PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CURSO DE ENGENHARIA DE SOFTWARE

DOCUMENTAÇÃO DO PROJETO JAVAFX CRUD

Equipe: FacilitaU(Grupo 6)

Integrantes:

- Éden Samuel
- Felipe Carneiro
- Fernando Lopes
- Henrique Ricardo
- Hugo Takeda

Curitiba

2025

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CURSO DE ENGENHARIA DE SOFTWARE

DOCUMENTAÇÃO DO PROJETO JAVAFX CRUD

Equipe: FacilitaU(Grupo 6)

Integrantes:

- Éden Samuel
- Felipe Carneiro
- Fernando Lopes
- Henrique Ricardo
- Hugo Takeda

Documento elaborado como parte das ativida- des acadêmicas da disciplina Programação Orientada a Objetos, sob orientação do professor Abimael Alves.

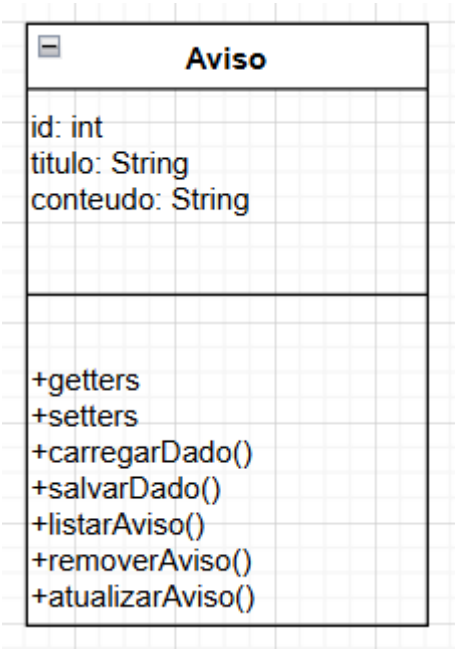
Curitiba

2025

SUMÁRIO

1 Descrição das Classes

1.1 Classe Aviso



Aluno responsável: Hugo Takeda

Descrição da classe Aviso:

A classe Aviso foi criada com o objetivo de armazenar e gerenciar os dados relacionados aos avisos do sistema. Essa classe está localizada dentro do pacote models do projeto JavaFX e implementa a interface Serializable, permitindo a persistência dos objetos em arquivos binários (.dat).

O propósito principal da classe é representar cada aviso que será mostrado ou manipulado pelo usuário, contendo um identificador único, um título e um conteúdo descritivo.

Ela é fundamental para o funcionamento do sistema de gerenciamento de avisos, sendo utilizada diretamente nas operações de CRUD (Create, Read, Update, Delete) implementadas na interface gráfica (AvisoView) e na camada de controle (AvisoController).

A persistência de dados é realizada por meio da serialização, o que garante que os avisos inseridos sejam salvos e possam ser recuperados posteriormente.

Atributos da Classe Aviso:

Modificador	Tipo	Nome	Descrição
private	int	id	Identificador único do aviso
private	String	titulo	Título do aviso
private	String	conteudo	Conteúdo ou descrição detalhada do aviso
private static final	long	serialVersionUID	Versão de serialização da classe

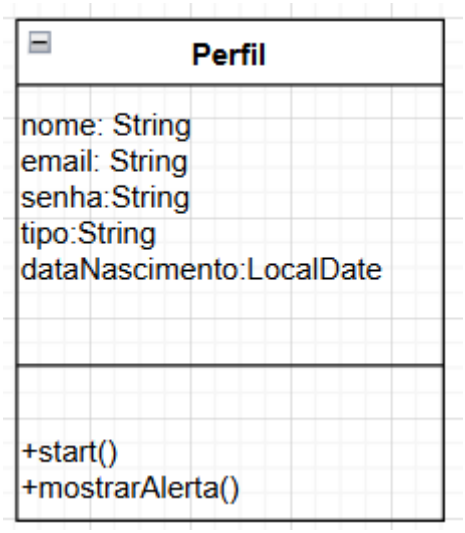
Métodos da Classe Aviso:

