Universidade Tiradentes
CIÊNCIA DA COMPUATÇÃO

ANA CAROLINA ANDRADE PASSOS
ERIC BOMFIM MARIANO
GABRIEL SOUZA SANTOS
JULIANA SAMPAIO SILVA IVO
PEDRO HENRIQUE CARVALHO NOVAES
ROQUE TAVARES DE JESUS NETO

PROJETO DE PROGRAMAÇÃO - MÓDULO CADASTRO

DETALHAMENTO DA ATIVIDADE

Aracaju - SE
2025

ANA CAROLINA ANDRADE PASSOS
ERIC BOMFIM MARIANO
GABRIEL SOUZA SANTOS
JULIANA SAMPAIO SILVA IVO
PEDRO HENRIQUE CARVALHO NOVAES
ROQUE TAVARES DE JESUS NETO

ATIVIDADE

DETALHAMENTO DA ATIVIDADE

ATIVIDADE sobre Módulo de Cadastro apresentado como requisito parcial da avaliação da disciplina Projeto de Programação, ministrada pela Profª. Layse Souza, no 2º semestre de 2025.

Aracaju - SE
2025

Sumário

1 INTRODUÇÃO

A cidade de Véridia deseja digitalizar e organizar a oferta de cursos e treinamentos oferecidos por instituições públicas e privadas. A prefeitura contratou a equipe para desenvolver uma Plataforma de Gerenciamento de Cursos e Treinamentos, que permita aos alunos se inscreverem em cursos, aos instrutores gerenciarem aulas e aos administradores controlarem toda a operação.

Nossa equipe ficou responsável por criar e implementar o Cadastro de Usuários no sistema de ponta a ponta, *backend* e *frontend*, conforme os requisitos definidos na primeira etapa do projeto. As principais atividades foram a criação de classes, cadastro, edição e autenticação de dados, validações de informação, e o desenvolvimento das telas de login e cadastro.

2 JUSTIFICATIVA

O módulo de cadastro constitui-se como o alicerce fundamental da Plataforma de Gerenciamento de Cursos e Treinamentos de Véridia, sendo responsável por estabelecer a base de dados essencial para o funcionamento de todos os demais componentes do sistema. Sua implementação adequada é crucial para garantir a integridade, segurança e eficiência operacional da plataforma como um todo.

2.1 Importância Estratégica

A digitalização dos serviços públicos representa uma demanda crescente na administração municipal contemporânea. No contexto educacional, a centralização das informações sobre cursos e treinamentos em uma plataforma única elimina a fragmentação de dados que tradicionalmente ocorre quando diferentes instituições mantêm sistemas isolados. O módulo de cadastro viabiliza essa unificação ao estabelecer um repositório centralizado e padronizado de informações.

Atualmente, a cidade de Véridia enfrenta desafios significativos na organização da oferta educacional, incluindo duplicidade de registros, dificuldade no acompanhamento de vagas disponíveis e ausência de um histórico consolidado dos participantes. A implementação de um módulo de cadastro robusto soluciona essas questões ao criar um sistema único de registro que serve como fonte de verdade para toda a operação da plataforma.

2.2 Fundamentos Técnicos

O módulo de cadastro opera como a camada de entrada de dados do sistema, sendo responsável por coletar, validar e armazenar informações críticas sobre três entidades principais: usuários, cursos e aulas. Esta arquitetura segue o princípio de separação de responsabilidades, onde o cadastro se concentra exclusivamente na gestão de dados mestres, permitindo que outros módulos consumam essas informações de forma consistente.

A escolha pela implementação utilizando o padrão Repository em Java com Spring Boot proporciona uma separação clara entre a lógica de negócio e a camada de persistência. Esta abordagem facilita a manutenção do código, permite a realização de testes unitários de forma isolada e possibilita futuras migrações de tecnologia de banco de dados sem impactar significativamente a lógica de aplicação.

2.3 Gestão de Usuários

O cadastro de usuários representa a porta de entrada para qualquer interação com o sistema. A necessidade de diferenciar três perfis distintos (alunos, instrutores e administradores) exige uma modelagem que contemple tanto informações comuns quanto atributos específicos de cada tipo de usuário. Alunos necessitam de campos relacionados ao histórico educacional e preferências de curso, instrutores requerem informações sobre qualificações e disponibilidade, enquanto administradores demandam permissões e níveis de acesso diferenciados.

