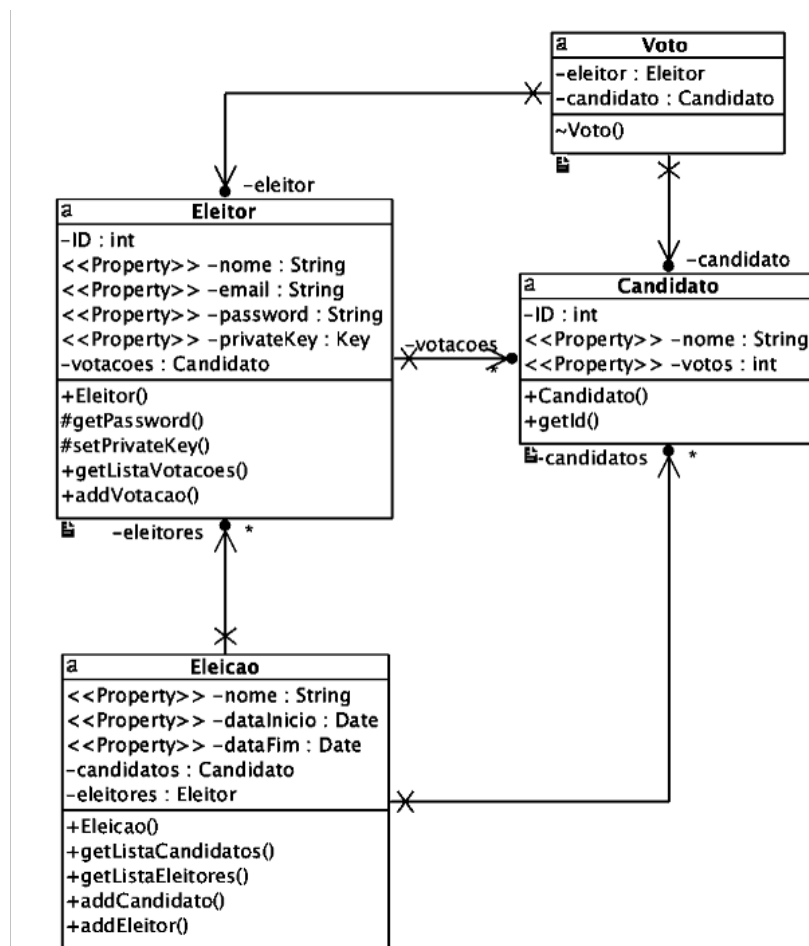
Figura 4 - Diagrama de Classes

- Descrição geral das bibliotecas
 - Recursos externos
 - SecurityUtils.java
 - Devido ao tamanho e à complexidade na criação de um ficheiro semelhante a este, usou-se o modelo disponibilizado nas aulas de forma a fazer todos os processos de encriptação dos dados de utilizador

4.2 SECode

4.2.1 Classe Candidato

- A classe Candidato tem como finalidade adquirir e guardar todos os dados de um candidato. O candidato é a pessoa ou identidade que vai ser escolhida/votada numa determinada eleição, como tal é necessário adquirir dados como o nome e o número de votos para posteriormente serem apresentados nos resultados de uma determinada eleição.
- Diagrama detalhado da classe e as ligações a outras classes

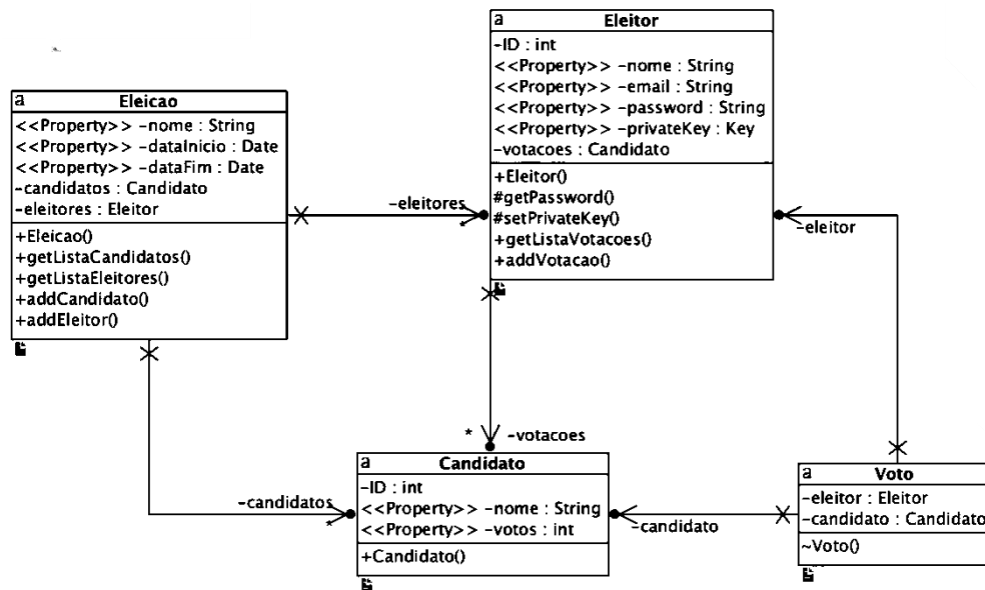


- Descrição dos atributos
 - **private int ID** Gera de forma aleatória um identificador único para cada candidato.
 - **private String nome** Guarda o nome do candidato num atributo string.
 - **private int votos** Guarda o número de votos do respetivo candidato num atributo inteiro.

4.2.2 Classe Eleitor

- A classe Eleitor assim como a classe Candidato tem como finalidade adquirir e guardar todos os dados de um eleitor. O eleitor é na prática o utilizador da aplicação, ou seja, será a identidade que vai proceder ao voto numa determinada eleição e num determinado candidato, este será obrigado a ter uma conta de utilizador na aplicação para impedir que exista mais que um voto da mesma pessoa e para permitir que este veja em que candidato votou após efetuar a votação, como tal é necessário adquirir alguns dados como o nome, o email, a password a privateKey e o voto efetuado.

