ETEC ORLANDO QUAGLIATO

Técnico em Desenvolvimento de Sistemas

IDEA HALL: Desenvolvimento de uma Ferramenta Colaborativa de Validação de Ideias para Desenvolvedores de Software

CLAUDINEI DA SILVA

DANIEL MENONI FERREIRA

EMERSOM CANDIDO

JAIRSON GOES SILVA FILHO

JEOVANY CARLOS QUEIROZ

PEDRO HENRIQUE FERREIRA BARROS

Santa Cruz do Rio Pardo - SP

2025

CLAUDINEI DA SILVA

DANIEL MENONI FERREIRA

EMERSOM CANDIDO

JAIRSON GOES SILVA FILHO

JEOVANY CARLOS QUEIROZ

PEDRO HENRIQUE FERREIRA BARROS

IDEA HALL: Desenvolvimento de uma Ferramenta Colaborativa de Validação de Ideias para Desenvolvedores de Software

Trabalho de Conclusão de Curso apresentado à Etec "Orlando Quagliato", do Centro Estadual de Educação Tecnológica Paula Souza, como requisito para obtenção do título de Técnico em Desenvolvimento de Sistemas sob orientação do Professor Alan de Andrade Euzébio

Santa Cruz do Rio Pardo - SP

2025

Folha de Aprovação

CLAUDINEI DA SILVA

DANIEL MENONI FERREIRA

EMERSOM CANDIDO

JAIRSON GOES SILVA FILHO

JEOVANY CARLOS QUEIROZ

PEDRO HENRIQUE FERREIRA BARROS

Idea Hall - Desenvolvimento de uma Ferramenta de Colaborativa de Validação de Ideas para Desenvolvedores de Software

Aprovada em: _____ / _____ / _____

Conceito: _____

Banca de Validação:

_____ - Presidente da Banca
Professor.....
ETEC “Orlando Quagliato”
Orientador

_____
Professor .....
ETEC “Orlando Quagliato”

_____
Professor .....
ETEC “Orlando Quagliato”

Santa Cruz do Rio Pardo – SP
2025

RESUMO

O presente trabalho examina os desafios enfrentados por pequenas equipes de desenvolvimento de software ao tentarem transformar conhecimento técnico em iniciativas comerciais viáveis. A partir de uma análise sistemática dos segmentos acessíveis a um grupo de estudantes com habilidades voltadas ao desenvolvimento de sistemas — abrangendo tanto projetos com componente de hardware quanto soluções exclusivamente de software —, o trabalho identifica que o principal obstáculo não é de natureza técnica nem financeira, mas epistêmica: a dificuldade de determinar, antes do início do desenvolvimento, se uma ideia resolve um problema real com demanda verificável. Diante dessa lacuna, o trabalho apresenta o IdeaHall, uma ferramenta de linha de comando desenvolvida em Java com as tecnologias Picocli, GraalVM Native Image, Spring Boot e PostgreSQL, que funciona simultaneamente como repositório pessoal de ideias e plataforma colaborativa de validação coletiva. O sistema permite que desenvolvedores registrem problemas tecnológicos não resolvidos, consultem contribuições de outros usuários e identifiquem oportunidades de projeto com demanda verificada antes de comprometer tempo de desenvolvimento. A arquitetura cliente-servidor adotada foi projetada para acomodar extensões futuras — interfaces web, mobile e desktop — sem reestruturação do núcleo. O trabalho conclui que o IdeaHall representa uma abordagem inédita para a etapa anterior ao desenvolvimento de produto: a organização colaborativa e estruturada de problemas com valor de mercado real, acessível a equipes técnicas pequenas e fundamentada na validação coletiva de demanda antes da primeira linha de código.

Palavras-chave: desenvolvimento de software; empreendedorismo tecnológico; validação de produto; ferramentas de linha de comando; colaboração distribuída.

ABSTRACT

This work examines the challenges faced by small software development teams when attempting to transform technical knowledge into viable commercial initiatives. Through a systematic analysis of segments accessible to a group of students with software development skills — encompassing both hardware-component projects and purely software-based solutions —, the work identifies that the primary obstacle is neither technical nor financial in nature, but epistemic: the difficulty of determining, prior to development, whether an idea addresses a real problem with verifiable demand. To address this gap, the work presents IdeaHall, a command-line tool developed in Java using Picocli, GraalVM Native Image, Spring Boot, and PostgreSQL, which functions simultaneously as a personal idea repository and a collaborative collective validation platform. The system allows developers to register unsolved technological problems, consult contributions from other users, and identify project opportunities with verified demand before committing development time. The client-server architecture was designed to accommodate future extensions — web, mobile, and desktop interfaces — without restructuring the core. The work concludes that IdeaHall represents a novel approach to the stage that precedes product development: the collaborative and structured organization of real-market problems, accessible to small technical teams and grounded in the collective validation of demand before the first line of code is written.

Keywords: software development; technology entrepreneurship; product validation; command-line tools; distributed collaboration.

SUMÁRIO

1 INTRODUÇÃO

Nas últimas duas décadas, as formações voltadas à tecnologia deixaram de ser uma escolha de nicho para se tornarem uma das trajetórias acadêmicas mais procuradas no Brasil e no mundo. Entre 2018 e 2019, a procura por cursos dessa categoria cresceu 11,5% ao ano, respondendo por 14,5% do total de matrículas no ensino superior brasileiro no período (INSTITUTO FEDERAL DE SANTA CATARINA, 2019) — números que refletem não apenas uma tendência de mercado, mas uma mudança estrutural na forma como jovens profissionais enxergam suas possibilidades de carreira. Esse movimento não é isolado: globalmente, o mesmo padrão se repete, impulsionado pela digitalização acelerada de setores econômicos que historicamente operavam sem dependência direta de tecnologia.

O resultado desse crescimento é uma geração de profissionais tecnicamente capacitados, inseridos em um mercado que remunera bem e que apresenta demanda consistentemente superior à oferta de mão de obra qualificada. Um mercado que, segundo relatório da ONU de 2021, movimentou 350 bilhões de dólares em tecnologias emergentes naquele ano, com projeção de atingir 3,5 trilhões de dólares até 2025 (ORGANIZAÇÃO DAS NAÇÕES UNIDAS, 2021) — números que indicam não apenas escala, mas velocidade de expansão sem precedente histórico.

Nesse contexto, uma ambição recorrente entre profissionais de tecnologia em formação é a de não apenas trabalhar para esse mercado, mas participar ativamente da sua construção — desenvolvendo suas próprias soluções, sejam elas inteiramente baseadas em software ou integradas a componentes de hardware. A capacidade técnica, em muitos casos, já existe. O que frequentemente falta não é habilidade de execução, mas direção: saber identificar, antes de qualquer linha de código escrita, qual problema merece ser resolvido e porque aquela solução teria valor real para quem a utilizaria.