A implementação de validações rigorosas no momento do cadastro previne inconsistências futuras. Verificações de unicidade de email e CPF, validação de formatos de dados e aplicação de regras de negócio específicas garantem que apenas informações íntegras sejam persistidas no banco de dados. Adicionalmente, a incorporação de mecanismos de autenticação segura desde o cadastro inicial estabelece as bases para um sistema protegido contra acessos não autorizados.

2.4 Integridade e Relacionamentos

A definição clara dos relacionamentos entre as entidades cadastradas constitui um aspecto crítico do módulo. Um curso pode ter múltiplas aulas, um instrutor pode ministrar diversos cursos, e um aluno pode estar inscrito em vários programas simultaneamente. O estabelecimento correto dessas relações no banco de dados, através de chaves estrangeiras e constraints apropriadas, garante a consistência referencial dos dados.

A implementação de regras de cascata para operações de atualização e exclusão previne situações de inconsistência. Por exemplo, ao desativar um curso, o sistema deve automaticamente cancelar as inscrições associadas e notificar os alunos afetados. Essas regras de integridade, definidas tanto no nível de banco de dados quanto na camada de aplicação, asseguram que o estado do sistema permaneça coerente mesmo diante de operações complexas.

2.5 Impacto nos Demais Módulos

O módulo de cadastro serve como alicerce para todas as demais funcionalidades do sistema. O módulo de inscrições depende dos dados de cursos e usuários para processar matrículas. O sistema de pagamentos necessita das informações cadastrais para gerar cobranças. O módulo de notificações utiliza os contatos registrados para enviar comunicações. Os relatórios gerenciais consolidam dados originados nos cadastros. Esta interdependência evidencia que qualquer falha ou inconsistência no módulo de cadastro se propagaria por todo o sistema, comprometendo sua operação.

3 OBJETIVOS

3.1 Objetivo Geral

Implementar o módulo de Gerenciamento de Identidade e Acesso (Grupo 1), garantindo que o sistema possa cadastrar, autenticar e diferenciar permissões de usuários (Aluno, Instrutor, Administrador) de forma segura, validada e persistente.

3.2 Objetivos Específicos

- Criar a entidade `Usuario` e implementar o `UsuarioRepository` com todas as operações CRUD (*create*, *read*, *update*, *delete*) e os métodos customizados (autenticação, listar por tipo);
- Desenvolver a lógica de autenticação (login por e-mail e senha) e o controle de acesso baseado nos papéis (Tipos de Usuário);
- Implementar as rotinas de validação de formato (e-mail, CPF, telefone) e as regras de negócio (verificação de duplicidade de CPF/e-mail);
- Implementar o fluxo de redefinição de senha via e-mail, utilizando um token de validação temporário;
- Desenvolver as telas (GUI) necessárias para as operações de Login e Cadastro de novos usuários;
- Validar todas as funcionalidades do módulo (cadastro, login, validações, redefinição de senha) para garantir a integridade e o correto funcionamento.

4 METODOLOGIA

Para a execução deste projeto, a equipe optou por uma abordagem de desenvolvimento iterativo, organizada em ciclos semanais fixos e gerenciada através de um fluxo de trabalho estruturado no Git e GitHub. O objetivo principal foi garantir um fluxo de trabalho contínuo, com entregas incrementais e uma divisão clara de responsabilidades.

4.1 Estrutura dos Ciclos e Versionamento

O processo de trabalho foi estruturado em *Sprints* semanais, onde cada ciclo de sete dias representava uma iteração completa de planejamento, execução e integração de código.

O funcionamento ocorria da seguinte forma:

1. **Planejamento Semanal:** No início de cada semana, a equipe definia as metas e as tarefas (*tasks*) prioritárias para aquela iteração, focando nos requisitos do sistema de login e cadastro.

2. **Distribuição de Tarefas:** As tarefas eram distribuídas de forma equitativa entre todos os membros da equipe.

3. **Desenvolvimento com Git (Branching):** Cada membro criava uma *branch* específica no Git para desenvolver a sua tarefa atribuída. Isso garantia que o código principal (*branch* `main` ou `develop`) permanecesse estável.