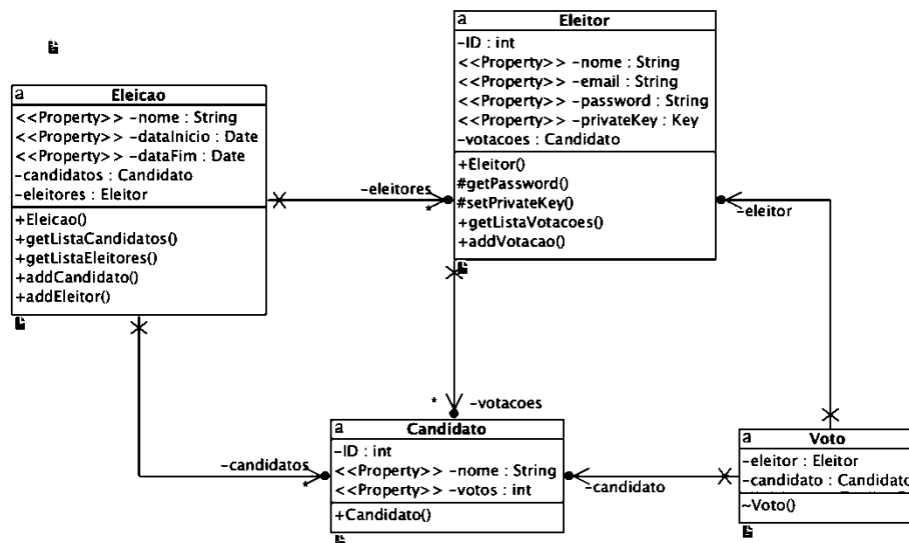
- Diagrama detalhado da classe e as ligações a outras classes



- Descrição dos atributos
 - **private int ID** Gera de forma aleatória um identificador único para cada utilizador/eleitor.
 - **private String nome** Guarda o nome do eleitor num atributo string.
 - **private String email** Guarda o email do eleitor num atributo string.
 - **private String password** Guarda a password do eleitor para posteriormente a encriptar
 - **private Key privateKey** Guarda a password encriptada do eleitor.
 - **private ArrayList<Candidato> votacoes**
- Descrição dos métodos
 - **void addVotacao(Candidato candidato)** adiciona a votação de um determinado eleitor ao candidato referente.

4.2.3 Classe Eleicao

- A classe Eleicao tem como objetivo armazenar e adquirir todos os dados que se considerem uteis para criar uma eleição. Para isso faz sentido ter um nome, uma data de início e uma de fim, ter uma lista de candidatos e ter uma lista de eleitores para procederem ao voto.
- Diagrama detalhado da classe e as ligações a outras classes

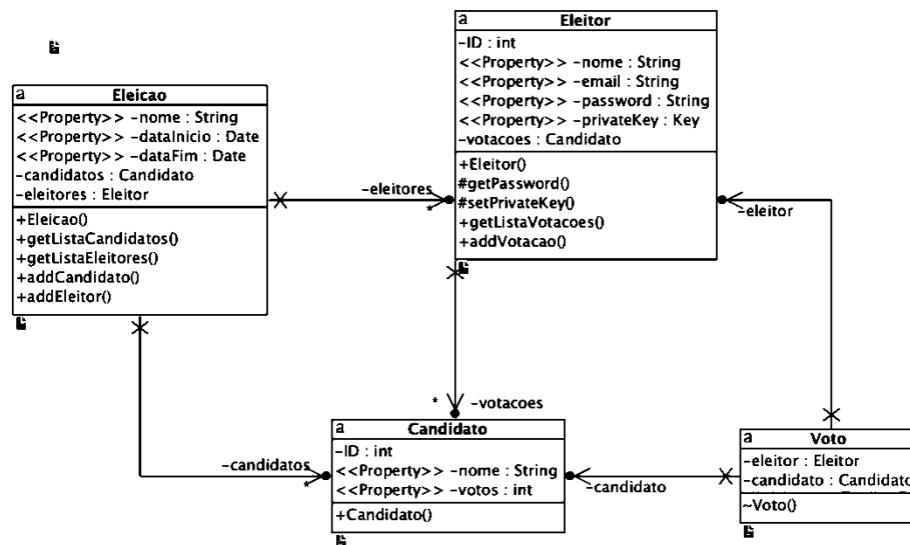


- Descrição dos atributos

- **private String nome** Guarda o nome da eleição num atributo string.
- **private Date dataInicio** Guarda o começo do período de votações num atributo do tipo date de forma a permitir que o programa controle a partir de quando os eleitores podem começar a votar em determinada eleição.
- **private Date dataFim** Guarda o fim do período de votações num atributo do tipo date de forma a permitir que o programa controle quando deixam os eleitores de poder votar em determinada eleição.
- **private ArrayList<Candidato> candidatos** Guarda os candidatos referentes a determinada eleição.
- **private ArrayList<Eleitor> eleitores** Guarda os eleitores que votaram na eleição.

4.2.4 Classe Voto

- A classe Voto tem como objetivo armazenar o voto atribuído por cada eleitor a um candidato.
- Diagrama detalhado da classe e as ligações a outras classes



- Descrição dos atributos
 - **private Eleitor eleitor** Guarda o eleitor autor do voto.
 - **private Candidato candidato** Guarda o candidato votado.

4.2.5 Classe SecurityConsole

- A classe SecurityConsole cria as chaves assimétricas de forma a encriptar e proteger a password da conta de cada eleitor.
- Diagrama detalhado da classe e as ligações a outras classes



- Descrição dos métodos
 - **void createAssimKeys(Eleitor eleitor)** é um método que encripta as credenciais dos eleitores após o seu registo, este cria 2 ficheiros, um que contém a chave pública e outro com a chave privada.

4.2.6 Classe SEGUI

- A classe SEGUI nada mais é que o centro de toda a interface da aplicação, esta é responsável por mostrar todos os elementos da interface e conter os métodos de interação.
- Descrição dos métodos
 - **void addLista()** é um método executado assim que o programa é iniciado e que tem como objetivo adicionar o nome de todas as eleições presentes no ArrayList<Eleicao> à lista da interface inicial com nome listEleicoes.
 - **void btnEntrarActionPerformed(java.awt.event.ActionEvent evt)** permite que um utilizador ao utilizar o botão “Entrar” se autentique na conta através dos dados inseridos no formulário.

Esse processo é feito verificando, o arquivo com o nome do e-mail do eleitor e o "EncryptedPassword" e tentando descriptar o conteúdo do arquivo utilizando a chave privada do eleitor. Seguidamente, o código gera um hash da *password* inserida pelo utilizador e compara o hash com a *password* descriptada lida do arquivo. Se as *passwords* forem iguais, isso significa que a password inserida é a correta, e a aplicação procede à atualização do atributo “loggedEleitor” para o eleitor atualmente a ser verificado, à colocação do atributo “loggedIn” a verdadeiro e à

exibição de uma mensagem pop-up a indicar que o login foi bem-sucedido. Se as *passwords* não forem iguais, o código passa para o próximo eleitor na lista e tenta verificar a password novamente. Se o loop terminar sem encontrar um eleitor com o email e password inseridos pelo utilizador, isso significa que a autenticação falhou.