É exatamente essa lacuna — entre a capacidade técnica de construir e o conhecimento sobre o que vale a pena construir — que este trabalho se propõe a examinar e a endereçar. A pergunta que orienta todo o desenvolvimento a seguir é simples na forma e complexa na resposta: é possível que um pequeno grupo de pessoas participe desse mercado de forma estruturada, com recursos limitados e sem infraestrutura corporativa — e, se sim, como encontrar o caminho certo para fazê-lo?

2 REFERENCIAL TEÓRICO

O referencial teórico deste trabalho difere do modelo convencional adotado em pesquisas acadêmicas de natureza bibliográfica. Em vez de se apoiar primariamente em obras da literatura de administração ou empreendedorismo, o embasamento central deste TCC é constituído pela documentação oficial das tecnologias que compõem o IdeaHall. Essa escolha é metodologicamente coerente com a natureza do trabalho: um TCC técnico cujo entregável principal é um sistema funcional deve ter como referência primária as especificações, comportamentos e limitações das ferramentas que tornam esse sistema possível.

2.1 Picocli

A documentação oficial do Picocli, disponível em picocli.info, constitui a referência central para todas as decisões relacionadas à construção da interface de linha de comando do IdeaHall (PICOCLI, 2026). Ela especifica o modelo de anotações Java utilizado para declarar comandos, sub-comandos e argumentos; o mecanismo de geração automática de mensagens de ajuda; e o processo de configuração para compilação com GraalVM Native Image, incluindo a geração automática do reflect-config.json via annotation processor.

A documentação do Picocli é particularmente relevante para este trabalho por tratar explicitamente do problema de experiência do usuário em CLIs — tema central na filosofia de design do IdeaHall. Ela documenta não apenas o comportamento técnico da biblioteca, mas os princípios que orientam a construção de interfaces de linha de comando simultaneamente poderosas e acessíveis, servindo como referência direta para as decisões de design da interface descritas no capítulo de desenvolvimento (PICOCLI, 2026).

2.2 GraalVM Native Image

A documentação do GraalVM Native Image, mantida pela Oracle e disponível em graalvm.org, fundamenta as decisões relacionadas à compilação e distribuição do CLI (GRAALVM, 2026). Ela especifica o processo de compilação ahead-of-time que

transforma o bytecode Java em um binário nativo sem dependência de JVM em tempo de execução, os requisitos de configuração para reflexão que precisam ser declarados explicitamente para o correto funcionamento do binário gerado, e as instruções de integração com o Picocli para geração automática dessas configurações (GRAALVM NATIVE BUILD TOOLS, 2026).

A relevância dessa documentação para o IdeaHall vai além do aspecto técnico. Ela fundamenta uma das decisões arquiteturais mais importantes do projeto — a distribuição como binário nativo único — e fornece os critérios técnicos que a justificam em termos de tempo de inicialização e ausência de dependência de ambiente na máquina do usuário final (GRAALVM, 2026).

2.3 Spring Boot

A documentação oficial do Spring Boot, disponível em docs.spring.io, é a referência central para a construção da camada de aplicação do IdeaHall (SPRING, 2026). Ela cobre a configuração por convenção que caracteriza o framework, o gerenciamento do ciclo de vida dos beans via injeção de dependência, o gerenciamento de transações via @Transactional, e a comunicação com a camada de persistência via Spring Data JPA.

No contexto específico do IdeaHall, a documentação do Spring Boot fundamenta a bridge entre Spring e Picocli implementada pela interface IFactory — mecanismo pelo qual o Picocli delega a instanciação de comandos ao contexto Spring, habilitando injeção de dependência completa na camada CLI e garantindo que todos os sub-comandos operem sobre as mesmas instâncias singleton dos serviços (SPRING, 2026).

2.4 Spring Data JPA e Hibernate

A documentação do Spring Data JPA, parte do ecossistema Spring disponível em docs.spring.io/spring-data, fundamenta as decisões relacionadas à camada de repositórios do sistema (SPRING, 2026). Ela especifica o mecanismo de derivação automática de queries a partir de nomes de métodos, o uso de anotações @Query com JPQL para operações de atualização direta, e o

comportamento do parâmetro clearAutomatically em operações @Modifying para evitar inconsistências no cache de primeiro nível do Hibernate após atualizações em massa.

2.5 PostgreSQL

A documentação oficial do PostgreSQL, disponível em postgresql.org/docs, fundamenta as decisões relacionadas à camada de persistência do sistema (POSTGRESQL, 2026). Ela especifica o comportamento do controle de concorrência multiversionado — MVCC — que garante consistência de dados em cenários de acesso simultâneo, os tipos de dados e constraints disponíveis, e as práticas recomendadas para modelagem de esquemas relacionais com integridade referencial — incluindo o comportamento de chaves primárias compostas que fundamenta o mecanismo de controle de votos duplicados do IdeaHall.

2.6 Maven

A documentação do Apache Maven, disponível em maven.apache.org, constitui a referência para o sistema de build do projeto (APACHE, 2026). Ela especifica a configuração do ciclo de build via pom.xml, o gerenciamento declarativo de dependências, e a integração com o plugin native-maven-plugin do GraalVM que automatiza a compilação nativa como fase adicional do ciclo de empacotamento.

2.7 Sobre o Papel das Referências Não Técnicas

As obras de Peter Thiel, Eric Ries e Adam Smith referenciadas em outros capítulos deste trabalho cumprem um papel distinto e secundário em relação às documentações técnicas listadas acima. Elas são utilizadas para enquadrar conceitualmente o problema que o IdeaHall resolve — a dificuldade de identificar o que vale a pena construir antes de construir — e para contextualizar o projeto dentro de um debate mais amplo sobre inovação e criação de produtos (THIEL; MASTERS, 2014; RIES, 2012). Não fundamentam decisões técnicas de implementação e não

constituem, portanto, o referencial central deste trabalho. São lentes conceituais, não especificações.

O referencial que de fato orientou cada decisão de arquitetura, cada escolha de biblioteca e cada comportamento documentado nos comandos do IdeaHall está nas documentações descritas acima — lidas, consultadas e aplicadas diretamente ao longo de todo o processo de desenvolvimento.

3 METODOLOGIA

Este capítulo descreve as escolhas metodológicas que orientaram a concepção e o desenvolvimento do IdeaHall, abrangendo o processo de identificação do problema, a seleção do conjunto tecnológico, as decisões arquiteturais tomadas ao longo da implementação e a abordagem de validação adotada pela equipe.

