UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA

LUIZ FELIPE MACHADO

Programação orientada a objetos:

Análise do resultado da eleição de 2022 com programação em Java.

Vitória, ES. 2022

INTRODUÇÃO

O propósito deste projeto é gerar relatórios a partir da base de dados das eleições brasileiras de 2022. Ao utilizar as informações disponibilizadas pelo Tribunal Superior Eleitoral, conduzimos análises dos votos destinados a deputados federais e estaduais, assim como dos candidatos que disputam esses cargos.

Este projeto adota a Clean Architecture como estrutura principal, com suas respectivas regras de dependências. A metodologia Clean Architecture visa organizar o código de forma a manter uma separação clara entre as camadas, promovendo a independência de frameworks e detalhes de implementação.

IMPLEMENTAÇÃO

O domínio da aplicação consiste em:

- **Entidades**: Representando as classes de domínio, como candidatos e partidos, essa camada é responsável por encapsular as regras de negócios puros, sem depender de detalhes de implementação.
- **Casos de Uso**: Nos casos de uso, localizam-se as classes que definem as operações específicas do sistema, como a geração de relatórios. Essa camada depende apenas das entidades, mantendo-se isolada das camadas externas.
- **Adaptadores**: As classes dos casos de usos não dependem diretamente de onde os dados estão vindo, mas sim de uma interface. Essas interfaces são implementadas por adaptadores definidos nas camadas mais internas. Os adaptadores lidam com a comunicação com o exterior, como a leitura de dados e a impressão na saída padrão.

O diagrama UML do projeto adota a Clean Architecture, dividindo-se em três camadas principais, cada uma identificada por uma cor específica:

Camada de Apresentação (Presentation - Azul)

- Esta camada é representada em azul e contém as classes responsáveis por toda a lógica de apresentação. Aqui, estão definidas as regras de como as informações serão mostradas no terminal. As classes desta camada interagem diretamente com as interfaces da camada de adaptadores para obter os dados a serem apresentados.

Camada de Domínio (Domain - Verde)

- A camada de domínio, destacada em verde, compreende as entidades centrais e os casos de uso do sistema. As entidades, como Candidato e Partido, encapsulam as regras de negócios, enquanto os casos de uso, implementam as operações específicas do sistema. Esta camada é independente de detalhes externos e representa o núcleo da lógica de negócios.

Camada de Adaptadores (Adapters - Vermelho)

- A camada de adaptadores, identificada em vermelho, serve como uma ponte entre o mundo externo (Arquivos) e o domínio. Nesta camada, são implementadas as interfaces definidas pelo domínio, facilitando a comunicação entre as partes internas e externas do sistema. As classes nesta camada lidam com a entrada e saída de dados, garantindo a conformidade com as regras de negócios estabelecidas na camada de domínio.

A figura 1 ilustra a separação realizada no programa, bem como as relações entre classes e entidades. Observe que a camada do domínio não depende das camadas mais externas.

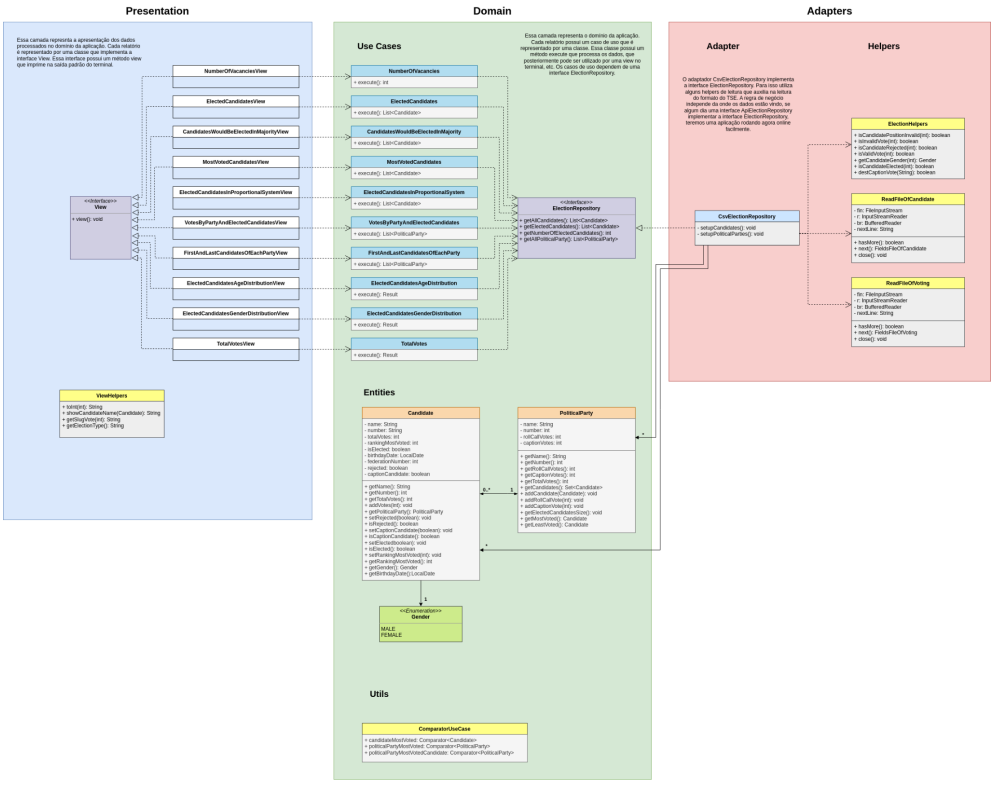

Figura 1 - Diagrama UML de classes do programa.

TESTES

O programa foi inicialmente desenvolvido para operar com os dados eleitorais do Espírito Santo. Posteriormente, foram conduzidos testes com o script fornecido, realizando adaptações conforme necessário para corrigir eventuais erros e assegurar a efetiva funcionalidade em diferentes contextos eleitorais.

Além disso, foram realizados testes em todos os estados para corrigir eventuais erros de runtime.

BUGS

Ao desenvolver o programa, observou-se algumas falhas e foram corrigidas imediatamente. Por exemplo, alguns candidatos não possuíam data de nascimento. Como não foi especificado na descrição do trabalho, assumiu-se uma data qualquer para o candidato. Isso não afetou a saída do programa, no entanto, é um ponto forte a se considerar ao encontrar uma saída inconsistente.

CONCLUSÃO

O trabalho não foi desenvolvido com uma arquitetura UML antes definida, o que tornou um pouco árduo o trabalho, visto que foi implementado uma arquitetura um pouco diferente do convencional. No entanto, é perceptível, com o diagrama UML, que o programa pode aceitar mudanças facilmente, compensando o trabalho árduo inicialmente.