- **void btnLimparActionPerformed(java.awt.event.ActionEvent evt)** este método, procede apenas à limpeza de todos os campos do formulário, inserindo texto vazio em cada um deles. (*txtFieldNome.setText("")*)
- **void btnRegistrarActionPerformed(java.awt.event.ActionEvent evt)** tem a função de criar novos utilizadores e para isso procede à recolha dos dados presentes nos campos do formulário, isto é, *txtFieldNome*, *txtFieldEmail* e *txtFieldPassword*, produzindo um alerta na label *txtAlert* se algum destes campos ficar por preencher. Após esta recolha e validação é verificado ainda, se o email já está associado a alguma conta através de um ciclo que verifica em todos os eleitores já existentes se existe o email digitado. Após esta segunda validação, são então criadas as chaves assimétricas para o eleitor através do parâmetro *createAssimKeys(eleitor)* e adicionados todos os dados recolhidos à base de dados. No fim, procede à limpeza de todos os campos do formulário
- **void jlistEleicoesMouseClicked(java.awt.event.MouseEvent evt)** este método é responsável por criar uma nova aba no menu superior da aplicação, quando selecionada pelo utilizador uma eleição presente na lista. Começa, portanto, por fazer a validação da existência de uma aba de votações já aberta através de uma variável *boolean*, se esta se verificar verdadeira quando este método é executado, é emitido um alerta em forma de pop-up através do método *JOptionPane.showOptionDialog()* a avisar que já se encontra uma aba aberta para esse fim e que esta tem de ser fechada antes de poder abrir outra votação, dando também a hipótese de a fechar diretamente no alerta. Após esta validação é corrida outra validação que pretende verificar se o utilizador está atualmente com *login* feito, esta validação também é feita através de uma variável *boolean*, variável esta que é colocada a *true* caso exista sucesso na execução do método já referido a cima **btnEntrarActionPerformed()**, no caso de o programa não conseguir passar nesta segunda validação, é também emitido um alerta à semelhança do descrito anteriormente a pedir que o utilizador proceda ao *login*. Passadas as validações é então adicionada uma nova tab ao “menuTabs”, com o nome de

“Votação”. Esta nova tab é composta por uma lista de candidatos e um botão de validação do voto.

- **void**

jButtonVotarActionPerformed(java.awt.event.ActionEvent evt)

corresponde ao método executado após a interação com o botão votar desta nova aba criada, este começa por verificar se foi selecionado algum candidato lista de candidatos. Se nenhum tiver sido selecionado, é exibido um aviso a solicitar que seja selecionado um candidato antes de validar o voto. Se um candidato já tiver sido selecionado, o código entra no bloco do else e começa a iterar na lista de candidatos. Para cada candidato, o código verifica se o nome corresponde ao nome do candidato selecionado na lista, se os nomes corresponderem, a aplicação cria um objeto chamado “Voto” com o eleitor autenticado e o candidato selecionado e adiciona o objeto à lista de votos “votosStack”. O eleitor autenticado é também adicionado à lista de eleitores da eleição atualmente selecionada. Depois disso, a aplicação mostra uma mensagem em formato pop-up a indicar que a votação foi completa e remove a tab/aba de votação da interface do utilizador. Se o loop terminar sem encontrar nenhum candidato com o nome selecionado na lista, isso significa que ocorreu algum erro e o método não dará ordens para qualquer ação. Após este procedimento, é executado o método **fechaAbaVotar()** descrito a seguir.

- **void fechaAbaVotar()**, este método serve, como o nome indica, para fechar a aba votar criada pelo método **menuTabs.addTab("Votação", jPanelVotacoes)**. Este procede à contagem de abas existentes no programa e procura a que contenha no título a palavra “Votações”, o título é adquirido pelo método “getTitleAt()”. Se encontrar uma aba com este nome, procede à sua eliminação com o método “menuTabs.remove(i)”, e coloca a variável *boolean* que identifica se existe ou não uma aba de votação já aberta a *false* de forma a permitir que o utilizador volte a abrir outra eleição. Este método é executado após o utilizador proceder a uma votação, ou seja, após a interação com o botão “Votar” e após a interação com o botão “fechar aba”, presente no alerta dado na tentativa de abertura de nova aba de votação quando já existi uma aberta.

4.3 Aplicação

4.3.1 Interfaces e Manual do Utilizador

Ao iniciar a aplicação é apresentado ao utilizador uma interface minimalista, com 3 elementos-chave para a sua utilização (Figura 5).

A ocupar todo o lado esquerdo desta interface, está um formulário com 3 campos de texto e 3 botões de interação. Ao preencher todos os campos com os dados pessoais, o utilizador fica habilitado a criar uma conta na aplicação com o botão “Registrar”, ou se este já tiver uma conta previamente criada, poderá aceder à mesma carregando diretamente no botão “Entrar”. O botão “Limpar” serve unicamente para apagar todo o conteúdo presente nos campos de texto, de forma a permitir uma rápida mudança de utilizador ou correção de erros digitados na maioria dos campos.

Como identificado em cada *label* presente no cimo dos campos, o utilizador fica obrigado a usar um email no campo “Email” e uma password no campo “Password” caso contrário será avisado à cerca da falta de preenchimento de determinado campo no retângulo branco presente no cimo do formulário, logo após o menu de tabs.

Ao se registar, é criada imediatamente uma chave publica e uma chave privada que permite a entrada posterior na conta do utilizador.