3.1 Natureza da Pesquisa

A metodologia deste trabalho classifica-se como pesquisa aplicada de natureza experimental. O objetivo central não é produzir conhecimento teórico generalizável, mas construir um artefato funcional que resolva um problema concreto e documentar, com rigor, as decisões que tornaram essa construção possível. A abordagem é predominantemente qualitativa na fase de identificação do problema e técnica na fase de implementação, onde as decisões são avaliadas por critérios objetivos de funcionalidade, desempenho e adequação arquitetural.

3.2 Identificação do Problema

A identificação da lacuna que o IdeaHall preenche não foi conduzida por revisão bibliográfica formal, mas por um processo de observação direta e uso sistemático das ferramentas existentes no ecossistema. Plataformas como GitHub, Stack Overflow, Product Hunt e Indie Hackers foram analisadas sob uma perspectiva funcional específica: a de uma equipe técnica de pequeno porte buscando problemas não resolvidos com demanda de mercado verificável antes de iniciar o desenvolvimento.

Essa análise revelou, de forma consistente, que nenhuma das plataformas existentes foi projetada para resolver esse problema com a especificidade necessária. O GitHub organiza repositórios, não demandas. O Stack Overflow registra dúvidas técnicas, não oportunidades de produto. O Product Hunt e o Indie Hackers operam na fase de lançamento, não na fase anterior à decisão de construir. A lacuna identificada é, portanto, estrutural — não uma limitação de implementação das plataformas existentes, mas uma ausência de categoria.

3.3 Definição do Escopo

Uma vez identificado o problema, a equipe conduziu um processo deliberado de delimitação de escopo. O princípio orientador foi a entregabilidade com qualidade: um sistema restrito e funcional é mais valioso, para os objetivos deste trabalho, do que um sistema amplo e incompleto.

As funcionalidades incluídas na versão inicial foram selecionadas por um critério único: são as que diretamente endereçam o problema identificado. Autenticação de usuários, publicação de ideias, mecanismo de sinalização de interesse e controle de acesso por autoria compõem o conjunto mínimo necessário para que o ciclo colaborativo proposto pelo IdeaHall seja operacional. Todas as demais funcionalidades previstas — interface web, aplicativos mobile, módulo de gerenciamento de projetos, modelo de patrocínio — foram conscientemente alocadas para fases futuras de desenvolvimento.

3.4 Seleção Tecnológica

A escolha das tecnologias que compõem o IdeaHall foi guiada por três critérios objetivos: adequação técnica ao problema, qualidade da documentação oficial disponível e alinhamento com o perfil técnico da equipe.

Java 21 foi adotado como linguagem principal pela portabilidade entre sistemas operacionais, pela natureza orientada a objetos adequada ao modelo de domínio do sistema, e pelo ecossistema de bibliotecas maduras disponíveis para todas as necessidades do projeto. A documentação oficial de cada tecnologia foi tratada como fonte primária de conhecimento técnico ao longo de todo o desenvolvimento — não como referência secundária. Essa distinção é metodologicamente relevante: reflete a forma como o conhecimento técnico é efetivamente adquirido e aplicado em contextos reais de desenvolvimento de software.

Picocli foi selecionado para a construção da interface de linha de comando pela API declarativa baseada em anotações, que reduz o volume de código de infraestrutura e concentra o desenvolvimento na lógica de negócio, e pela compatibilidade documentada com compilação nativa via GraalVM (PICOCLI, 2026) — decisão que seria tomada posteriormente no ciclo de desenvolvimento.

Spring Boot foi adotado como framework de aplicação pela filosofia de configuração por convenção, que elimina boilerplate de infraestrutura, e pela integração nativa com Spring Data JPA, que simplifica o acesso à camada de persistência sem abrir mão do controle sobre as operações de banco de dados (SPRING, 2026).

GraalVM Native Image foi incorporado ao projeto em resposta a um problema concreto identificado durante os primeiros testes: o tempo de inicialização da JVM convencional, da ordem de centenas de milissegundos, era inaceitável para uma ferramenta de linha de comando interativa. A migração para compilação nativa reduziu o tempo de inicialização (GRAALVM, 2026) — decisão emergente do ciclo de desenvolvimento, não prevista na especificação inicial.

PostgreSQL foi escolhido como sistema de gerenciamento de banco de dados pela robustez em ambientes de múltiplos usuários concorrentes via controle de concorrência multiversionado, pela maturidade do ecossistema de integrações com Spring Data JPA, e pela disponibilidade em plataformas de hospedagem gerenciadas adequadas ao estágio inicial do projeto (POSTGRESQL, 2026).

3.5 Processo de Desenvolvimento

O desenvolvimento do IdeaHall seguiu um ciclo iterativo composto por três fases recorrentes: especificação de requisito, implementação e observação de comportamento. Esse processo de tentativa, observação e ajuste constitui o ciclo metodológico central da parte prática deste trabalho — documentado não como desvio de um método mais rigoroso, mas como o método apropriado para o tipo de problema abordado: um sistema novo, sem precedente direto, construído por uma equipe pequena em ambiente de recursos limitados.

Várias decisões arquiteturais relevantes emergiram diretamente desse ciclo, e não de especificações prévias. A adoção do GraalVM Native Image, como descrito na seção anterior, foi uma resposta ao problema concreto de latência observado em testes (GRAALVM, 2026). A separação entre modo de execução padrão e modo contínuo via -dry_run emergiu da necessidade prática de suportar sessões interativas com estado de autenticação persistente entre comandos — necessidade que se tornou evidente apenas durante a implementação do fluxo de login. A centralização

do Scanner sobre System.in como singleton no SessionContext foi uma correção a um bug identificado na análise do código: dois Scanner instanciados sobre o mesmo InputStream competem pelos bytes do stream, produzindo leituras não determinísticas — problema que só se manifestou ao testar os comandos de autenticação dentro do modo contínuo.

O ciclo de desenvolvimento também envolveu múltiplas rodadas de revisão estática do código — análise sistemática de cada fluxo de execução sem execução real do programa, dado que o ambiente de desenvolvimento não dispunha de PostgreSQL para testes de integração (POSTGRESQL, 2026). Esse processo produziu a identificação e correção de três categorias de bugs:

- Primeiro era uma falha na inicialização do sistema de comandos - que fazia com que os componentes não fossem reconhecidos corretamente ao entrar no modo contínuo.
- O segundo era um erro que desativava acidentalmente o sistema de tratamento de erros da aplicação - fazendo com que falhas não fossem capturadas e exibidas corretamente ao usuário.
- Terceiro era uma ineficiência no processo de exclusão de ideias, onde o sistema buscava todos os votos do banco de dados antes de deletá-los, em vez de remover diretamente apenas os relacionados à ideia em questão.

