**Documentação Completa do Aplicativo AlunoSaqua**

**1. Introdução**

O AlunoSaqua é um sistema web desenvolvido para aprimorar a comunicação e o acompanhamento acadêmico e psicossocial de alunos no ambiente escolar. Originado de um pré-projeto de Práticas Extensionistas Integradas III da Universidade de Vassouras, o aplicativo visa combater o bullying, monitorar o desempenho escolar (faltas e notas) e facilitar a interação entre alunos, pais, professores e psicopedagogos. Esta documentação detalha a versão do aplicativo implementada em Java com Spring Boot e Thymeleaf.

**2. Visão Geral do Projeto**

**Objetivos:**

* Fornecer um ambiente seguro para denúncias anônimas de bullying e violência escolar.
* Permitir que alunos e pais acompanhem o desempenho acadêmico (notas e frequência).
* Capacitar professores a gerenciar e atualizar notas e faltas de suas disciplinas.
* Oferecer a psicopedagogos uma ferramenta para visualizar e gerenciar denúncias por nível de urgência, com acesso a informações relevantes do aluno.
* Promover o bem-estar emocional e a saúde mental dos estudantes.

**Público-alvo:**

* **Alunos:** Para denúncias e acompanhamento próprio.
* **Pais/Responsáveis:** Para monitorar o desempenho dos filhos.
* **Professores:** Para gestão de dados acadêmicos.
* **Psicopedagogos:** Para gerenciamento de casos e denúncias.

**Tecnologias Utilizadas:**

* **Backend:** Java 17+, Spring Boot 3.2.7, Maven, Lombok.
* **Frontend:** HTML5, CSS3 (customizado com base em design fornecido), JavaScript, Thymeleaf (motor de templates), Chart.js (gráficos).
* **Armazenamento de Dados:** Estruturas de dados em memória (Maps estáticos) para fins de MVP.

**3. Requisitos Funcionais (RF)**

Os requisitos funcionais descrevem as funcionalidades que o sistema deve executar.