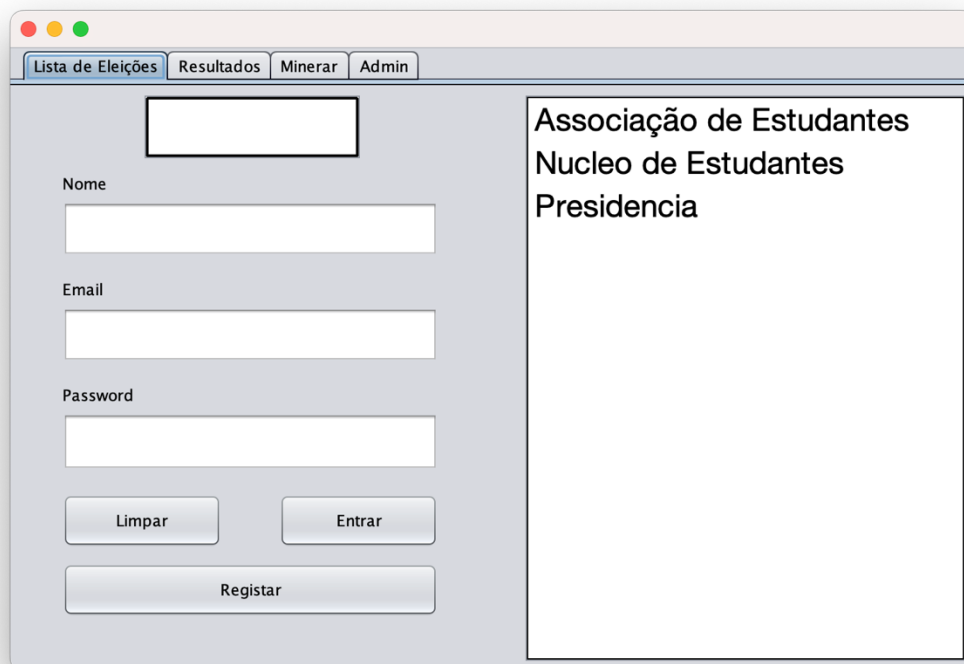


Figura 5 - Interface inicial do programa Sistema Eleitoral

Do lado direito da aplicação é apresentado uma lista de eleições a decorrer, ou seja, só serão apresentadas eleições que se encontrem dentro dos prazos de votação.

Ao se seleccionar uma eleição na lista, é criada uma aba no menu superior com o nome “Votação” como representado na Figura 6 destacado a azul. Esta aba permitirá que o utilizador proceda ao voto entre um determinado número de candidatos listados.

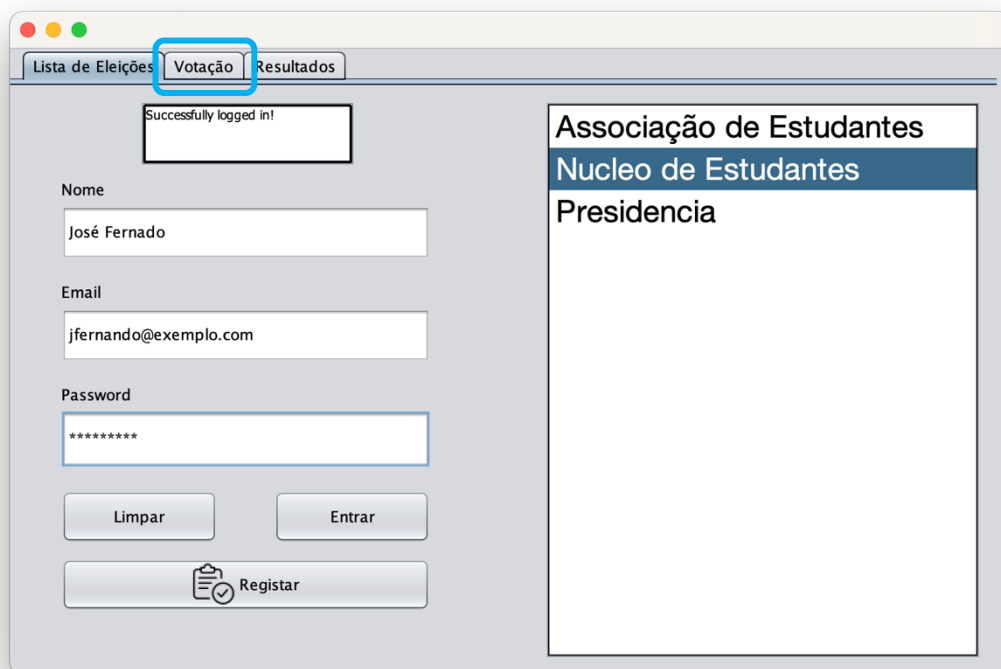


Figura 6 - Seleção de uma eleição na lista de eleições

Nesta nova aba, é disponibilizado uma lista de candidatos em que o utilizador pode votar. Para proceder à votação o utilizador deverá escolher uma opção das disponíveis e carregar no botão “Votar” para validar o voto e fechar a aba. (Figura 7)

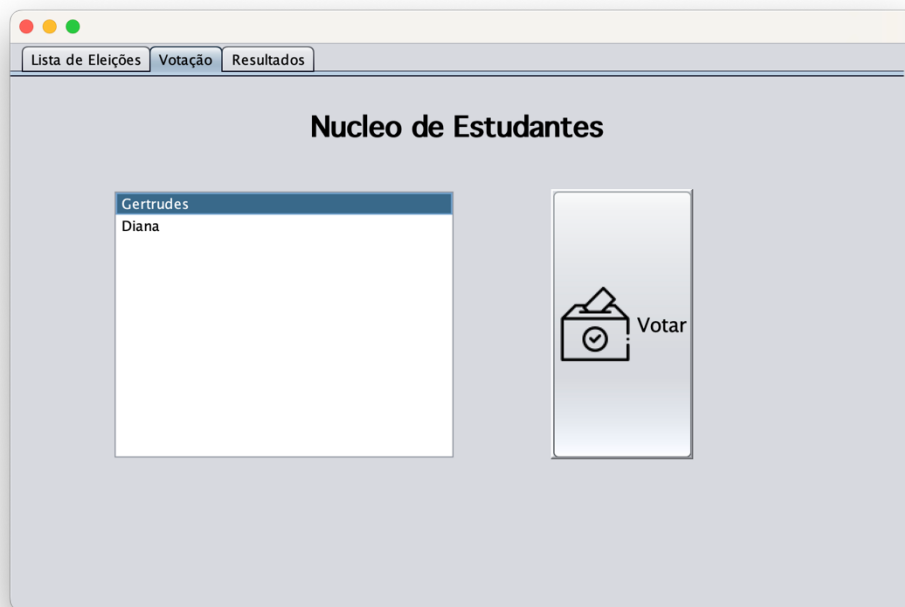


Figura 7 - Formulário de votação

Para se aceder à votação de uma das eleições presentes na lista do lado direito da interface inicial, será necessário ter o *login* efetuado numa conta, caso contrário, o utilizador será impedido de aceder à votação e receberá o aviso representado na Figura 9. A aplicação avisará e impedirá também que o utilizador tente abrir duas votações em simultâneo (Figura 8), ou seja, se este já tiver uma aba de votação aberta, não conseguirá abrir outra, mesmo que de eleições diferentes, terá de proceder à votação da eleição já aberta ou ao cancelamento da mesma, para aceder a outra, este impedimento é meramente por questões organizacionais e para evitar confusões nas votações por parte dos utilizadores.