3.6 Decisões de Design de Interface

O design da interface de linha de comando partiu de uma premissa explícita: o terminal não precisa ser hostil. Ferramentas como Git e Docker demonstram que CLIs podem ser ao mesmo tempo poderosas e acessíveis. O IdeaHall adotou esse referencial em três dimensões:

- A primeira é a clareza das mensagens. Toda saída produzida pelo sistema — confirmações, erros e orientações — foi escrita para comunicar com precisão o que ocorreu e, quando aplicável, o que o usuário pode fazer a seguir. Mensagens genéricas e stack traces foram substituídos por texto em linguagem

natural, interceptados pelo handler global de exceções antes de atingir o terminal.

- A segunda é a previsibilidade estrutural. O comando -show_idea produz sempre a mesma estrutura de saída, com campos nomeados e alinhados, facilitando tanto a leitura humana quanto o processamento por ferramentas assistivas de terminal.
- A terceira é o feedback imediato. Nenhum comando encerra silenciosamente — o usuário sempre recebe uma resposta que confirma o sucesso ou descreve o problema, seguindo o princípio de feedback imediato das heurísticas de usabilidade de Nielsen (NIELSEN, 1994).

3.7 Construção da Documentação

A documentação do IdeaHall foi produzida em paralelo ao desenvolvimento, não como etapa posterior. Essa decisão metodológica tem uma consequência prática relevante: a documentação reflete o sistema como ele foi construído, incluindo as decisões que mudaram no meio do caminho, e não uma versão idealizada da proposta original.

O site de documentação foi desenvolvido como artefato HTML único, sem dependências de frameworks externos, em conformidade com as diretrizes WCAG 2.1 AA de acessibilidade e com design responsivo para dispositivos móveis. Sua estrutura foi planejada para suportar expansão futura quando a versão web da plataforma for incorporada, evitando retrabalho na base já construída.

3.8 Validação

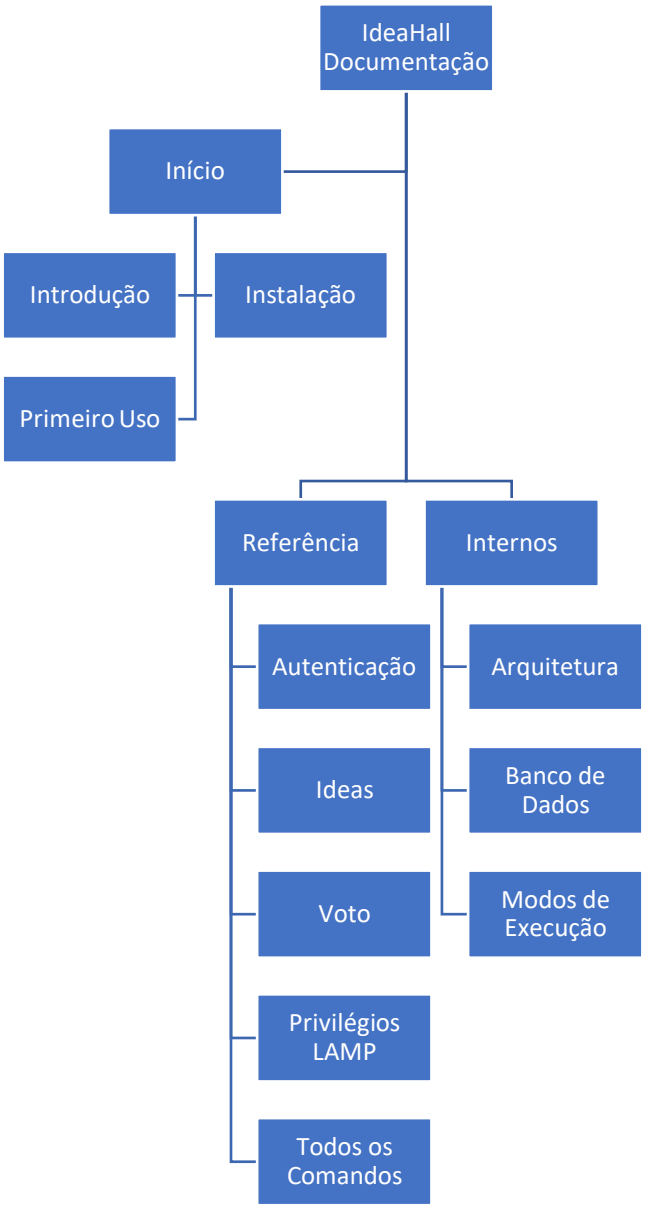
A validação do sistema foi conduzida em duas dimensões complementares:

- A validação técnica avaliou o comportamento funcional do sistema: integridade das operações de leitura e escrita no banco de dados, comportamento dos fluxos de comando sob diferentes entradas e condições de erro, e correção das verificações de acesso por autoria. Essa validação foi realizada por análise estática do código, rastreando cada fluxo de execução do entry point até a

camada de persistência, identificando os pontos onde invariantes de negócio poderiam ser violados.

- A validação conceitual avaliou a aderência do sistema ao problema que motivou sua criação. O critério foi direto: o IdeaHall produz, de forma operacional, o tipo de insumo que o cenário descrito nos capítulos iniciais demanda? Um desenvolvedor pode publicar uma ideia, outros usuários podem sinalizar interesse nela, e o autor pode consultar esse interesse antes de decidir se vale investir tempo no desenvolvimento? A resposta afirmativa a essas três perguntas constitui a principal evidência de validação conceitual disponível nesta fase do projeto.

4 MAPA DO SITE



O site de documentação do IdeaHall é composto por um único arquivo HTML, sem dependências de frameworks externos. Sua estrutura é dividida em duas áreas principais: uma barra lateral fixa de navegação à esquerda, com links organizados em três grupos — Início, Referência e Internos — e uma área de conteúdo principal à direita, onde as seções são carregadas de forma contínua com scroll suave.

O layout utiliza fontes do Google Fonts — Space Mono para código e elementos técnicos, e Syne para textos corridos — com paleta de cores escura baseada em tons de verde-escuro e verde-neon como cor de destaque. As seções de conteúdo incluem

blocos de código com botão de cópia, tabelas de comandos, listas numeradas em formato de passos, callouts informativos e um diagrama de banco de dados em formato de grade. O site conta com animações de entrada por fadeUp nas seções, highlight automático do item ativo na navegação via scroll, e um menu colapsável para dispositivos móveis, seguindo o modelo Mobile First com três faixas de responsividade. A acessibilidade é tratada com skip link, navegação por teclado e conformidade com WCAG 2.1 AA.

4.1 Mapa do Sistema

O diagrama apresenta a arquitetura interna do IdeaHall em formato de mapa de dependências. No topo, o IdeaHallApplicationSpring atua como ponto de entrada

da aplicação, inicializando o contexto Spring e delegando o controle ao IdeaHallCommand, que é o comando raiz do Picocli e coordena todos os sub-comandos.