Modificador	Retorno	Nome do Método	Descrição
public	int	getId()	Retorna o ID do aviso
public	String	getTitulo()	Retorna o título do aviso
public	String	getConteudo()	Retorna o conteúdo do aviso
public	void	setId(int id)	Define um novo valor para o ID
public	void	setTitulo(String titulo)	Define um novo título para o aviso
public	void	setConteudo(String conteudo)	Define um novo conteúdo para o aviso
public	String	toString()	Retorna uma representação em String do objeto Aviso

Exemplo de Uso de IA (Inteligência Artificial) para Auxílio no Desenvolvimento:

Data/Hora	Prompt Utilizado	Ferramenta (IA)	Motivação
8/06/2025 20h16	"Como posso criar uma tabela em JavaFX para exibir uma lista de objetos da classe Aviso com colunas para ID, Título e Conteúdo?"	DeepSeek	Aprender a utilizar o **TableView** e **TableColumn** no JavaFX para exibir os dados da classe Aviso de forma dinâmica.

1.2Classe Perfil



Aluno responsável: Éden Samuel

Descrição da classe Perfil:

A classe **PerfilView** foi criada com o objetivo de permitir ao usuário visualizar e atualizar seus próprios dados de perfil dentro do sistema. Essa classe pertence ao pacote **views** do projeto JavaFX e é responsável por construir uma interface gráfica que exibe informações como **nome**, **data de nascimento**, **senha** (apenas leitura) e **tipo de usuário** (também somente leitura).
A principal funcionalidade da **PerfilView** é permitir a alteração controlada do nome e da data de

nascimento de um usuário, garantindo que informações sensíveis como senha e tipo de usuário permaneçam imutáveis por essa interface.
A interação com a lógica de negócios acontece por meio da classe **UsuarioController**, que contém os métodos para atualização e persistência dos dados.
Além disso, a interface foi construída utilizando elementos básicos do JavaFX, como **VBox**, **Label**, **TextField**, **DatePicker**, **PasswordField**, **Button** e **Alert**, garantindo uma experiência amigável ao usuário e um design funcional.

Atributos da Classe Perfil:

Tipo	Nome da Variável	Função
Label	nomeLabel	Exibe o rótulo "Nome:"
TextField	nomeField	Campo de texto para editar o nome do usuário
Label	emailLabel	Rótulo para campo email
TextField	emailField	Campo do email(apenas leitura)
Label	dataNascimentoLabel	Exibe o rótulo "Data de Nascimento:"
DatePicker	dataNascimentoField	Campo para selecionar a nova data de nascimento
Label	senhaLabel	Rótulo para campo de senha
PasswordField	senhaField	Campo de senha (apenas leitura)
Label	tipoLabel	Rótulo para tipo de usuário
TextField	tipoField	Campo que exibe o tipo de usuário (somente leitura)
Button	salvarBtn	Botão para salvar as alterações feitas no perfil

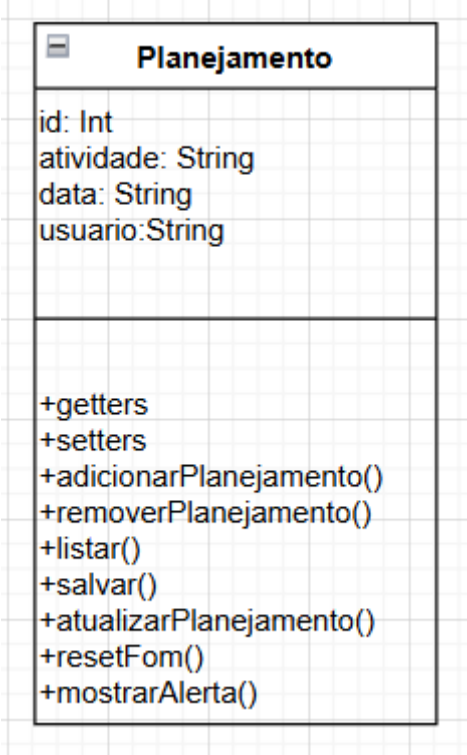
Métodos da Classe Perfil:

Modificador	Retorno	Nome do Método	Descrição
public	void	start(Stage stage, Usuario usuario)	Inicia a interface gráfica, exibindo os dados do perfil e permitindo edição do nome e data de nascimento
private	void	mostrarAlerta(String titulo, String mensagem)	Exibe mensagens de alerta para o usuário (informações, erros ou sucesso)

Exemplo de Uso de IA (Inteligência Artificial) para Auxílio no Desenvolvimento:

Data/Hora	Prompt Utilizado	Prompt Utilizado	Prompt Utilizado
9/06/2025 22h34	"Como criar uma tela de perfil de usuário com campos editáveis e outros apenas leitura usando JavaFX?"	ChatGPT (OpenAI)	Entender como implementar uma tela com campos de edição controlada, utilizando **TextField**, **DatePicker** e **PasswordField** no JavaFX.

1.3Classe Planejamento



Aluno responsável: Felipe Carneiro

Descrição da classe Planejamento:

A classe **Planejamento** foi criada com o objetivo de representar, armazenar e manipular os dados referentes às atividades planejadas pelos usuários dentro do sistema. Ela pertence ao pacote **models** e implementa a interface **Serializable**, o que permite que seus objetos sejam persistidos em arquivos binários (.dat) com segurança e compatibilidade de versão.
Cada instância da classe representa um registro de planejamento, contendo um identificador único, uma descrição da atividade, uma data planejada e o nome do usuário ao qual pertence.
Essa estrutura foi projetada para funcionar em conjunto com as classes **PlanejamentoController** e **PlanejamentoView**, viabilizando o ciclo completo de operações de **CRUD**.
A classe **PlanejamentoController** é responsável pela manipulação da lista de planejamentos em arquivos binários separados por usuário, enquanto a **PlanejamentoView** permite ao usuário visualizar, adicionar, editar e excluir planejamentos por meio de uma interface JavaFX com suporte à tabela dinâmica.

Atributos da Classe Planejamento:

Modificador	Tipo	Nome	Descrição
private	int	id	Identificador único do planejamento
private	String	atividade	Descrição textual da atividade planejada
private	String	data	Data associada à atividade, no formato texto (dd/mm/aaaa)
private	String	usuario	Nome do usuário responsável pelo planejamento
private static final	long	serialVersionUID	Versão da classe para controle de serialização

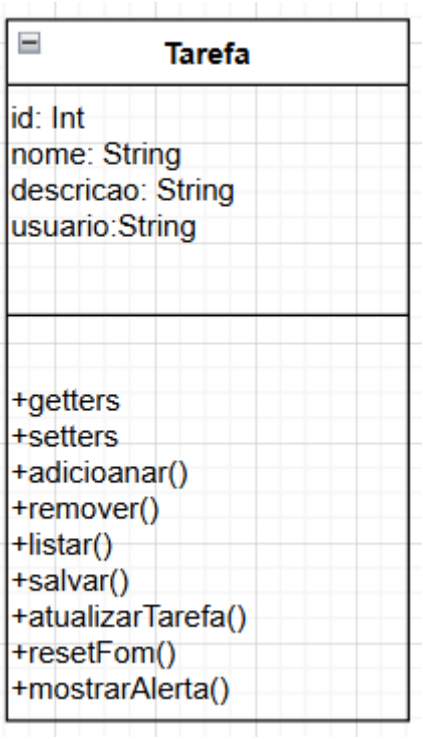
Métodos da Classe Planejamento:

Modificador	Retorno	Nome do Método	Descrição
public	int	getId()	Retorna o ID do planejamento
public	String	getAtividade()	Retorna a descrição da atividade planejada
public	String	getData()	Retorna a data do planejamento
public	String	getUsuario()	Retorna o nome do usuário associado
public	void	setId(int id)	Define um novo ID para o planejamento
public	void	setAtividade(String atividade)	Define uma nova descrição para a atividade
public	void	setData(String data)	Define uma nova data para o planejamento
public	void	setUsuario(String usuario)	Define o nome do usuário responsável pelo planejamento
public	String	toString()	Retorna uma representação em String do objeto Planejamento

Exemplo de Uso de IA (Inteligência Artificial) para Auxílio no Desenvolvimento:

Prompt Utilizado	Ferramenta (IA)	Motivação
"Como salvar listas de objetos diferentes para cada usuário em arquivos separados usando Java e serialização?"	DeepSeek	Aprender como criar arquivos `.dat` únicos por usuário e garantir a integridade dos dados ao persistir.

1.4Classe Tarefa



Aluno responsável: Fernando Lopes

Descrição da classe Tarefa:

A classe **Tarefa** foi desenvolvida com o objetivo de representar e armazenar informações relacionadas às tarefas dos usuários dentro do sistema. Localizada no pacote **models**, ela implementa a interface **Serializable**, o que permite salvar e carregar listas de tarefas de arquivos binários (.dat), garantindo a persistência de dados.
Cada objeto da classe **Tarefa** representa uma atividade que um usuário precisa realizar, contendo um **ID único**, **nome da tarefa**, **descrição detalhada** e o **nome do usuário responsável**.
A classe **TarefaController** é responsável por realizar as operações de **CRUD (Create, Read, Update, Delete)** sobre as tarefas, enquanto a interface **TarefaView**, construída com JavaFX, permite ao usuário adicionar, listar, editar e excluir tarefas de forma visual e interativa.

Atributos da Classe Tarefa:

Modificador	Tipo	Nome	Descrição
private	int	id	Identificador único da tarefa
private	String	nome	Nome curto da tarefa
private	String	descricao	Descrição detalhada da tarefa
private	String	usuario	Nome do usuário responsável pela tarefa
private static final	long	serialVersionUID	Versão da classe para controle de serialização

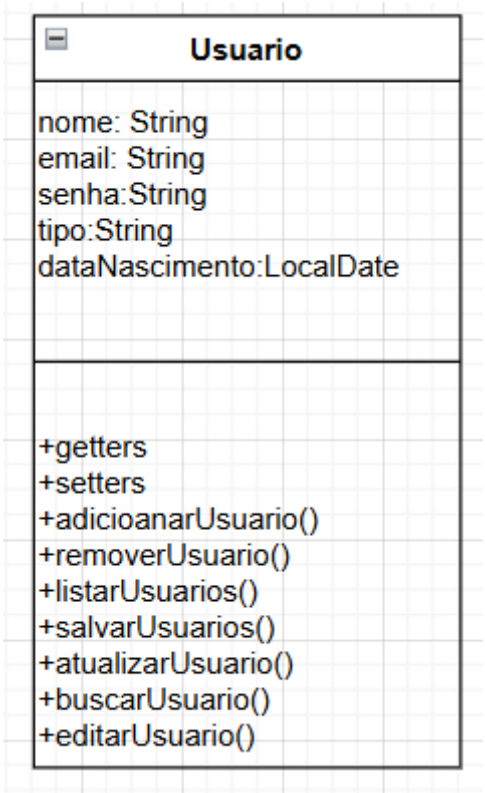
Métodos da Classe Tarefa:

Modificador	Retorno	Nome do Método	Descrição
public	int	getId()	Retorna o ID da tarefa
public	String	getNome()	Retorna o nome da tarefa
public	String	getDescricao()	Retorna a descrição detalhada da tarefa
public	String	getUsuario()	Retorna o usuário associado à tarefa
public	void	setId(int id)	Define um novo ID para a tarefa
public	void	setNome(String nome)	Define um novo nome para a tarefa
public	void	setDescricao(String descricao)	Define uma nova descrição para a tarefa
public	void	setUsuario(String usuario)	Define o usuário associado à tarefa

Exemplo de Uso de IA (Inteligência Artificial) para Auxílio no Desenvolvimento:

Data/Hora	Prompt Utilizado	Ferramenta (IA)	Motivação
08/06/2025 22h58	"Como implementar um TableView JavaFX que permita adicionar, editar e remover registros de planejamento com campos dinâmicos?"	GROK3	Saber como estruturar uma interface JavaFX responsiva e funcional para CRUD de planejamentos.