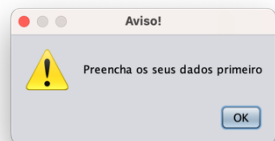


Figura 9 - Aviso da necessidade de login

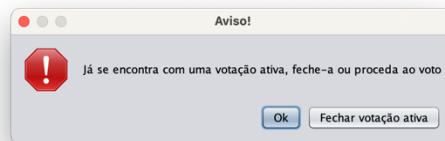


Figura 8 - Aviso da existência de uma votação ativa, na tentativa de abertura de outra

Na aba “Resultados”, é possível escolher a partir do *dropdown* a eleição que se pretende consultar os dados. Para cada eleição será apresentado no lado esquerdo o eleitor que votou e em quem votou, assim como uma lista de candidatos no lado direito correspondente à eleição selecionada.

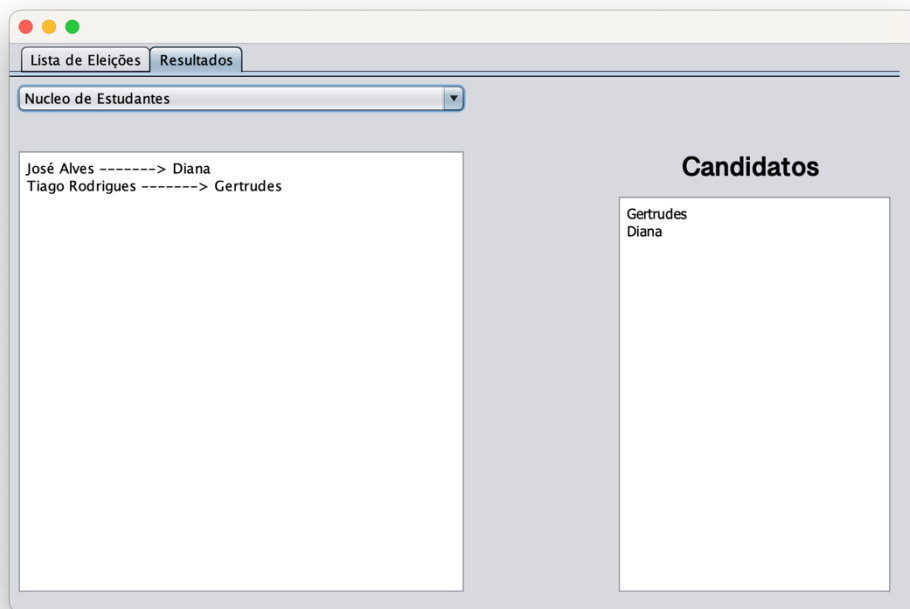
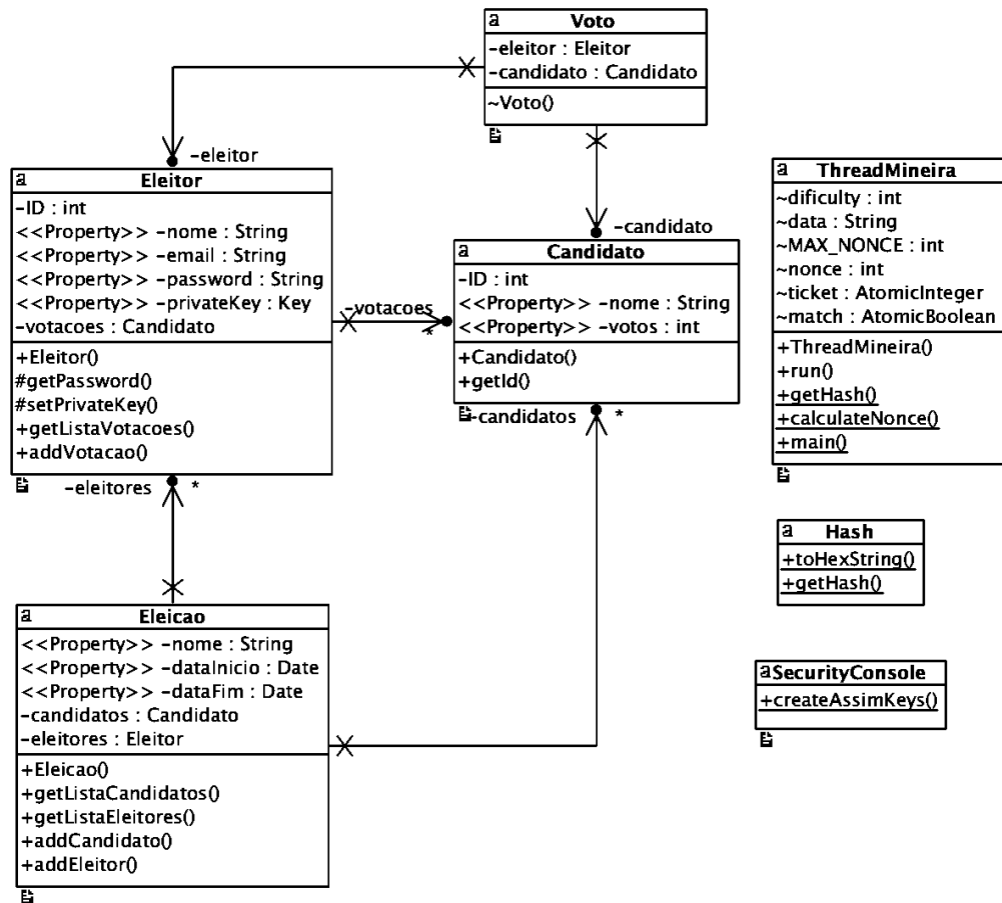


Figura 10 - Aba "Resultados"

Trabalho 2 – Computação multitarefa

4.4 Arquitetura da aplicação

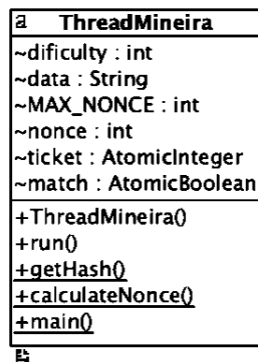
- Diagrama de classes



4.5 SECode

4.5.1 Classe ThreadMineira

- Esta Classe tem por objetivo colocar todos os recursos disponibilizados pelo dispositivo em que a aplicação se encontra em execução à procura de uma determinada *hash*. Para este processo são utilizadas as várias *threads* disponíveis a executar vários métodos em simultâneo de forma a tornar todo o processo de procura da *hash* com as especificações pretendidas, mais rápido.
- Diagrama detalhado da classe e as ligações a outras classes



- Descrição dos atributos
 - **int difficulty** define a dificuldade do processo de geração do hash.
 - **String data** adquire o conteúdo que será introduzido no bloco.
 - **int MAX_NONCE**
 - **int nonce**
- Descrição dos métodos
 - **static String getHash(String data)** este método adquire uma string e cria uma *hash* utilizando o algoritmo *SHA-256*.
 - **static String calculateNonce(int difficulty, String data, int MAX_NONCE)** através deste método é criado um *array* de *threads*, que contenha todas as *threads* disponíveis no processador no momento de execução do programa e posteriormente são colocadas inicia-as.
 - **void run()** este método serve para fazer a mineração de cada bloco, ou seja, é definida a dificuldade logo no começo do mesmo adicionando a uma variável *string* um determinado número de zeros requeridos para o início da *hash*, conforme o nível de

dificuldade presente no atributo “difficulty”. Posteriormente a partir de um ciclo é procurada uma *hash*, que comece com esse número de zeros e apenas quando esta é encontrada permite a saída deste ciclo e a passagem a outro bloco.

