UnB Departamento de Ciência da Computação

Disciplina Técnicas de Programação 1

Implementação da Lógica do Software – Trabalho Prático

Grupo 7 – 2025-2

Davi Lopes Brito – 242023425 – 242023425@aluno.unb.br
Ricardo Rian da Silva Melo – 242003861 – rianricardo836@gmail.com
Samara da Conceição Gomes – 242014097 – samaracgomiis@gmail.com
Valquíria dos Santos Machado – 242003807 – valquiria.masan@gmail.com

Profa. Roberta Barbosa Oliveira

1 Introdução

O documento a seguir descreve a etapa de implementação da lógica do software do trabalho prático da disciplina **Técnicas de Programação 1**, no semestre 2/2025, pelo grupo 07. Para a etapa atual, o grupo teve de desenvolver os seguintes requisitos: **Implementação Técnica do Sistema**, **Transformação da Modelagem em Código**, **Implementação das Regras de Negócio**, **Mecanismos de Persistência** e **Tratamento de Exceções**. Além disso, o diagramaa de classe, estruturando seus principais elementos e relacionamentos.

DISCLAIMER: O grupo optou por já integrar as telas juntamente da lógica do software, a fim de manter a coerência com as telas já criadas, por esse motivo, ainda estamos na etapa final da implementação da lógica.

1.1 Implementação da Lógica do Software

1.1.1 Implementação Técnica do Sistema

Como definimos no início do projeto, trataremos nosso projeto com o modelo MVC (Model, View e Controller) estendida, que divide o projeto em alguns grandes grupos:

- **Model**: Onde as classes das entidades do sistema estão localizadas (Ex.: Usuario, Candidato);

- **View**: Telas FXML (para o caso do JavaFX), definem a estrutura visual;
- **Controller**: Classes que atuam como uma ponte entre o **Model** e **View**;
- **Service**: Contém lógicas e regras de negócios. Essa camada é essencial para a descrição de diversos serviços não definidos nas entidades;
- **Repository**: Camada de persistência de arquivos, que abstrai a entrada e saída de dados para os documentos CSV;
- **Util**: Classes adicionais que atuam como auxiliares.

1.1.2 Implementação das Regras de Negócio

Para o tratamento das regras de negócio, implementamos algumas Classes e Métodos a fim de garantir a integridade, segurança e cumprir com os requisitos do sistema, são eles:

- **Validação de Dados de Entrada**:
  - **CPF, E-mail e Senha**: Foram criadas classes especificamente para a validação de CPF (ValidaCpf), E-mail (ValidaEmail) e Senha (ValidaSenha), que contém a lógica de validação, como o cálculo dos dígitos do CPF, verificações de caracteres da senha e uso de Regex para e-mail;
  - **Preenchimento automático CEP**:
  - **Feedback ao Usuário**: Quando alguma das verificações falha, o sistema lança uma exceção (detalhada mais à frente), que é capturada no controller e gera um *PopUp*.
- **Controle de Acesso e Permissão**:
  - **LoginController e UserSession**: O LoginController autentica o usuário (utilizando, inclusive, criptografia, com a biblioteca *jBCrypt*) e o armazena em UserSession, uma classe Singleton;
  - **Seleção de Perfil**: As telas correspontdentes à cada nível de acesso (Administrador, Gestor, Recrutador e Funcionário) são selecionadas com base no perfil do usuário selecionado.
- **Lógica de Relatórios (Padrão Strategy)**:

  Para os relatórios, abordamos de uma maneira diferente, seguindo o requisito de flexibilidade, optamos por criar uma interface IGeradorRelatorio, que define qual tipo de relatório coletar e a interface IReportFormatter que define como formatar.

1.1.3 Mecanismos de Persistência

- **Formato**: Os dados são armazenados em formato **CSV** (Comma-Separated Values), utilizando ***;*** como delimitador;
- **Padrão Singleton**: As classes repositórios são implementadas como **Singletons**. Isso garante que exista apenas *uma* instância de cada repositório, mantendo uma única fonte de dados em memória;

- **Leitura**: Na inicialização do sistema, cada repositório usa um ***BufferedReader*** para ler os arquivos *.csv* linha por linha. O método de conversão é chamado e converte a String de texto para um objeto, que é adicionado à lista em memória;
- **Escrita**: Ao salvar ou atualizar o objeto, o método para persistir é chamado, esse método reescreve o CSV **INTEIRO**, usando um método para converter os objetos de volta em Strings e por fim gravando-os.

1.1.4 Tratamento de Exceções

Para garantir a robustez do sistema, utilizamos o tratamento de exceções, a fim de evitar erros.

- **Exceções Personalizadas**: Para a validação dos dados, necessitamos de algumas exceções personalizadas para serem disparadas, assim foram criadas tais exceções;
- **Exceções de I/O**: São exceções utilizadas para a entrada e saída de arquivos (Como fotos de perfil e upload de documentos);
- **Exceções padrão**: Por fim, utilizamos tratamentos padrão para entrada e saída de diversos tipos de dados, erros, formatações e etc.