* **RF001: Autenticação de Usuários**
  + **Descrição:** O sistema deve permitir que usuários (alunos, pais, professores, psicopedagogos) realizem login informando seu tipo de usuário, nome de usuário e senha.
  + **Entradas:** Tipo de Usuário, Nome de Usuário, Senha.
  + **Saídas:** Redirecionamento para o dashboard correspondente ou mensagem de erro.
  + **Regras:** Credenciais devem ser válidas para o tipo de usuário selecionado.
* **RF002: Cadastro de Alunos**
  + **Descrição:** O sistema deve permitir que novos alunos se cadastrem fornecendo matrícula, nome completo e senha.
  + **Entradas:** Matrícula, Nome Completo, Senha.
  + **Saídas:** Mensagem de sucesso e redirecionamento para a página de login, ou mensagem de erro se a matrícula já existir.
  + **Regras:** Matrícula deve ser única. Novos alunos são inicializados com matérias padrão e zero faltas/notas vazias.
* **RF003: Dashboard do Aluno**
  + **Descrição:** O aluno deve ter acesso a um dashboard pessoal que exibe seu nome, total de faltas, porcentagem de faltas, status de aprovação por faltas, notas por matéria (Nota 1, Nota 2, Soma e Status), status geral de notas, lista de matérias reprovadas e status final de aprovação.
  + **Entradas:** Login de Aluno.
  + **Saídas:** Página de dashboard com informações acadêmicas e gráficos.
  + **Regras:** Apenas o aluno logado pode ver suas próprias informações.
* **RF004: Denúncia Anônima (Aluno)**
  + **Descrição:** O aluno deve poder registrar denúncias anônimas de bullying ou violência, informando uma descrição e o grau de urgência.
  + **Entradas:** Descrição da denúncia, Grau de Urgência (baixa, média, alta).
  + **Saídas:** Mensagem de sucesso e redirecionamento para o dashboard do aluno.
  + **Regras:** A denúncia é associada à matrícula do aluno, mas seu nome não é diretamente visível para o psicopedagogo na listagem inicial.
* **RF005: Dashboard dos Pais**
  + **Descrição:** Os pais devem ter acesso a um dashboard que exibe as informações acadêmicas de seu(sua) filho(a), incluindo total de faltas, porcentagem de faltas, status de aprovação por faltas, notas por matéria (Nota 1, Nota 2, Soma e Status), status geral de notas, lista de matérias reprovadas e status final de aprovação.
  + **Entradas:** Login de Pais.
  + **Saídas:** Página de dashboard com informações acadêmicas do filho(a) e gráficos.
  + **Regras:** A conta de pais deve estar associada à matrícula de um aluno existente.
* **RF006: Dashboard do Professor**
  + **Descrição:** O professor deve ter acesso a um dashboard que lista todos os alunos. Para cada aluno, deve exibir matrícula, nome, turma, total de faltas, porcentagem de faltas, status por faltas, e, se o professor leciona uma disciplina específica, a soma das notas e o status de aprovação para **apenas essa disciplina**. O dashboard também mostra o status final de aprovação do aluno.
  + **Entradas:** Login de Professor.
  + **Saídas:** Página de dashboard com lista de alunos e suas informações relevantes.
  + **Regras:** Professores com disciplina específica veem apenas o desempenho nessa disciplina na tabela. Professores "gerais" veem o status geral de notas. A lista é ordenada por turma e nome.
* **RF007: Atualização de Dados do Aluno (Professor)**
  + **Descrição:** O professor deve poder atualizar o número de faltas de um aluno. Se o professor leciona uma disciplina específica, ele deve poder atualizar apenas as duas notas dessa disciplina para o aluno. Professores "gerais" podem atualizar as notas de qualquer disciplina. O professor também pode adicionar uma nova disciplina e suas notas para um aluno.
  + **Entradas:** Matrícula do aluno, Novo número de faltas, Notas (Nota 1, Nota 2) para a disciplina relevante, Nome e Notas (Nota 1, Nota 2) para nova disciplina (opcional).
  + **Saídas:** Mensagem de sucesso ou erro, e redirecionamento para o dashboard do professor.
  + **Regras:** Notas devem ser numéricas (0-10). Apenas o professor pode acessar esta funcionalidade.
* **RF008: Dashboard do Psicopedagogo**
  + **Descrição:** O psicopedagogo deve ter acesso a um dashboard que lista todas as denúncias abertas, categorizadas por grau de urgência (alta, média, baixa). Cada item da lista deve exibir o nome do aluno (não anônimo aqui), a matrícula, uma breve descrição da denúncia e um resumo das notas do aluno (disciplina, soma e status).
  + **Entradas:** Login de Psicopedagogo.
  + **Saídas:** Página de dashboard com denúncias organizadas por urgência.
* **RF009: Detalhes da Denúncia (Psicopedagogo)**
  + **Descrição:** Ao clicar em uma denúncia no dashboard do psicopedagogo, o sistema deve exibir uma página com todos os detalhes da denúncia (descrição completa, urgência, status), o nome completo do aluno, matrícula e uma tabela detalhada com todas as suas notas por matéria.
  + **Entradas:** ID da denúncia.
  + **Saídas:** Página de detalhes da denúncia com informações do aluno.
* **RF010: Gerenciamento de Denúncias (Psicopedagogo)**
  + **Descrição:** O psicopedagogo deve poder marcar uma denúncia como "fechada" a partir da página de detalhes da denúncia.
  + **Entradas:** ID da denúncia, Ação de "Fechar Denúncia".
  + **Saídas:** Mensagem de sucesso e redirecionamento para o dashboard do psicopedagogo.
  + **Regras:** Apenas denúncias com status "aberta" podem ser fechadas.
* **RF011: Exibição de Gráficos de Desempenho**
  + **Descrição:** Os dashboards de Aluno e Pais devem exibir gráficos circulares (doughnut charts) para visualização de:
    - **Frequência:** Porcentagem de presença vs. faltas.
    - **Notas:** Porcentagem de disciplinas aprovadas vs. reprovadas.
  + **Entradas:** Dados de faltas e notas do aluno.
  + **Saídas:** Gráficos interativos no dashboard.
  + **Regras:** Gráficos são gerados dinamicamente com base nos dados do aluno.
* **RF012: Regras de Aprovação/Reprovação**
  + **Descrição:** O sistema deve calcular e exibir o status de aprovação/reprovação com base nas seguintes regras:
    - **Faltas:** Reprovado se o total de faltas for superior a 8 (considerando 100 aulas padrão).
    - **Notas por Matéria:** Reprovado se a soma das duas notas for inferior a 14 (média 7).
    - **Status Final:** Reprovado se reprovado por faltas OU por notas. Detalhes se reprovado por "Faltas", "Notas" ou "Faltas e Notas".
  + **Entradas:** Total de faltas, Notas por matéria.
  + **Saídas:** Status textuais e classes CSS para destaque visual.
* **RF013: Logout do Sistema**
  + **Descrição:** Qualquer usuário logado deve poder sair do sistema, encerrando sua sessão.
  + **Entradas:** Clique no botão "Sair".
  + **Saídas:** Redirecionamento para a página inicial ou de login, e mensagem de desconexão.