4.6 Aplicação

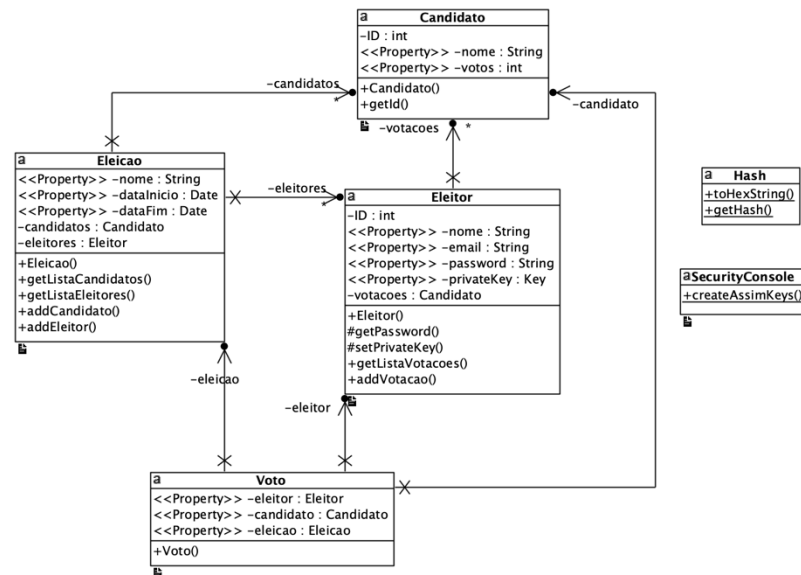
Este processo mudou apenas o forma de funcionamento do programa em termos de processamento, não tendo qualquer alteração em relação às interfaces do utilizador descritas no capítulo “4.3.1 Interfaces e Manual do Utilizador”. Tal que este tópico poderá ser consultado ainda de forma atualizada.

5 Trabalho 3 – Computação distribuída

Nesta fase do projeto foi adicionada a mineração distribuída à aplicação, tornando possível que vários computadores minerem um bloco com informações sobre os votos e as eleições.

5.1 Arquitetura da aplicação

- Diagrama de classes



Estruturado do código da aplicação:

- loggedEleitor** é um objeto da classe "Eleitor", que representa um eleitor no sistema de votação
- eleicaoSelecionada** é um objeto da classe "Eleicao", que representa uma eleição no sistema de votação.
- eleitores** e "listaEleicoes" são listas de objetos, "Eleitor" e "Eleicao" respetivamente.
- candidatos** é uma lista de objetos "Candidato".
- votosStack** é uma lista de objetos "Voto".
- tree** é um objeto da classe "MerkleTree", que representa uma árvore Merkle.
- miner** é um objeto remoto que implementa uma interface chamada "InterfaceRemoteMiner".

- Foram utilizadas 2 bibliotecas fornecidas pelo professor nas aulas.
 - A biblioteca **SecurityUtils.java** que serviu para o sistema de autenticação dos utilizadores, de forma a criar e assegurar que as contas dos utilizadores registados sejam protegidas por um sistema de chaves assimétricas. Foi autorizado o uso desta biblioteca e como tal permitiu manter um funcionamento prático e simples do código da aplicação.
 - A biblioteca **DistributedBlockchain2022** que foi usada para fazer o sistema de mineração remoto e as Merkle trees. Devido ao tamanho e à complexidade na criação de um ficheiro semelhante a este, usou-se o modelo disponibilizado nas aulas.

5.2 SEGUI

5.2.1 Classe SEGUI

- A classe SEGUI é o centro de toda a interface da aplicação, esta é responsável por mostrar todos os elementos da interface e conter os métodos de interação. Alguns métodos estão descritos no tópico 4.2.6 Classe SEGUI, como tal não foram repetidos neste tópico.
- Descrição dos atributos
 - **loggedIn** é um *boolean* que indica se há um eleitor autenticado no sistema.
 - **loggedEleitor** é um objeto “Eleitor” que representa o eleitor autenticado no sistema.
 - **eleicaoSelecionada** é um objeto “Eleicao” que representa a eleição atualmente selecionada.
 - **eleitores** é uma lista de objetos “Eleitor” que representam todos os eleitores registados no sistema.
 - **listaEleicoes** é uma lista de objetos “Eleicao” que representam todas as eleições registadas no sistema.
 - **candidatos** é uma lista de objetos “Candidato” que representam todos os candidatos registados no sistema.
 - **votosStack** é uma lista de objetos “Voto” que armazena os votos registados no sistema.
 - **votação** é um atributo do tipo *boolean* que indica se a votação está em andamento.
 - **Scanner** é um objeto “Scanner” que é utilizado para ler dados de entrada do utilizador.
 - O objeto **miner** é uma instância da classe que implementa a interface “InterfaceRemoteMiner”. Esta classe pode ser acedida remotamente através de uma rede de computadores. Isso significa que os métodos da interface “InterfaceRemoteMiner” podem ser chamados pelo objeto **miner** mesmo que o código que os implementa esteja noutra máquina.

- Descrição dos métodos
 - O método **void escondeTabs()** é executado ao iniciar a aplicação, este esconde as abas 1, 3 e 4 correspondentes à “Votação”, “Minerar” e “Admin” respectivamente.
 - **jButtonMinerarActionPerformed(java.awt.event.ActionEvent evt)** este método é executado quando o botão “jButtonMinerar” é carregado. Começando por fazer a verificação de se a mineração está atualmente em execução utilizando o retorno do método “isMining()” da classe “Miner”, se este retornar verdadeiro, ou seja se a mineração estiver em andamento, chama o método “stopMining(int)” da classe “Miner” e adiciona uma mensagem de log a indicar que a mineração foi interrompida. Se por outro lado a mineração não estiver em execução, cria uma *thread* para iniciar a mineração.