1.5Classe Usuário



Aluno responsável: Henrique Costa

Descrição da classe Usuario:

A classe **Usuario** foi criada com o objetivo de armazenar e gerenciar as informações dos usuários do sistema. Ela está localizada no pacote **models** e implementa a interface **Serializable**, permitindo a persistência dos dados em arquivos binários (**usuarios.dat**).
Cada instância da classe representa um usuário cadastrado no sistema, contendo informações essenciais como **nome**, **email**, **senha**, **tipo de usuário** e **data de nascimento**.
As operações de **CRUD (Create, Read, Update, Delete)** para a classe Usuario são controladas pela classe **UsuarioController**, responsável por ler e salvar os usuários no arquivo. A interação gráfica com o usuário é feita através da interface **UsuarioView**, construída com JavaFX.
Além das operações básicas de cadastro, listagem e remoção, o sistema também permite a edição de dados dos usuários.

Atributos da Classe Usuário:

Modificador	Tipo	Nome	Descrição
private	String	nome	Nome completo do usuário
private	String	email	Email do usuário (utilizado como identificador único)
private	String	senha	Senha de acesso do usuário
private	String	tipo	Tipo do usuário (Ex: Estudante, Professor, Coordenador)
private	LocalDate	dataNascimento	Data de nascimento do usuário

Métodos da Classe Usuário:

Modificador	Retorno	Nome do Método	Descrição
public	String	getNome()	Retorna o nome do usuário
public	void	setNome(String nome)	Define o nome do usuário
public	String	getEmail()	Retorna o email do usuário
public	void	setEmail(String email)	Define o email do usuário
public	String	getSenha()	Retorna a senha do usuário
public	void	setSenha(String senha)	Define a senha do usuário
public	String	getTipo()	Retorna o tipo do usuário
public	void	setTipo(String tipo)	Define o tipo do usuário
public	LocalDate	getDataNascimento()	Retorna a data de nascimento do usuário
public	void	setDataNascimento(LocalDate	Define a data de

		dataNascimento)	nascimento do usuário
public	String	toString()	Retorna uma representação textual resumida do usuário

Exemplo de Uso de IA (Inteligência Artificial) para Auxílio no Desenvolvimento:

Data/Hora	Prompt Utilizado	Ferramenta (IA)	Motivação
8/06/2025 17h14	"No meu CRUD de usuários em JavaFX, como posso fazer para preencher automaticamente os campos de um formulário ao selecionar um usuário na tabela, permitindo a edição dos dados e depois salvar as alterações?"	ChatGPT (OpenAI)	Entender como implementar a funcionalidade de **edição de dados**, preenchendo os campos com os dados do usuário selecionado e salvando as alterações corretamente.

2 Relatos Individuais

2.1 Relato Individual – Hugo Takeda

Análise Pessoal:

Durante o desenvolvimento da classe **Aviso**, pude consolidar meus conhecimentos em orientação a objetos **e** persistência de dados com Java. Trabalhar com a criação do CRUD em JavaFX me ajudou a melhorar minhas habilidades com construção de interfaces gráficas sem o uso do SceneBuilder, o que exigiu uma codificação mais manual e detalhada. Foi gratificante ver a aplicação funcionando com a listagem dinâmica dos avisos e com todas as operações de inclusão, edição e exclusão de registros.