O IdeaHallCommand se conecta a dois componentes centrais:

- o SessionContext, responsável por manter o estado de autenticação do usuário durante a sessão, e o PicocliConfig, que configura a bridge entre o Picocli e o Spring via IFactory, garantindo que os comandos recebam injeção de dependência corretamente;
- Na ramificação de autenticação, o AuthService gerencia as operações de login, cadastro e logout, acessando os dados de usuários por meio do UserRepository, que persiste as informações na user_table do PostgreSQL.

Na ramificação de ideias, o IdeaService centraliza toda a lógica de negócio relacionada às ideias — criação, consulta, modificação e exclusão —, acessando duas tabelas distintas: a idea_table via IdeaRepository, que armazena os dados das ideias, e a voters_table via IdeaVoteRepository, que controla os votos registrados por cada usuário, impedindo duplicidades por chave primária composta.

5 CUSTOS, DOMÍNIOS E HOSPEDAGEM

A análise de centro de custo do IdeaHall deve ser conduzida separando três dimensões distintas: o registro do nome de domínio, a hospedagem do banco de dados PostgreSQL e a hospedagem do site de documentação — cada uma com estrutura de custo e periodicidade de cobrança próprias.

5.1 Considerações Arquiteturais Sobre Hospedagem

O CLI do IdeaHall é um binário que executa localmente na máquina do usuário — não há custo de hospedagem para o executável em si. O componente que demanda infraestrutura remota é exclusivamente o banco de dados PostgreSQL. O site de documentação é um artefato HTML estático, sem servidor de aplicação, sem banco de dados e sem processamento dinâmico, enquadrando-se na categoria de static hosting. A esses dois componentes soma-se o registro do domínio — um custo recorrente independente da hospedagem, necessário para que a plataforma possua um endereço próprio na internet.

5.2 Registro de Domínio

O registro de um nome de domínio é cobrado anualmente e é independente da hospedagem — domínio e hospedagem são serviços distintos, ainda que frequentemente comercializados em conjunto por uma mesma empresa.

Para domínios nacionais sob a extensão .com.br, o Registro.br — entidade responsável pelo registro de domínios .br no Brasil — pratica o valor fixo de R$ 40,00 por ano, tanto no registro inicial quanto em todas as renovações subsequentes, sem variação progressiva de preço (REGISTRO.BR, 2026). Essa estabilidade de preço é uma vantagem relevante frente a registradoras privadas, que frequentemente praticam valores promocionais no primeiro ano seguidos de aumento expressivo na renovação.

Para domínios internacionais como .com, .dev ou .io — relevantes caso o IdeaHall opte por um domínio sem vínculo geográfico explícito, alinhado à proposta de alcance internacional descrita nas considerações finais — o Cloudflare

Registrar oferece registro ao preço de custo (at-cost), sem markup sobre o valor cobrado pelo registry, com privacidade WHOIS incluída sem custo adicional (CLOUDFLARE, 2026). O valor de um domínio .com no Cloudflare Registrar gira em torno de US$ 10,00 a US$ 11,00 por ano.

A tabela a seguir sintetiza as opções de registro de domínio relevantes para o projeto:

Tabela 1 – Comparativo de custos de registro de domínio

Tipo de domínio	Registradora recomendada	Custo anual	Observação
.com.br	Registro.br	R$ 40,00 R$ 40,00	Preço fixo, sem variação na renovação
.com / .dev / .io	Cloudflare Registrar	US$ 10,00 – US$ 11,00	Preço de custo, privacidade WHOIS inclusa

Fonte: elaborado pelo autor (2026), com base em Registro.br (2026) e Cloudflare (2026).

5.3 Hospedagem do Banco de Dados PostgreSQL

O PostgreSQL é o componente de maior criticidade operacional do sistema. Sua disponibilidade determina diretamente a disponibilidade da plataforma — uma instância fora do ar implica indisponibilidade total para todos os usuários conectados.

Para o estágio atual do projeto, três plataformas de banco de dados gerenciado se destacam por oferecerem camadas gratuitas adequadas ao volume de dados esperado em fase inicial:

Supabase oferece uma instância PostgreSQL gerenciada com 500MB de armazenamento, 2 CPUs compartilhadas e 1GB de RAM na camada gratuita, com pausa automática após sete dias de inatividade. O custo da camada paga inicia em US$ 25,00 mensais, com 8GB de RAM dedicada e 250GB de armazenamento — adequado para uma base de usuários de pequeno a médio porte. A plataforma oferece ainda uma dashboard administrativa e backups automáticos diários na camada paga.

Railway oferece US$ 5,00 de crédito mensal gratuito, suficiente para uma instância PostgreSQL de baixo tráfego. A cobrança é baseada em uso efetivo de recursos — CPU, memória e transferência de dados — o que torna o custo proporcional ao crescimento da plataforma. Uma instância PostgreSQL com 1GB de RAM e armazenamento de 1GB consome aproximadamente US$ 5,00 a US$ 10,00 mensais em tráfego moderado.

oferece uma instância PostgreSQL gratuita com 256MB de RAM e 1GB de armazenamento, com expiração após 90 dias. A camada paga inicia em US$ 7,00 mensais para instâncias com 256MB de RAM e armazenamento expandível.

Para a versão futura com API REST centralizada, adiciona-se o custo de hospedagem do servidor Spring Boot. Uma instância com 512MB de RAM — adequada para o volume de requisições esperado em fase inicial — custa aproximadamente US$ 7,00 mensais no Render, US$ 5,00 a US$ 10,00 no Railway, e US$ 6,00 em um Droplet básico na DigitalOcean.

A tabela a seguir sintetiza os custos estimados para o sistema backend em três cenários de estágio:

Tabela 02 — Estimativa de custos de hospedagem do sistema IdeaHall por estágio de maturidade

Estágio	Componentes	Plataforma recomendada	Custo estimado/mês
Acadêmico / MVP	PostgreSQL apenas	Supabase (free tier)	US$ 0,00
Lançamento inicial	PostgreSQL + API REST	Railway	US$ 10,00 – US$ 20,00
Crescimento	PostgreSQL gerenciado + API escalável	Supabase Pro + Render	US$ 32,00 – US$ 60,00

Fonte: Elaborado pelo autor (2026).

5.4 Hospedagem do Site de Documentação

O site de documentação é um arquivo HTML estático sem dependências de backend, o que o torna elegível para plataformas de static hosting — categoria com múltiplas opções gratuitas de alta disponibilidade e desempenho.

GitHub Pages é a opção de menor atrito operacional: o arquivo ideahall-docs.html pode ser servido diretamente de um repositório público no GitHub sem configuração adicional, com HTTPS automático e CDN global. O custo é zero para projetos públicos, sem limitações de tráfego para sites de documentação técnica.