**4. Requisitos Não Funcionais (RNF)**

Os requisitos não funcionais descrevem como o sistema deve se comportar.

* **RNF001: Usabilidade**
  + **Descrição:** O aplicativo deve ser intuitivo e fácil de usar para todos os perfis de usuário, com uma navegação clara e feedback visual.
  + **Métrica:** Interface de usuário limpa e responsiva.
  + **Prioridade:** Alta.
* **RNF002: Desempenho**
  + **Descrição:** O sistema deve responder rapidamente às interações do usuário, com tempos de carregamento de página mínimos.
  + **Métrica:** Tempo de resposta da página (ex: < 2 segundos para carregamento completo).
  + **Prioridade:** Média. (Para MVP em memória, é naturalmente rápido).
* **RNF003: Segurança**
  + **Descrição:** O sistema deve proteger as informações dos usuários e garantir que apenas usuários autorizados acessem dados específicos.
  + **Métrica:** Implementação de sessões seguras e controle de acesso baseado em tipo de usuário. Senhas em texto simples (MVP) devem ser substituídas por hashes em produção.
  + **Prioridade:** Alta (para produção). Média (para MVP).
* **RNF004: Manutenibilidade**
  + **Descrição:** O código-fonte deve ser bem organizado, comentado e seguir padrões de codificação para facilitar futuras modificações e extensões.
  + **Métrica:** Código modularizado (Controller, Service, Model), uso de Lombok.
  + **Prioridade:** Alta.
* **RNF005: Compatibilidade**
  + **Descrição:** O aplicativo web deve ser compatível com os navegadores web modernos (Chrome, Firefox, Edge, Safari) e ser responsivo em diferentes tamanhos de tela (desktop, tablet, mobile).
  + **Métrica:** Testes em navegadores e dispositivos variados. Uso de Media Queries no CSS.
  + **Prioridade:** Alta.
* **RNF006: Escalabilidade (Futura)**
  + **Descrição:** A arquitetura do sistema deve permitir a expansão futura para lidar com um número crescente de usuários e dados.
  + **Métrica:** Uso de Spring Boot (que facilita a integração com bancos de dados reais e deploy em nuvem).
  + **Prioridade:** Média (para MVP). Alta (para produção).
* **RNF007: Confiabilidade**
  + **Descrição:** O sistema deve operar de forma consistente e sem falhas, processando as requisições de forma correta.
  + **Métrica:** Tratamento básico de erros (mensagens flash).
  + **Prioridade:** Média.

**5. Arquitetura e Componentes**

O aplicativo AlunoSaqua segue uma arquitetura MVC (Model-View-Controller) com camadas de serviço bem definidas, típica de aplicações Spring Boot.