4. **Entrega e Revisão (Pull Requests):** Ao concluir a tarefa semanal, cada membro abria um *Pull Request* (PR) no GitHub. Isso permitia que os outros membros da equipe revisassem o código (*Code Review*) antes que ele fosse mesclado à *branch* principal.

5. **Integração Contínua:** Após a aprovação e a mesclagem (*merge*) do PR, o novo código era integrado ao projeto principal, marcando a conclusão daquela tarefa.

4.2 Ferramentas de Gestão e Colaboração

O sucesso desta abordagem foi suportado por duas ferramentas principais:

- **Git:** Utilizado para o controle de versão local, permitindo que cada desenvolvedor trabalhasse em suas funcionalidades de forma isolada e segura.

- **GitHub:** Serviu como o repositório centralizado (remoto) e a plataforma de colaboração. O GitHub foi essencial para o gerenciamento das *branches*, a realização de *Pull Requests* e a revisão de código.

5 RESULTADOS E DISCUSSÕES

5.1 Implementação do Módulo de Gestão de Usuários

O módulo de gestão de usuários foi desenvolvido conforme os objetivos definidos, implementando a entidade `Usuario` e o `UsuarioRepository` com todas as operações CRUD e métodos personalizados de autenticação e listagem por tipo. A aplicação realiza o login por e-mail e senha, identificando automaticamente o papel do usuário e controlando o acesso conforme as permissões definidas, o que garante segurança e organização nas operações.

5.1.1 Validações e Integridade de Dados

As rotinas de validação foram aplicadas para e-mail, CPF e telefone, assegurando o formato correto e bloqueando cadastros duplicados. Também foram implementadas regras de negócio que impedem inconsistências e preservam a integridade dos dados. O fluxo de redefinição de senha foi desenvolvido com envio de token temporário por e-mail, reforçando a confiabilidade e a proteção das credenciais.

5.1.2 Interface Gráfica e Usabilidade

As interfaces gráficas de login e cadastro foram criadas com layout responsivo e validações automáticas, proporcionando uma experiência simples e intuitiva ao usuário. Após testes de todas as funcionalidades — incluindo cadastro, autenticação, validações e redefinição de senha —, verificou-se o funcionamento correto e estável do módulo, confirmando o atendimento aos requisitos de segurança, usabilidade e integridade do sistema.

5.1.3 Análise dos Resultados

Em síntese, o módulo cumpre integralmente os objetivos propostos e garante um fluxo de autenticação seguro e eficiente. A implementação demonstrou-se robusta durante os testes realizados, validando a arquitetura escolhida e as decisões técnicas adotadas no desenvolvimento.

6 CONSIDERAÇÕES FINAIS

Este projeto alcançou seu objetivo principal: a implementação bem-sucedida do Módulo de Gerenciamento de Identidade e Acesso para a plataforma educacional da cidade. Mais do que apenas um requisito técnico, este módulo representa o alicerce fundamental sobre o qual todo o ecossistema de aprendizado será construído. Ao centralizar e validar o cadastro de alunos, instrutores e administradores, demos o primeiro passo concreto para resolver a fragmentação e a duplicidade de dados que historicamente dificultavam o acesso à educação na cidade.

A adoção de uma metodologia iterativa, com ciclos semanais e revisões de código via *Pull Requests*, foi crucial para garantir a robustez e a segurança do sistema. O resultado é um fluxo de autenticação, validação e cadastro que se provou funcional e seguro, oferecendo uma interface de entrada simples e confiável para os futuros usuários.

7 REFERÊNCIAS

PIVOTAL SOFTWARE. *Spring Boot Reference Documentation*. Version 3.2. Disponível em: `https://docs.spring.io/spring-boot/docs/current/reference/html/`. Acesso em: 10 nov. 2025.

PIVOTAL SOFTWARE. *Spring Data JPA - Reference Documentation*. Disponível em: `https://docs.spring.io/spring-data/jpa/docs/current/reference/html/`. Acesso em: 10 nov. 2025.

ORACLE. *Java Platform, Standard Edition Documentation*. Version 17. Disponível em: `https://docs.oracle.com/en/java/javase/17/`. Acesso em: 10 nov. 2025.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL Documentation*. Version 16. Disponível em: `https://www.postgresql.org/docs/16/`. Acesso em: 10 nov. 2025.

OBJECT MANAGEMENT GROUP. *Unified Modeling Language (UML) Specification*. Version 2.5.1. Disponível em: `https://www.omg.org/spec/UML/`. Acesso em: 10 nov. 2025.