Cloudflare Pages oferece hospedagem estática com rede de distribuição em mais de 300 pontos de presença globais, HTTPS automático, e limites generosos na camada gratuita — 500 builds mensais e largura de banda ilimitada. Para um site de documentação com tráfego concentrado em desenvolvedores, a distribuição global do Cloudflare reduz a latência de carregamento de forma mensurável.

Netlify e Vercel oferecem camadas gratuitas equivalentes com deploy automático via integração com repositórios Git, previews de pull requests e domínios customizados com HTTPS. Ambas são adequadas para o estágio atual do projeto sem custo.

Para a expansão futura do site quando a versão web do IdeaHall for incorporada — introduzindo rotas dinâmicas, autenticação e chamadas à API — a hospedagem migrará da categoria de static hosting para um servidor de aplicação, com estrutura de custo equivalente à descrita na seção anterior.

Tabela 03 — Comparativo de plataformas de hospedagem estática para o site de documentação

Plataforma	Camada Gratuita	CDN	Custo Pago
GitHub Pages	Ilimitado (projeto público)	SIM	N/A
Cloudflare	Largura de banda ilimitada	Sim (300+ PoPs)	USD$ 20,00/mês
Netlify	100GB transferência/mês	SIM	USD$ 19,00/mês
Versel	100GB transferência/mês	SIM	USD$ 20,00/mês

Fonte: Elaborado pelo autor (2026).

Para o estágio atual do projeto, GitHub Pages ou Cloudflare Pages são as recomendações técnicas — custo zero, desempenho adequado e zero complexidade operacional.

5.5 Custo Total Consolidado

Considerando o estágio de entrega deste TCC, o custo total de infraestrutura do IdeaHall é zero, uma vez que o domínio ainda não foi registrado nesta fase acadêmica, o banco de dados opera na camada gratuita do Supabase, e o site de documentação é servido via GitHub Pages ou Cloudflare Pages sem custo.

Para o estágio de lançamento comercial previsto nas considerações finais, o custo operacional estimado passa a incluir o registro de domínio (R$ 40,00/ano ou aproximadamente US$ 10,00/ano, conforme a extensão escolhida) somado à hospedagem de banco de dados e servidor de aplicação, projetando um custo total de infraestrutura entre US$ 30,00 e US$ 60,00 mensais — valor compatível com o modelo de sustentabilidade descrito no capítulo de proposta do sistema.

6 DA IDEIA À VIABILIDADE: O DESAFIO DE EMPREENDER EM TECNOLOGIA

O crescimento do setor de TI reduziu as barreiras para jovens empreendedores criarem soluções de software. Este capítulo acompanha o raciocínio de um grupo hipotético de três estudantes de computação que, ao decidir transformar seus conhecimentos em um negócio, precisa responder a uma pergunta central: em qual ideia vale a pena investir tempo?

6.1 Projetos com Componente de Hardware

O grupo analisou seis segmentos que envolvem hardware, identificando em cada um suas oportunidades e limitações centrais:

- Automação residencial e empresarial: mercado em expansão, mas exige conhecimentos distantes do perfil de desenvolvimento do grupo — CLPs, protocolos industriais, eletrônica aplicada.
- Suporte técnico e manutenção: demanda constante, porém margens baixas no atendimento ao público e exigência de credibilidade prévia no segmento corporativo.
- Dispositivos eletrônicos de baixa complexidade: viável tecnicamente com plataformas como ESP32, mas a homologação pela Anatel e os desafios de escala o tornam mais adequado como horizonte de médio prazo.
- Redes e infraestrutura: diferenciação sustentável exige estrutura operacional de empresa de TI gerenciada, incompatível com uma equipe de três pessoas no início.
- Segurança eletrônica: oportunidades de integração interessantes, mas com exigências legais e responsabilidade técnica elevadas desde o início da operação.
- Fabricação de hardware customizado: alto potencial de propriedade intelectual, mas o salto do protótipo ao produto comercializável é o gargalo mais crítico do setor.

6.2 Projetos de Software

Quatro segmentos puramente digitais foram analisados, todos com barreiras de entrada menores:

- Aplicações web e mobile para nichos: grande potencial, mas o risco é crescer em funcionalidades enquanto a base de usuários permanece estagnada — a validação de demanda é tão importante quanto a qualidade técnica.
- Ferramentas para desenvolvedores: vantagem de resolver problemas próprios, mas monetização é historicamente difícil dado o histórico de gratuidade da cultura open source.
- Automação de processos e integrações: modelo de projeto sob demanda é ideal para o início, mas existe o risco de permanecer em consultoria sem evoluir para um produto escalável.
- Sistemas de gestão para pequenos negócios: forte aderência ao mercado local, mas gera dependência operacional dos clientes e exige suporte contínuo difícil de sustentar em equipe pequena.

6.3 O Problema que Emerge

Ao percorrer todos os segmentos, o grupo percebeu que a limitação mais recorrente nos projetos de software não era técnica nem financeira — era epistêmica: o grupo sabe programar, mas não sabe o que construir. Mais precisamente, não sabe como determinar, antes de investir meses de trabalho, se uma ideia resolve um problema real pelo qual alguém pagaria.

Metodologias como Lean Startup, Jobs to be Done e Design Thinking existem para endereçar exatamente esse problema (RIES, 2012; CHRISTENSEN et al., 2016; BROWN, 2010), mas exigem um investimento metodológico em pesquisa de campo que uma equipe pequena, sem rede corporativa estabelecida, dificilmente consegue aplicar com profundidade.

Plataformas como GitHub, Stack Overflow e Product Hunt organizam código, dúvidas e lançamentos, respectivamente — mas nenhuma foi construída para

organizar problemas tecnológicos não resolvidos de forma estruturada, consultável e verificável antes da primeira linha de código.

6.4 Síntese

O grupo não chegou ao final do processo com uma ideia escolhida, mas com algo mais valioso: a identificação precisa do problema anterior à escolha. Antes de decidir o que construir, é necessário ter acesso a uma fonte confiável de problemas reais com demanda verificável. É dessa lacuna que nasce a proposta do capítulo seguinte.

7 IDEAHALL: CONCEPÇÃO, ARQUITETURA E ESPECIFICAÇÃO TÉCNICA

O capítulo anterior identificou a lacuna central que este trabalho se propõe a endereçar: antes de escolher o que construir, é necessário ter acesso a um mecanismo estruturado para identificar o que vale a pena construir. É a partir dessa constatação que nasce o IdeaHall.