* **Visão Geral da Arquitetura:**
  + **Cliente (Navegador):** Envia requisições HTTP e renderiza o HTML/CSS/JS recebido.
  + **Controlador (Controller):** Recebe as requisições HTTP, processa-as, interage com a camada de Serviço e decide qual View (template HTML) exibir, passando os dados necessários.
  + **Serviço (Service):** Contém a lógica de negócio principal, cálculos, e gerencia o "banco de dados" em memória. É a camada onde as regras de negócio são aplicadas.
  + **Modelo (Model):** Classes Java que representam as entidades de dados (Aluno, Denuncia, Usuario, DadosCalculadosAluno).
* **Camada de Apresentação (Frontend - src/main/resources/templates/ e src/main/resources/static/)**
  + **Thymeleaf (.html):** Arquivos HTML com atributos th: que permitem a renderização dinâmica de dados vindos do backend. Responsável por exibir a interface do usuário.
  + **CSS (style.css):** Define a aparência visual do aplicativo, cores, fontes, layout responsivo e estilos para status (aprovado/reprovado).
  + **JavaScript:** Usado para interatividade no lado do cliente, como a geração dos gráficos com Chart.js.
* **Camada de Lógica de Negócio (Service - src/main/java/com/alunosaqua/service/)**
  + **AlunoSaquaService.java:** É o coração da lógica de negócio. Contém métodos para:
    - Autenticação de usuários (checkLogin).
    - Cadastro de alunos (registerAluno).
    - Gerenciamento de dados de alunos (getAlunoByMatricula, updateAlunoData).
    - Gerenciamento de denúncias (addDenuncia, getDenunciaById, updateDenunciaStatus).
    - Cálculos de desempenho acadêmico (calcularDadosAluno).
    - Simulação de banco de dados com static Maps.
* **Camada de Dados (In-memory - implementada dentro do Service)**
  + Para este MVP, os dados são armazenados em static HashMaps dentro do AlunoSaquaService.java. Isso significa que os dados são voláteis e são perdidos a cada reinício do servidor.
  + Em uma aplicação real, esta camada seria substituída por uma integração com um banco de dados persistente (ex: H2, MySQL, PostgreSQL) usando Spring Data JPA.
* **Camada de Controle (Controller - src/main/java/com/alunosaqua/controller/)**
  + **AlunoSaquaController.java:** Contém os métodos que respondem às requisições HTTP (anotações @GetMapping, @PostMapping).
  + Recebe os dados do formulário (@RequestParam, @PathVariable).
  + Chama os métodos apropriados da camada de Serviço para processar a lógica.
  + Adiciona dados ao objeto Model para que o Thymeleaf possa renderizá-los.
  + Redireciona o usuário para outras páginas (redirect:/).
  + Gerencia mensagens flash (RedirectAttributes).

