UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

DEPARTAMENTO DE INFORMÁTICA

PROGRAMAÇÃO ORIENTADA A OBJETOS – INF15933

Arthur Trarbach Sampaio

Sofia Morais Sarcinelli

Primeiro Trabalho de Implementação

Processamento de dados da Justiça Eleitoral

Vitória – ES

2023

Descrição da implementação

A Figura 1 representa o diagrama de classes UML referente à implementação das classes que foram criadas e utilizadas durante a realização desse trabalho, e que serão comentadas em sequência.

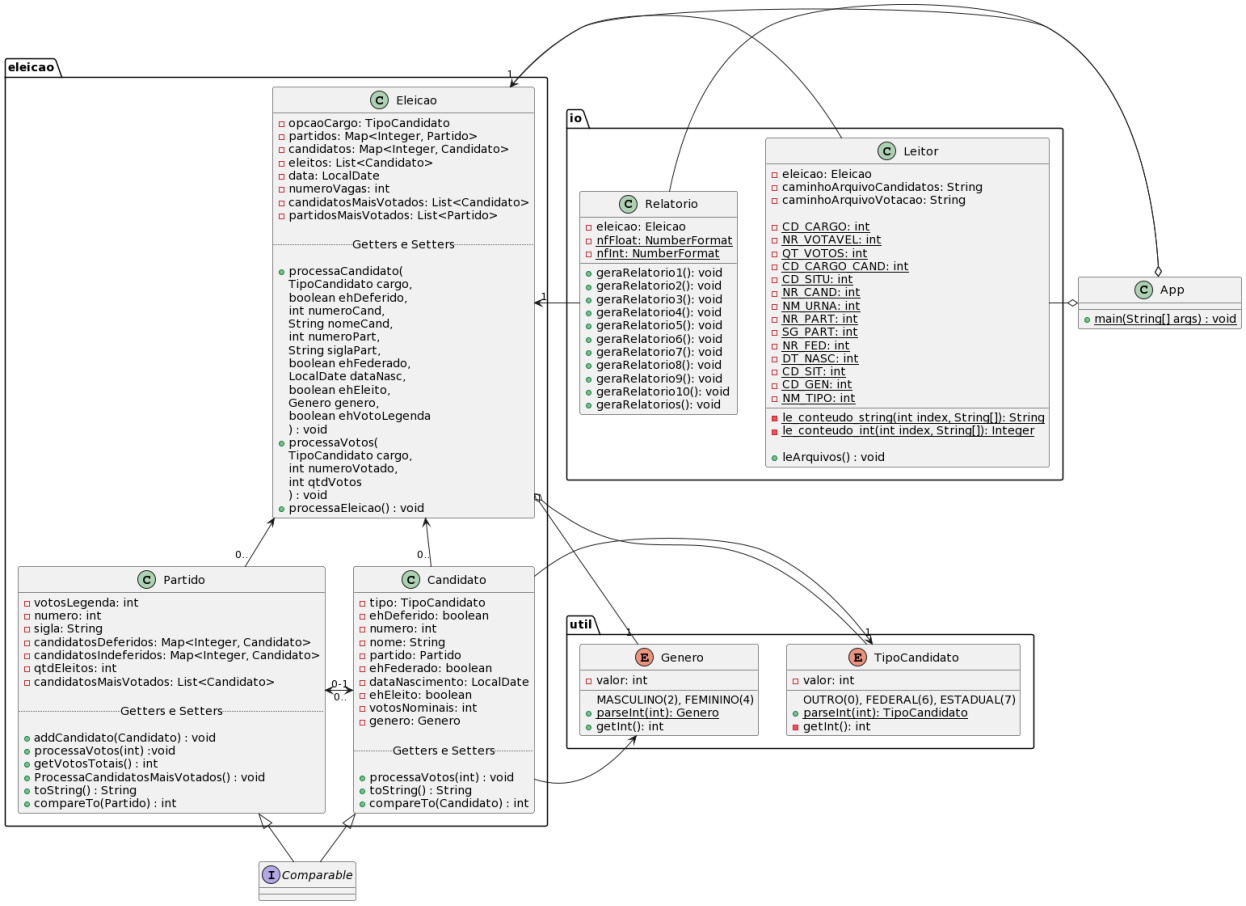

Figura 1 – Diagrama de classes UML

A principal classe implementada é a ‘Eleicao’ que é responsável por coordenar todo o processamento de dados, além do armazenamento dos partidos e candidatos lidos (realizado através de Mapas e Listas). Esta classe também é utilizada dentro das classes ‘Leitor’ e ‘Relatorio’, para que seja possível alocar e utilizar esses dados de forma coerente. Além disso, a classe ‘Eleicao’ gera e armazena algumas Listas com dados que representam os “resultados” da eleição, i.e. candidatos mais votados, partidos mais votados, candidatos eleitos e número de vagas

As classes ‘Leitor’ e ‘Relatorio’ são as classes de entrada e saída do programa, e interagem com a classe ‘Eleicao’. A classe ‘Leitor’ é responsável pela leitura dos arquivos .csv (através da utilização do ‘BufferedReader’), e formatação destes dados para que possam ser processados na ‘Eleicao’. A classe ‘Relatorio’ se responsabiliza de exibir os

dados conforme solicitado na especificação do projeto, podendo, em situações específicas, realizar ordenações para possibilitar a geração correta do relatório.

A classe ‘Candidato’ armazena e processa informações referentes à um candidato específico, interagindo com a classe ‘Partido’, com bidirecionalidade, ou seja, o candidato possui uma referência ao partido que participa, assim como o partido possui referência aos candidatos que participam da eleição.

A classe ‘Partido’, assim como a ‘Candidato’, armazena e processa informações referentes à um partido, lembrando da relação de bidirecionalidade entre ‘Candidato e ‘Partido’. Além disso, ‘Partido’ também armazena alguns resultados partidários, são eles: o número de candidatos eleitos no partido e uma lista de todos os candidatos deferidos, ordenada de forma decrescente com base no número de votos.