Dentro da *thread*, é adicionada uma mensagem de log a indicar o início da mineração, altera o texto do botão para “Stop” e limpa os campos de texto de *nonce* e *hash*. De seguida, chama o método “mine(String, int)” também da classe “Miner” utilizando a mensagem e o número de zeros especificados pelo utilizador como argumentos. Quando a mineração é concluída, atualiza os campos de texto com o *nonce* e o *hash* encontrado e altera o texto do botão de volta para “Start”.

- **void jButtonConectarActionPerformed(java.awt.event.ActionEvent evt)**, este método começa por obter o endereço do servidor a partir da caixa de texto “jTextFieldEnderecoServidor”, após isso, é usado o método “RMI.getRemote()” para tentar obter uma referência para um objeto remoto que implementa a interface “InterfaceRemoteMiner” no servidor especificado. Se a conexão for bem-sucedida, o objeto remoto é armazenado no atributo “miner” da classe e uma mensagem é exibida na caixa de texto “jTextPaneLogs” a indicar que a conexão foi estabelecida.
- **void btnEntrarActionPerformed(java.awt.event.ActionEvent evt)** serve para os utilizadores se autenticarem, quando autenticados, têm acesso apenas a 3 abas diferentes, “Lista de Eleições”, “Votação”, “Resultado”. Se no campo email for introduzida a palavra “admin” são apresentadas duas novas abas. Uma com o nome “Minerar” que permite a estabelecer conexão ao servidor de mineração e outra com o nome “Admin” que permite adicionar eleições e candidatos à lista de eleições.

- **void**
jButtonConectarActionPerformed(java.awt.event.ActionEvent evt) este método permite efetuar uma conexão entre uma rede de mineração e a aplicação
- **void**
jButtonAdicionarEleicaoActionPerformed(java.awt.event.ActionEvent evt) este método permite a criação de eleições, através do botão “jButtonAdicionarEleicao” presente na interface de “Admin”. O método começa por validar o preenchimento dos campos, para ver se nenhum se encontra vazio e adiciona a eleição através da classe Eleição. A esta eleição são também associados os vários candidatos descritos. No fim procede à atualização da lista de eleições para aparecer a eleição adicionada e é apresentada uma mensagem em pop-up com a mensagem de confirmação de criação da eleição.
- **void**
jButtonMerkleTreeActionPerformed(java.awt.event.ActionEvent evt) Permite, através da classe MerkleTree a criação de uma árvore de Merkle com os dados das votações, isto inclui. Candidato, eleição e eleitor.

5.3 Aplicação

Na terceira versão da aplicação foram adicionadas 2 novas interfaces à aplicação, destinadas a um utilizador específico.

Para estas aparecerem, é preciso entrar com um atualizador que contenha a palavra “admin” no *txtField* do email e ao se autenticar são adicionadas as abas “Minerar” e “Admin” ao menu.

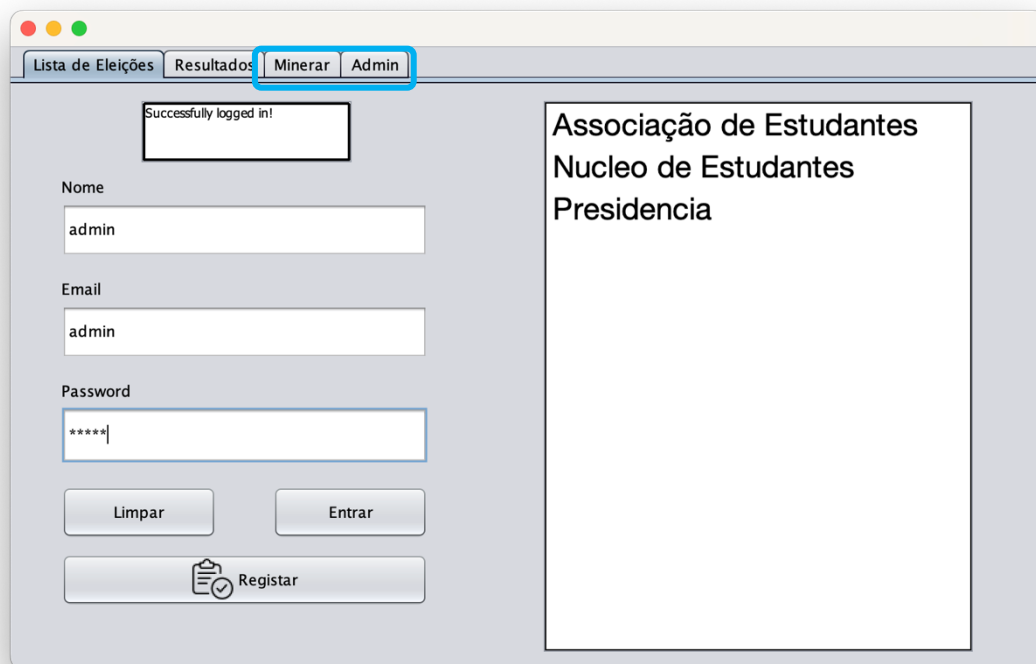


Figura 11 - Interface Lista de Eleições com destaque nas abas "Minerar" e "Admin"

Na nova aba “Minerar” é possível conectar-se a uma rede de mineração através do campo “Endereço do servidor” e seguido do uso do botão “Conectar”. Feito isso a aplicação informa se a conexão foi bem-sucedida ou não, através dos logs, como representado na figura a seguir.