Dificuldades Encontradas:

Minha maior dificuldade foi estruturar corretamente o uso da **TableView** para listar os avisos de forma dinâmica. Também tive alguns desafios no início com a serialização dos objetos para gravar os dados em arquivos binários, principalmente para evitar erros de leitura ou gravação de objetos não serializáveis.

2.2 Relato Individual – Éden Samuel

Análise Pessoal:

Ficar responsável pela classe **PerfilView** foi um processo muito enriquecedor. Trabalhei diretamente com a criação de formulários JavaFX para exibir e permitir a edição de dados do usuário, o que me deu uma visão mais clara sobre a manipulação de componentes como **TextField, DatePicker e PasswordField**. O fato de ter que manipular apenas campos específicos, enquanto outros permaneciam apenas para leitura, me ensinou muito sobre controle de acesso aos dados da interface**.**

Dificuldades Encontradas:

Tive dificuldades em manter o estado correto dos campos entre a seleção de usuários diferentes, principalmente ao lidar com eventos de botões e validação de formulários. Também levei um tempo para entender como atualizar corretamente o objeto Usuario sem afetar os dados imutáveis como email e tipo**.**

2.3 Relato Individual – Felipe Carneiro

Análise Pessoal:

Desenvolver a classe **Planejamento** me proporcionou um aprendizado profundo sobre persistência por usuário, já que precisei salvar planejamentos em arquivos separados para

cada usuário. Também aprendi muito sobre como manipular **TableViews** para listar e atualizar os registros em tempo real na interface gráfica. Foi uma boa oportunidade para melhorar meu entendimento sobre **Streams**, **Collections** e **file I/O em Java**.

Dificuldades Encontradas:

Minha maior dificuldade foi garantir que os planejamentos de cada usuário fossem salvos de forma isolada, sem sobrescrever os arquivos de outros usuários. Também tive problemas iniciais com a atualização de registros dentro da lista e com o **refresh da TableView** após a edição dos dados.

2.4 Relato Individual – Fernando Lopes

Análise Pessoal:

Ao trabalhar na classe **Tarefa**, aprendi a implementar operações de **CRUD** completo com **JavaFX**, incluindo o gerenciamento de arquivos por usuário. A implementação da interface gráfica me ajudou a melhorar minha prática com componentes como **TextField**, **TableView** e **event handling** no JavaFX. Também aprendi sobre boas práticas de serialização de listas de objetos.

Dificuldades Encontradas:

As principais dificuldades foram na edição de registros, principalmente ao atualizar corretamente o objeto selecionado na lista e sincronizar os dados na **TableView** após as alterações. Também precisei fazer alguns ajustes na validação de campos para evitar cadastros incompletos.

2.5 Relato Individual – Henrique Ricardo

Análise Pessoal:

Desenvolver a classe **Usuario** foi um desafio e uma experiência muito importante. Trabalhei diretamente com o controle de usuários, criando o CRUD completo e também a lógica de login baseada em email e senha. Pude aprofundar meus conhecimentos em **JavaFX**, principalmente no uso de **TableView** com múltiplas colunas, além de lidar com **ComboBox** e **DatePicker** para entrada de dados mais sofisticada.

Dificuldades Encontradas:

Minhas maiores dificuldades foram na parte de validação de campos obrigatórios, especialmente para evitar cadastros de usuários com dados incompletos ou inválidos. Além disso, ajustar a lógica para permitir a edição de dados sem alterar o email (chave primária) exigiu bastante atenção.

Referências Bibliográficas e Online:

- Oracle. **Java Platform, Standard Edition 8 API Specification**. Disponível em: https://docs.oracle.com/javase/8/docs/api/.
- Sierra, Kathy; Bates, Bert. **Use a Cabeça! Java: Aprenda de forma divertida**.
- GeeksforGeeks. **Serialization in Java**. Disponível em: https://www.geeksforgeeks.org/serialization-in-java/.
- Documentações do Canvas POO
- ChatGPT (OPEN IA).
- DeepSeek
- Grok3