É válido ressaltar a implementação da interface ‘Comparable’ nas classes ‘Partido’ e ‘Candidato’, permitindo que um partido pudesse ser comparado à outros partidos, assim como candidatos pudessem ser comparados à outros candidatos. Esse passo é essencial para possibilitar a ordenação correta dos resultados da eleição.

A classe ‘App’ realiza a integração entre as classes citadas anteriormente, além de processar os argumentos de entrada do programa (opção de cargo, caminho de arquivos e data da eleição). É nesta classe em que são criadas ‘Eleicao’, ‘Leitor’ e ‘Relatorio’, conforme os parâmetros fornecidos. ‘App’ também é responsável por garantir que a eleição for devidamente processando, antes da geração dos relatórios.

Além das classes apresentadas, foram utilizadas duas enumerações (‘TipoCandidato’ e ‘Genero’), com o intuito de facilitar o entendimento do código, evitando a representação por inteiros, além de facilitar em caso de possíveis expansões do projeto, e.g. adicionar nova opção de cargo.

Todo o projeto foi realizado com o padrão de projeto ‘Factory’ em mente, ou seja, a classe ‘Eleicao’ é a responsável por criar os candidatos e partidos que serão utilizados durante o projeto, facilitando o armazenamento correto e implementação de possíveis bidirecionalidades.

Em relação às exceções, foram tratadas todas as relacionadas com leitura de arquivos, conversão de tipos de dados, valores inválidos que pudessem ser fornecidos pelo usuário. Além disso, foram tratadas as exceções em que um candidato tem data de nascimento inválida, resultando no não cadastramento do usuário no sistema. Nas seções mais internas do projeto, não foram realizados tantos tratamentos de exceções, levando em consideração que possíveis erros causados pelo usuário já haveriam sido tratados.

Descrição dos testes

Durante, e após, a implementação do projeto, foram utilizados diversos arquivos para teste, além de diferentes formas de avaliar a execução do programa (tanto para deputados federais, quanto para estaduais).