7.1 Proposta do Sistema

O IdeaHall é uma ferramenta de linha de comando com duplo propósito. No plano individual, funciona como um repositório pessoal de ideias — um espaço estruturado onde o desenvolvedor registra, organiza e refina percepções sobre problemas que pretende resolver. No plano coletivo, funciona como uma plataforma colaborativa onde essas ideias são compartilhadas com outros usuários, que podem validá-las por meio de um mecanismo de sinalização de interesse — denominado UP — sem alterar seu conteúdo.

A escolha da interface de linha de comando como vetor principal de acesso é deliberada. Desenvolvedores têm familiaridade profunda com ferramentas como Git, npm e Docker — todas operadas via terminal. Para esse perfil de usuário, um CLI bem construído não é uma limitação de interface, mas uma preferência: é rápido, scriptável e integrável a fluxos de trabalho existentes.

7.2 Escopo da Versão Inicial

A versão entregue neste trabalho contempla as seguintes funcionalidades: criação e armazenamento de ideias com campos de nome, categoria, descrição e autoria; sistema de autenticação por sessão com cadastro e login de usuários; mecanismo de validação coletiva via UP com controle de duplicidade; e interface de linha de comando interativa com dois modos de execução — single-shot e modo contínuo -dry_run.

Extensões previstas — interface web, aplicativos mobile e cliente desktop — estão fora do escopo desta entrega. A arquitetura foi, no entanto, projetada para acomodá-las sem reestruturação do núcleo.

7.3 Decisões de Arquitetura

O IdeaHall é estruturado em quatro camadas verticais com responsabilidades bem delimitadas: entrada, serviços, repositórios e banco de dados, seguindo o padrão arquitetural em camadas amplamente adotado em aplicações Spring Boot.

A linguagem de implementação é Java 21, escolhida pela portabilidade entre sistemas operacionais, pela adequação ao modelo de domínio orientado a objetos, e pela disponibilidade de bibliotecas maduras para todas as necessidades do projeto:

- O Picocli é a biblioteca responsável pela interface de linha de comando — parsing de argumentos, geração automática de mensagens de ajuda e suporte nativo à compilação com GraalVM Native Image (PICOCLI, 2026).
- O GraalVM Native Image resolve o problema estrutural do Java em contextos de CLI: o tempo de inicialização da JVM, que pode chegar a um segundo em aplicações convencionais, é eliminado pela compilação ahead-of-time, produzindo um binário nativo que inicializa em menos de cinquenta milissegundos e é distribuível sem exigir instalação prévia de Java (GRAALVM, 2026).
- O Spring Boot é responsável pelo gerenciamento do ciclo de vida da aplicação, pela injeção de dependências, pelo gerenciamento de transações e pela integração com a camada de persistência via Spring Data JPA. A bridge entre Spring e Picocli, implementada pela interface IFactory, garante que todos os sub-comandos operem sobre as mesmas instâncias singleton dos serviços — condição necessária para que o estado de autenticação persista corretamente entre comandos consecutivos (SPRING, 2026).
- O PostgreSQL gerencia a persistência do sistema. Sua adoção é orientada pela robustez em ambientes de múltiplos usuários concorrentes via MVCC, pela maturidade do ecossistema de integrações com Spring Data JPA, e pela disponibilidade em plataformas de hospedagem gerenciadas adequadas ao estágio inicial do projeto (POSTGRESQL, 2026).

7.4 Modelo de Permissões

O IdeaHall adota um modelo de permissões orientado por dois princípios. O princípio de transparência determina que ideias publicadas são visíveis a qualquer usuário autenticado. O princípio de integridade autoral determina que modificações no conteúdo de uma ideia são restritas ao seu criador — denominado LAMP — enquanto outros usuários têm acesso de leitura e validação. Esse modelo binário é suficiente para o escopo da versão inicial e pode ser expandido com papéis adicionais em versões futuras.

7.5 Propriedade Intelectual

Por design, o sistema não adota precedência autoral: o registro de uma ideia não confere ao usuário direito exclusivo sobre ela. Ideias publicadas são contribuições ao bem comum da comunidade técnica — seu valor está na visibilidade e na validação coletiva, não na exclusividade de posse. Um termo de uso formal que estabeleça direitos e responsabilidades de publicação e consumo de ideias está previsto como entregável de versão futura.

7.6 Módulo de Gerenciamento de Projetos

Além do repositório de ideias, o IdeaHall contempla um segundo módulo funcional para o ciclo de vida posterior à ideia: o gerenciamento de projetos em desenvolvimento ativo. Ele permite que equipes registrem projetos em andamento, direcionem colaboradores para papéis específicos e publiquem atualizações progressivas sobre o estado do desenvolvimento. Os comandos específicos desse módulo serão documentados conforme sua implementação for concluída.

7.7 Sustentabilidade e Expansão

A versão inicial opera como plataforma de acesso aberto sem monetização ativa. O mecanismo previsto para versões futuras é o patrocínio de ideias: usuários

ou organizações interessadas em ver uma ideia transformada em produto poderiam aportar recursos para patrocinar seu desenvolvimento, com a plataforma retendo uma parcela para cobrir custos operacionais e repassando o restante aos desenvolvedores responsáveis pela execução. Esse mecanismo fecha o ciclo proposto pela plataforma — da identificação do problema à entrega da solução — com sustentabilidade econômica integrada.

7.8 Posicionamento e Acessibilidade

O IdeaHall, em sua versão inicial, é fundamentalmente uma ferramenta para desenvolvedores e equipes técnicas. As limitações inerentes à interface de linha de comando — ausência de recursos visuais elaborados e pressuposto de familiaridade com o terminal — são reconhecidas como limitações reais, mas representam uma escolha de posicionamento consciente e adequada ao público-alvo. O alcance a perfis de usuário sem familiaridade com terminais pertence ao escopo das extensões futuras, que serão desenvolvidas sobre a mesma base arquitetural entregue neste trabalho.

8 GUIA DE USO DO IDEAHALL, INSTALAÇÃO E GERENCIAMENTO DE DEPENDÊNCIAS

8.1 Dependências Necessárias

O IdeaHall requer três dependências externas instaladas no ambiente antes da execução: Java 21, Maven e PostgreSQL. As demais dependências da aplicação — Spring Boot, Picocli, Hibernate, BCrypt e o driver JDBC do PostgreSQL — são gerenciadas automaticamente pelo Maven e baixadas na primeira compilação.

8.1.1 Java 21

O Java Development Kit 21 é o requisito de runtime da aplicação. A distribuição recomendada é o Eclipse Temurin, mantido pela Adoptium, por ser open source, gratuita e compatível com GraalVM Native Image para compilação nativa futura. O download está disponível em https://adoptium.net. Após a instalação, a versão pode ser verificada pelo terminal:

Quadro 01 – Verificação da Versão do Java

```
java –version
```

Fonte: Elaborado pelo autor (2026).