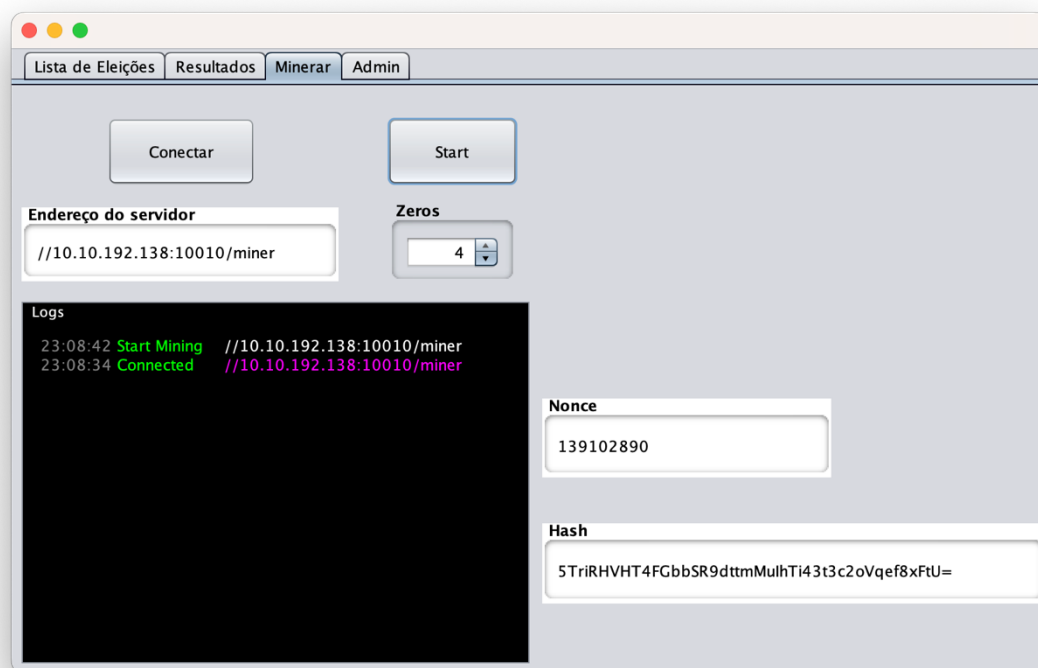


Figura 12 - Interface "Minerar"

Escolhem-se os zeros que se pretende que a hash tenha e após criar a árvore de Merkle é possível dar início à mineração, através do botão “Start”. Concluída a mineração, é transmitido através do servidor a nonce e a hash da blockchain.

Na aba/tab “Admin” é possível criar eleições para votos, para isso é atribuído um nome através do campo “Nome Eleição” disponível no formulário e posteriormente adicionados os candidatos, definindo um nome para os mesmos e adicionando-os no botão representado com um “+”, estes serão associados à eleição que se pretende criar e vistos na lista à direita da interface. Quando validada a adição da eleição através do botão “Adicionar Eleição”, esta é adicionada à lista de eleições e fica válida para receber votações.

A aplicação faz uma validação para verificar se o nome da eleição foi preenchido e se os candidatos foram adicionados. Se tudo estiver preenchido, é criada e adicionada à lista de eleições uma nova eleição.

The screenshot shows a web application window titled 'Sistema Eleitoral' with a navigation bar containing four tabs: 'Lista de Eleições', 'Resultados', 'Minerar', and 'Admin'. The 'Admin' tab is currently selected. The main content area of the 'Admin' interface includes:

- A button labeled 'Adicionar Eleição' on the left.
- A form with two input fields: 'Nome Eleição' containing the text 'Organização Cultural' and 'Candidatos' containing the text 'Alexandre'. To the right of the 'Candidatos' field is a small button with a '+' sign.
- A box on the right titled 'Candidatos Adicionados' containing a list of names: 'Maria', 'Luís', and 'Manuel'.
- At the bottom center, a button labeled 'Criar Merkle Tree' followed by a small box containing the number '0'.

Figura 13 - Interface "Admin"

Esta interface contém também o botão “Criar Merkle Tree” seguido de um número. O número representa a contagem total de votos efetuados, já o botão serve exatamente para o que o próprio nome indica, cria uma árvore de Merkle para poder se colocar posteriormente os dados adquiridos numa blockchain.

6 Limitações e Desenvolvimentos Futuros

Uma limitação que por falta de tempo permaneceu foi que não existem datas associadas às eleições, ou seja, estas permanecem por tempo indeterminado válidas para votação. A falta desta funcionalidade e de alguma menos destacáveis, devem-se à definição de prioridades no projeto, que acabou a causar falta de tempo, como tal, seriam funcionalidades que detêm todo o interesse de serem implementadas no futuro.

7 Conclusão

Contrariamente à conclusão descrita nos trabalhos 1 e 2, foi possível terminar todos os projetos.

Existem algumas funcionalidades que numa fase inicial eram pretendidas já para esta versão do projeto, no entanto não foi possível o seu desenvolvimento atempado. Isto deve-se maioritariamente ao tempo atribuído a dificuldades no projeto posteriormente ultrapassadas e também ao tempo dedicado aos estudos e desenvolvimento de projetos de outras unidades curriculares.

Muito do conteúdo utilizado neste projeto derivou da aprendizagem adquirida nas aulas, mas a sua maioria no desenvolvimento deste projeto, sendo que este obrigou de forma autónoma a compreender e saber aplicar cada tecnologia direcionada ao projeto.

Conclui-se, portanto, que este projeto se tornou indispensável no processo de aprendizagem e compreensão das tecnologias lecionadas nas aulas da unidade curricular Computação Distribuída, e contrariamente ao descrito na conclusão no entanto existiu uma grande dificuldade na consolidação do tempo dedicado a este projeto, devido ao já referido tempo dedicado a outras unidades curriculares, ao tempo necessário para desenvolver um projeto desta dimensão e à quantidade de aulas dedicada ao desenvolvimento do mesmo.

No entanto com o desenvolver do trabalho 3, foi-nos possível terminar e polir todo o trabalho que ficou por fazer no prazo de entrega do trabalho 1 e 2, e com este esforço acrescido para conseguir acompanhar os prazos e ainda completar o que tinha ficado por fazer, foi necessário não só para a nossa aprendizagem geral da unidade curricular, como para nos ensinar métodos mais eficazes de organização. Concluímos que este 3 trabalho foi o que mais nos ofereceu a nível rotineiro e de aprendizagem, visto que lhe conseguimos dedicar mais tempo, não estando completamente sobreposto com trabalhos e frequências de outras unidades curriculares como aconteceu com as primeiras fases do trabalho (trabalho 1 e 2).

8 Referências:

8.1 Fonte Externa

- [F1] Professor António Manso. "SecurityUtils.java" do projeto "Security2022".
- [F2] Professor António Manso. "SecurityUtils.java" do projeto "Security2022".

8.2 Referência web

- [W1] O que é e como funciona o blockchain: além das cripto moedas, URL: <https://distrito.me/blog/blockchain-o-que-e-como-funciona/>, Amarílis Ferreira, Consultado em 05/09/2022
- [W2] 10 projetos e tokens importantes no Ecosystema Ethereum, URL: <https://www.binance.com/pt-BR/blog/fiat/10-projetos-e-tokens-importantes-no-ecossistema-ethereum-421499824684903881>, Consultado em 06/09/2022
- [W3] Blockchain, URL: <https://pt.wikipedia.org/wiki/Blockchain>, Consultado em 06/05/2022
- [W4] Decentralized Voting System using Blockchain, URL: <https://www.geeksforgeeks.org/decentralized-voting-system-using-blockchain/>, minsuga98, Consultado em 21/11/2022