Documentação de Decisões de Projeto Arquitetural e Projeto Detalhado

Integrantes

- Gabriel Victor Carvalho Rocha
- Gustavo Ribeiro Alves Rodrigues
- Misael Saraiva de Rezende
- Samuel William Almeida Santos
- Tainan Henrique de Albuquerque

Repositório no github: https://github.com/gabreuvcr/electoral-system

Atividades realizadas

Atividades
Refatoração da estrutura original
Arquitetura em 3 camadas
Padrão Singleton
Padrão Repository
Criação de aspectos
Preprocessadores
Elaboração da documentação

Padrões de Arquitetura:

1. Arquitetura em 3 Camadas:
   - A aplicação adota uma arquitetura em 3 camadas, composta por:
     - Camada de Apresentação: Representada pela classe UrnaCli, atuando como a View da aplicação.
     - Camada de Domínio: Localizada na pasta src/domain, contém a lógica principal do sistema, com destaque para a classe ElectionController, que é a principal responsável pelo controle das eleições.
     - Camada de Persistência de Dados: Implementada na pasta src/data, contém várias classes seguindo o padrão Repository, que definem contratos por meio de interfaces para as funções disponíveis e facilitam a alteração das implementações de persistência de dados.

Padrões de Projeto:

1. Singleton:
   - O padrão Singleton foi adicionado à classe ElectionController, considerando que há apenas uma eleição por execução do sistema.

Isso garante que somente uma instância dessa classe seja criada e acessível em todo o sistema, evitando a criação de múltiplas instâncias desnecessárias.

2. Repository:
    - Foi adotado o padrão Repository para a implementação da persistência de dados.
    - As classes na pasta src/data seguem esse padrão, onde cada classe representa uma entidade de dados específica e possui uma interface definindo as operações disponíveis para essa entidade.
    - Isso permite que diferentes implementações de persistência de dados possam ser facilmente alternadas, apenas modificando as classes que implementam as interfaces definidas, sem afetar o restante do sistema.

Linguagens de Reutilização

1. Programação orientada a aspecto utilizando AspectJ:
    - Adicionado o aspecto de visualização dos candidatos na classe "CandidateSelectionAspect" que realiza um ponto de corte na votação de presidentes e deputados federais para adicionar antes uma visualização dos candidatos.
2. Preprocessadores FeatureIDE e Antenna
    - Adicionado preprocessador para definir se haverá exibição ou não dos candidatos no momento da votação

Novas funcionalidades

- Votação para deputado estadual
- Votação para governador
- Visualização dos candidatos no momento da votação

Produtos da LPS

- Presidente/Deputado Federal com Exibição de Candidatos
- Presidente/Deputado Federal sem Exibição de Candidatos
- Governador/Deputador Estadual com Exibição de Candidatos
- Governador/Deputador Estadual sem Exibição de Candidatos

Demais modificações:

- Correção de indentações para 4 espaços.
- Adição de chaves "{}" em estruturas condicionais.
- Simplificação de verificações para redução de estruturas condicionais.
- Adicionado Singleton.
- Separação dos arquivos em pastas no padrão 3 camadas.

- Início do sistema pela classe Application, que possui apenas o método main e instancia as classes concretas e injeta as dependências.
- Corrigido bug onde votos brancos para Deputados Federais estavam contando como nulo.
- Todas pré-carregamentos (Presidentes, Deputados, Eleitores) agora são realizados nos seus respectivos repositórios.
- Removido padrão Builder pré-existente da classe ElectionController (não fazia muito sentido).
- Todos repositórios estão na classe ElectionController. Agora a Urna (que precisa ser o "frontend") apenas utiliza os serviços fornecidos por ElectionController e pelos modelos.
- Resultado da eleição agora é gerado por uma classe específica nomeada ElectionResult invocada por ElectionController.
- Removido da classe Voter a responsabilidade de votar chamando a própria eleição. Agora TODOS os votos são armazenados pela classe IVoteRepository e orquestrados por ElectionController.
- Removido da classe TSEEmployee a responsabilidade de adicionar/remover candidatos chamando a própria eleição. Agora a adição e remoção é controlada por ElectionController.
- Removido da classe CertifiedProfessional a responsabilidade de iniciar/finalizar uma eleição chamando a própria eleição. Agora o início e fim são controlados por ElectionController.
- Adicionado método login em TSEProfessional.
- ElectionController possui um método para cada tipo de voto, que são utilizados pela UrnaCli.
- Se para um cargo não houver votos válidos, nenhum candidato será eleito.