FOWLER, Martin. *UML Essencial: Um Breve Guia para Linguagem Padrão de Modelagem de Objetos*. 3. ed. Porto Alegre: Bookman, 2005.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de Software: Uma Abordagem Profissional*. 8. ed. Porto Alegre: AMGH, 2016.

GIT. *Git Documentation*. Disponível em: `https://git-scm.com/doc`. Acesso em: 10 nov. 2025.

GITHUB. *GitHub Guides*. Disponível em: `https://guides.github.com/`. Acesso em: 10 nov. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. *The Scrum Guide: The Definitive Guide to Scrum*. 2020. Disponível em: `https://scrumguides.org/`. Acesso em: 10 nov. 2025.

GAMMA, Erich et al. *Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos*. Porto Alegre: Bookman, 2000.

MARTIN, Robert C. *Código Limpo: Habilidades Práticas do Agile Software*. Rio de Janeiro: Alta Books, 2009.

BAUER, Christian; KING, Gavin; GREGORY, Gary. *Java Persistence with Hibernate*. 2. ed. Greenwich: Manning Publications, 2015.

REENSKAUG, Trygve. *The Model-View-Controller (MVC): Its Past and Present*. University of Oslo, 2003. Disponível em: `https://folk.universitetetioslo.no/trygver/`. Acesso em: 10 nov. 2025.

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de Banco de Dados*. 7. ed. São Paulo: Pearson, 2019.

STALLINGS, William. *Criptografia e Segurança de Redes: Princípios e Práticas*. 6. ed. São Paulo: Pearson, 2015.