**6. Fluxo de Funcionamento**

1. **Inicialização do Aplicativo:**
   * A classe AlunoSaquaApplication.java inicia o contêiner Spring Boot.
   * O application.properties configura a porta do servidor (8080) e o Thymeleaf.
   * O bloco static em AlunoSaquaService.java inicializa os dados de teste em memória (alunos, pais, professores, psicopedagogos, denúncias).
2. **Processo de Login:**
   * O usuário acessa http://localhost:8080/login.
   * O AlunoSaquaController exibe login.html.
   * O usuário preenche o formulário e envia via POST para /login.
   * O AlunoSaquaController recebe os dados (@RequestParam), chama service.checkLogin().
   * Se o login for bem-sucedido, a sessão é criada (HttpSession) e o usuário é redirecionado para o dashboard correspondente ao seu userType. Caso contrário, uma mensagem de erro é exibida.
3. **Processo de Cadastro (Alunos):**
   * O usuário acessa http://localhost:8080/register.
   * O AlunoSaquaController exibe register.html.
   * O aluno preenche o formulário e envia via POST para /register.
   * O AlunoSaquaController chama service.registerAluno().
   * Um novo aluno é adicionado aos dados em memória com matérias padrão inicializadas.
4. **Navegação e Dashboards:**
   * Após o login, o usuário é direcionado ao seu dashboard (/aluno/dashboard, /pais/dashboard, /professor/dashboard, /psicopedagogo/dashboard).
   * Cada rota de dashboard verifica a autenticação (checkAuth).
   * O controlador obtém os dados relevantes do service (ex: service.getAlunoByMatricula(), service.getAllAlunos()).
   * Para alunos e pais, service.calcularDadosAluno() é chamado para obter estatísticas e dados para gráficos.
   * Os dados são adicionados ao objeto Model e o Thymeleaf renderiza a página HTML correspondente, exibindo as informações dinamicamente.
5. **Atualização de Dados (Professor):**
   * No professor\_dashboard.html, o professor clica em "Atualizar" para um aluno, acessando /professor/atualizar\_dados/{matricula}.
   * O AlunoSaquaController exibe professor\_atualizar\_dados.html.
   * O professor altera faltas e/ou notas (apenas da sua disciplina se for específico, ou todas se for geral) e envia o formulário via POST.
   * O controlador chama service.updateAlunoData() com os novos valores.
   * Mensagens flash (RedirectAttributes) informam o resultado da operação.
6. **Denúncias e Gerenciamento (Psicopedagogo):**
   * Alunos usam /aluno/denunciar para enviar denúncias, que são salvas via service.addDenuncia().
   * Psicopedagogos acessam /psicopedagogo/dashboard para ver a lista de denúncias abertas, categorizadas por urgência. O nome do aluno é exibido.
   * Ao clicar em "Ver Detalhes", o psicopedagogo vai para /psicopedagogo/denuncia/{denunciaId}, onde vê todos os detalhes da denúncia e as notas do aluno denunciante.
   * Nessa página, o psicopedagogo pode marcar a denúncia como "fechada" via POST para /psicopedagogo/fechar\_denuncia/{denunciaId}, chamando service.updateDenunciaStatus().

**7. Configuração e Execução**

**Pré-requisitos:**

* **JDK 17+** instalado e configurado.
* **Apache Maven** instalado (ou use o Maven embutido do IntelliJ).
* **IntelliJ IDEA** com o **Lombok Plugin** instalado e "Annotation Processing" habilitado.

**Passos para Configurar e Rodar:**

1. **Baixe o Projeto:** Se você ainda não o fez, baixe ou clone o projeto alunosaqua-java para o seu computador.
2. **Abra no IntelliJ IDEA:**
   * Abra o IntelliJ IDEA.
   * Vá em File > Open e selecione a pasta raiz do projeto alunosaqua-java (onde está o pom.xml).
   * O IntelliJ deve detectar que é um projeto Maven e importar as dependências automaticamente. Aguarde a conclusão.
3. **Verifique o pom.xml:** Confirme que todas as dependências estão presentes e que a versão do Java (<java.version>) corresponde à sua instalação.
4. **Verifique application.properties:** Confirme que o server.port=8080 e as configurações do Thymeleaf estão corretas.
5. **Reconstrua o Projeto (Importante para Lombok e correções):**
   * Vá em Build > Clean Project.
   * Vá em Build > Rebuild Project.
6. **Execute a Aplicação:**
   * No painel Project (geralmente à esquerda), navegue até src/main/java/com/alunosaqua/AlunoSaquaApplication.java.
   * Clique com o botão direito no arquivo AlunoSaquaApplication.java.
   * Selecione Run 'AlunoSaquaApplication.main()'.
   * Aguarde no console até ver a mensagem Tomcat started on port 8080 e Started AlunoSaquaApplication.
7. **Acesse no Navegador:**
   * Abra seu navegador e vá para http://localhost:8080/.