A primeira forma de teste foi o script fornecido pelo professor, que era capaz de avaliar a precisão nos estados do Acre, Alagoas, Minas Gerais, Pernambuco e Rio Grande do Sul.

A segunda forma de teste foi a comparação, dos resultados obtidos para o Espírito Santo, com o exemplo fornecido na especificação do trabalho.

A última forma de teste, foi a execução manual para cada um dos estados do Brasil, incluindo o Distrito Federal, e comparação visual com os dados apresentados no site da Justiça Eleitoral.

A Figura 2 representa os diretórios dos arquivos de teste utilizados, tantos os arquivos de candidatos, quanto os de votação.

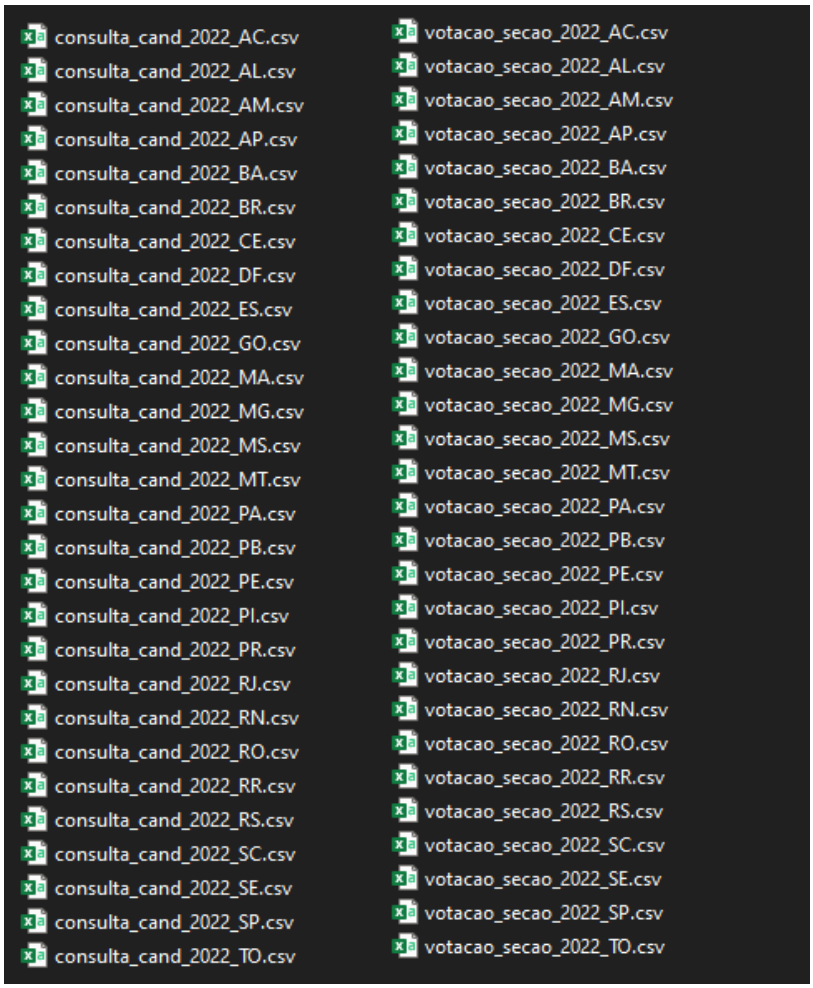

Figura 2 – Arquivos utilizados durante os testes do projeto

Bugs conhecidos

Ao final do projeto, o programa era capaz de passar tranquilamente por todos os testes realizados, contando apenas com algumas pequenas diferenças, principalmente relacionadas à contagem de votos. Esses problemas, possivelmente, ocorreram devido a formas diferentes de contabilizar votos de legenda, ou devido ao fato de o candidato não ser cadastrado, por conta de dados inconsistentes.

Em resumo, não existem bugs conhecidos que não sejam causados por erros nos arquivos de candidato ou votação, ou por inconsistências da parte do usuário durante a utilização do programa. Mesmo assim, enquanto for possível, o programa é capaz de continuar sua execução, gerando apenas possíveis desvios quantitativos nos dados.