8 ANEXOS

8.1 Diagramas UML

8.1.1 Diagrama de Classes

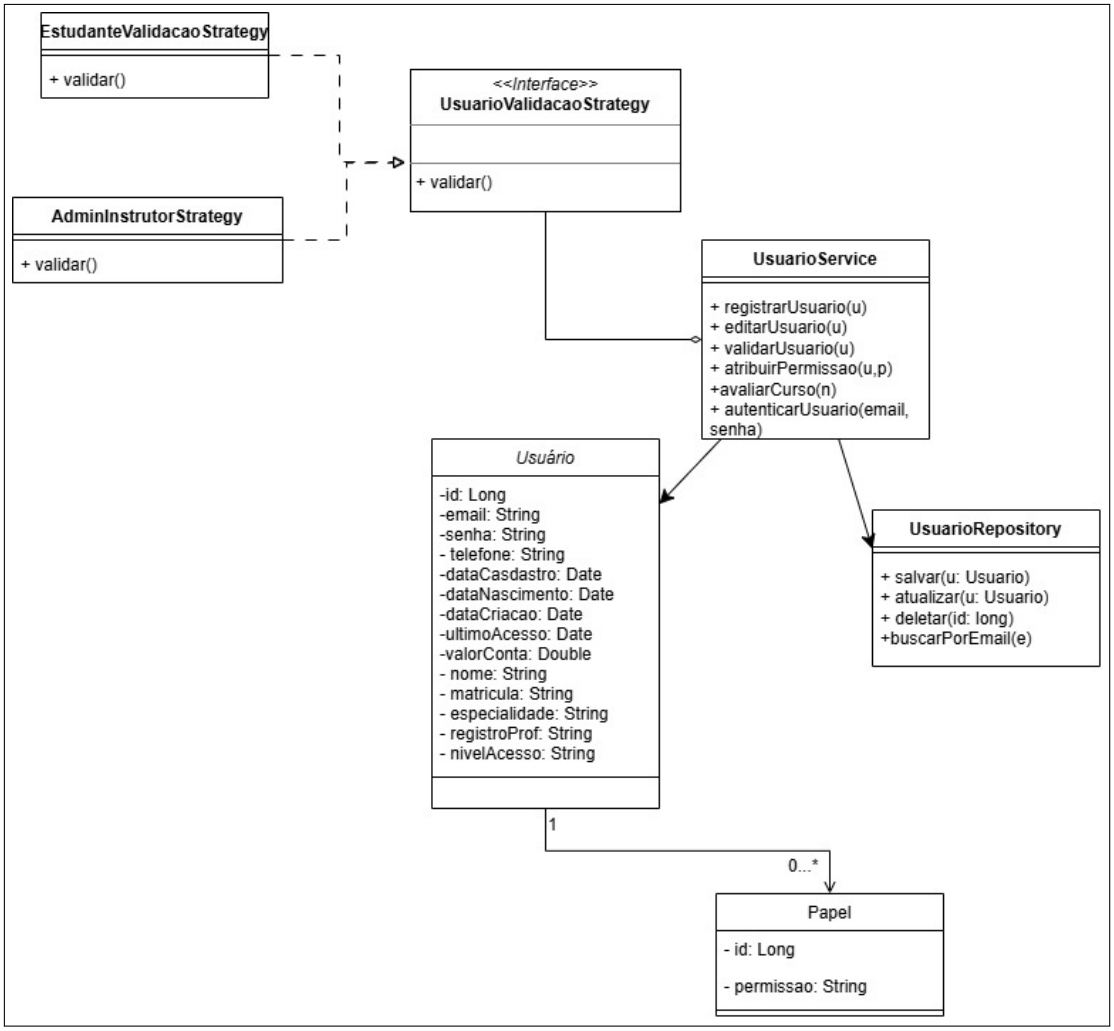

Figura 1 – Diagrama de Classes do Sistema

8.1.2 Diagrama de Casos de Uso

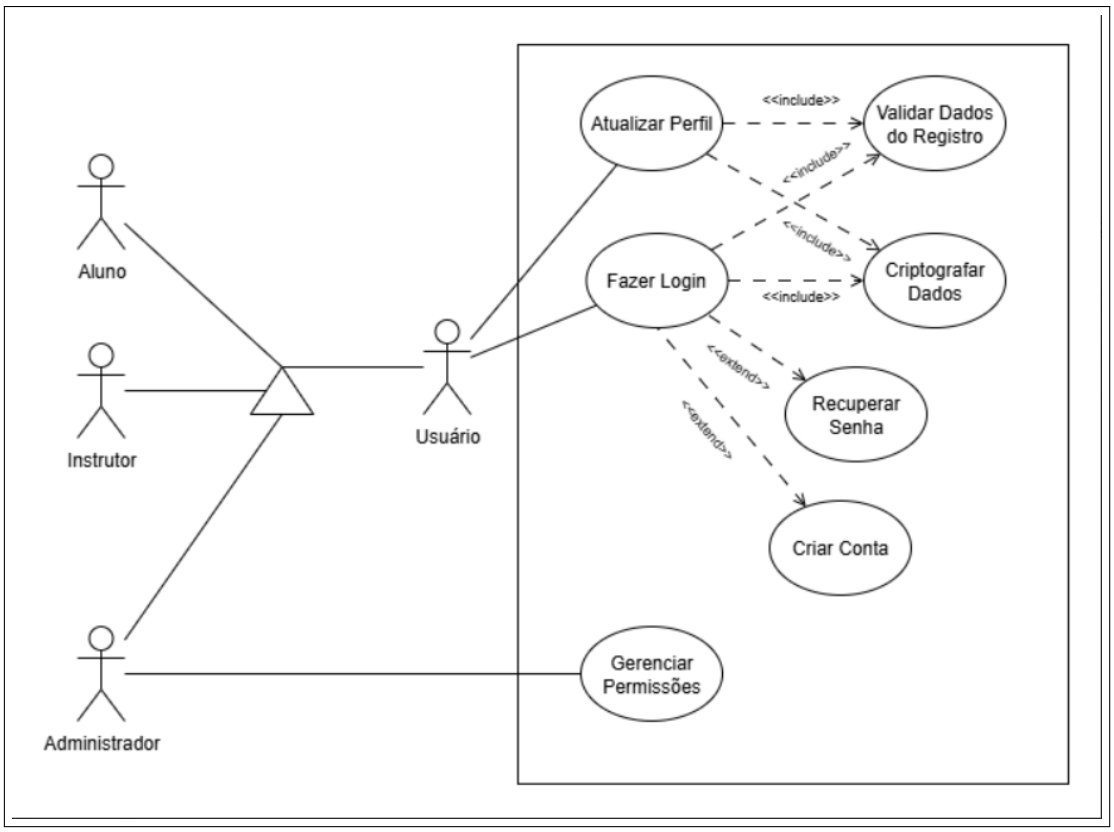

Figura 2 – Diagrama de Casos de Uso do Sistema