**8. Dados de Teste (Credenciais)**

Use estas credenciais para testar as diferentes funcionalidades:

* **Alunos:**
  + Matrícula: 12345 / Senha: aluno (Rosonatt Ferreira Ramos)
  + Matrícula: 67890 / Senha: aluno (Ryan Guiwison - Reprovado por Faltas)
  + Matrícula: 11223 / Senha: aluno (Bruno Alves - Reprovado em Artes e Português)
  + Matrícula: 22334 / Senha: aluno (Natalia Crys Cardoso)
  + Matrícula: 33445 / Senha: aluno (Kevin Paula)
  + Matrícula: 44556 / Senha: aluno (Raissa Leite)
  + Matrícula: 55667 / Senha: aluno (João Candia - Reprovado em Biologia)
  + Matrícula: 66778 / Senha: aluno (Ronald Carvalho)
* **Pais:**
  + Usuário: pai\_rosanatt / Senha: pai (Filho: Rosonatt - Matrícula 12345)
  + Usuário: pai\_ryan / Senha: pai2 (Filho: Ryan - Matrícula 67890)
  + Usuário: pai\_bruno / Senha: mae (Filho: Bruno - Matrícula 11223)
  + Usuário: mae\_natalia / Senha: pais (Filho: Natalia - Matrícula 22334)
  + Usuário: pai\_kevin / Senha: pais2 (Filho: Kevin - Matrícula 33445)
  + Usuário: mae\_raissa / Senha: pais3 (Filho: Raissa - Matrícula 44556)
  + Usuário: pai\_joao / Senha: pais4 (Filho: João - Matrícula 55667)
  + Usuário: mae\_ronald / Senha: pais5 (Filho: Ronald - Matrícula 66778)
* **Professores:**
  + Usuário: prof\_mat / Senha: prof (Professor de Matemática)
  + Usuário: prof\_port / Senha: prof2 (Professor de Português)
  + Usuário: prof\_hist / Senha: prof3 (Professor de História)
  + Usuário: prof\_cienc / Senha: prof4 (Professor de Ciências)
  + Usuário: prof\_edfis / Senha: prof5 (Professor de Educação Física)
  + Usuário: prof\_geo / Senha: prof6 (Professor de Geografia)
  + Usuário: prof\_artes / Senha: prof7 (Professor de Artes)
  + Usuário: prof\_ingles / Senha: prof8 (Professor de Inglês)
* **Psicopedagogos:**
  + Usuário: psi\_joana / Senha: psi
  + Usuário: psi\_pedro / Senha: psi2
  + Usuário: coord\_educ / Senha: coord
  + Usuário: sup\_psi / Senha: sup

**9. Considerações e Melhorias Futuras**

Este aplicativo AlunoSaqua é um **MVP (Minimum Viable Product)** funcional, demonstrando as funcionalidades principais. Para uma aplicação de produção, as seguintes melhorias seriam essenciais:

* **Persistência de Dados:** Substituir o armazenamento em memória por um banco de dados real (ex: PostgreSQL, MySQL, H2) usando Spring Data JPA.
* **Segurança Robusta:** Implementar autenticação e autorização com Spring Security, incluindo hash de senhas e controle de acesso baseado em papéis (RBAC) mais granular.
* **Gerenciamento de Turmas:** Criar uma interface para administradores ou coordenadores gerenciarem turmas e associarem alunos a elas.
* **Gerenciamento de Usuários:** Interface para criar, editar e excluir contas de pais, professores e psicopedagogos.
* **Notificações:** Sistema de notificações (e-mail, push) para denúncias, atualizações de notas, etc.
* **Chat/Mensagens:** Implementar um sistema de chat para comunicação entre alunos e psicopedagogos.
* **Validação de Formulários:** Validação mais robusta no lado do servidor e do cliente.
* **Testes Automatizados:** Adicionar testes unitários e de integração para garantir a qualidade do código.
* **Deploy:** Configurar o aplicativo para deploy em um ambiente de produção (servidor web, nuvem).
* **Interface do Usuário:** Aprimorar a experiência do usuário com mais interatividade e animações.
* **Funcionalidades de IA:** Explorar a integração de IA para análise de texto em denúncias ou para oferecer suporte psicológico inicial.