A saída esperada deve indicar a versão 21 ou superior.

8.1.2 Maven

O Apache Maven é a ferramenta de build responsável por gerenciar as dependências declaradas no pom.xml, compilar o código-fonte e empacotar a aplicação em um arquivo JAR executável. A versão 3.8 ou superior é compatível com o projeto. O download está disponível em https://maven.apache.org/download.cgi. Após a instalação:

Quadro 02 – Verificação da Versão do Maven

```
mvn –version
```

Fonte: Elaborado pelo autor (2026).

8.1.3 PostgreSQL

O PostgreSQL é o sistema de banco de dados utilizado pelo IdeaHall. A versão 14 ou superior é compatível. O download está disponível em https://www.postgresql.org/download, com instaladores disponíveis para todos os sistemas operacionais. Durante a instalação, o PostgreSQL solicitará a definição de uma senha para o usuário postgres — essa senha será utilizada na configuração da conexão.

Após a instalação, o banco de dados ideahall deve ser criado manualmente. Isso pode ser feito pelo terminal utilizando o cliente psql:

Quadro 03 – Criação do Banco de Dados no PostgreSQL

```
sql
psql -U postgres
CREATE DATABASE ideahall;
\q
```

Fonte: Elaborado pelo autor (2026).

Ou por qualquer cliente gráfico PostgreSQL, como o pgAdmin, incluído na instalação padrão do PostgreSQL.

8.2 Configuração da Conexão

As credenciais de conexão com o banco de dados são configuradas no arquivo src/main/resources/application.properties. Os valores padrão assumem uma instalação local com o usuário postgres:

Quadro 04 – Configuração da Conexão com o PostgreSQL

```
properties
spring.datasource.url=jdbc:postgresql://localhost:5432/ideahall
spring.datasource.username=postgres
spring.datasource.password=postgres
```

Fonte: Elaborado pelo autor (2026).

Caso a senha definida durante a instalação do PostgreSQL seja diferente, o campo spring.datasource.password deve ser atualizado para refletir a senha correta antes da compilação.

8.3 Compilação

Com as dependências instaladas e o banco configurado, a compilação é realizada a partir da pasta raiz do projeto — o diretório que contém o arquivo pom.xml — pelo seguinte comando:

Quadro 05 – Compilação do Projeto com Maven

```
mvn clean package -DskipTests
```

Fonte: Elaborado pelo autor (2026).

O Maven baixa automaticamente todas as dependências declaradas no pom.xml na primeira execução, o que pode levar alguns minutos dependendo da velocidade da conexão. Nas execuções subsequentes, as dependências já estarão em cache local e o processo será consideravelmente mais rápido. O artefato final é gerado em target/ideahall-1.0.0-SNAPSHOT.jar.

Este capítulo apresenta o IdeaHall do ponto de vista do usuário final — como instalar, iniciar e interagir com o sistema na prática. Os exemplos a seguir simulam uma sessão real de uso, cobrindo desde o primeiro acesso até as operações mais avançadas disponíveis na plataforma.

8.3.1 Iniciando o sistema

Com o projeto compilado e o banco de dados configurado conforme descrito no capítulo anterior, o IdeaHall é iniciado pelo terminal com o seguinte comando:

Quadro 06 – Comando de Inicialização do IdeaHall

```
java -jar target/ideahall-1.0.0-SNAPSHOT.jar -dry_run
```

Fonte: Elaborado pelo autor (2026).

O sistema responde imediatamente com a tela de boas-vindas:

Quadro 07 – Saída do Sistema após Inicialização

```
IdeaHall — continuous mode. Type 'Q' to quit.
Use '-register' or '-login' to authenticate.
Use '-help' to list available commands.
```

Fonte: Elaborado pelo autor (2026).

A partir desse momento, o sistema está aguardando comandos. Todos os comandos são digitados diretamente no terminal, um por linha.

8.4 Criando sua conta

Na primeira vez que o sistema é utilizado, é necessário criar uma conta com o comando -register, informando email e nome de usuário:

Quadro 08 – Comando de Registro de Conta

```
-register alice@email.com alice
```

Fonte: Elaborado pelo autor (2026).

O sistema solicitará uma senha de forma interativa — o que for digitado não aparece na tela, garantindo que a senha não fique visível:

Quadro 09 – Interação do Sistema durante o Registro

```
Choose a password:
Confirm password:
Account created for 'alice'. You can now use -login.
```

Fonte: Elaborado pelo autor (2026).

A senha é armazenada como hash criptográfico no banco de dados. Nem a equipe de desenvolvimento tem acesso à senha original.

8.4.1 Entrando na sua conta

Com a conta criada, o login é feito com o comando -login:

Quadro 10 – Autenticação de Usuário no Sistema

```
-login alice@email.com
  Password:
  Logged in as 'alice'.
```

Fonte: Elaborado pelo autor (2026).

A partir deste momento, todos os comandos executados nesta sessão estão associados ao usuário alice. Não é necessário informar email ou senha novamente enquanto a sessão estiver ativa.

8.4.2 Publicando uma Ideia

Para publicar uma ideia na plataforma, utiliza-se o comando -create_idea com três informações: nome, categoria e descrição. Argumentos com mais de uma palavra devem ser envolvidos por aspas:

Quadro 11 – Criação de Ideia via Comando -create_idea

```
     -create_idea "App de Carona Solidária" transporte "Aplicativo para
conectar motoristas e passageiros em trajetos já planejados, sem cobrança
por corrida"
       Idea 'App de Carona Solidária' created by 'alice'.
```

Fonte: Elaborado pelo autor (2026).

A ideia é publicada imediatamente na plataforma com o contador de votos em zero e com alice registrada como autora — e portanto como LAMP desta ideia.

Qualquer usuário autenticado pode consultar os detalhes de uma ideia com -show_idea:

Quadro 12 – Exibição de Ideia via Comando -show_idea

```
-show_idea "App de Carona Solidária"
  +--------------------------------------------------+
  | IDEA                                             |
  +--------------------------------------------------+
  | Name        : App de Carona Solidária            |
  | Category    : transporte                         |
  | Author      : alice                              |
  | Votes       : 0                                  |
  +--------------------------------------------------+
  | Description :                                    |
  | Aplicativo para conectar motoristas e            |
  | passageiros em trajetos já planejados...         |
  +--------------------------------------------------+
```

Fonte: Elaborado pelo autor (2026).

8.4.3 Registrando interesse em uma ideia

Esse é o mecanismo central do IdeaHall. Qualquer usuário autenticado pode registrar interesse em uma ideia com -upp_idea. Suponha que um segundo usuário, bob, entre no sistema e encontre a ideia de Alice:

Quadro 13 – Registro de Voto via Comando -upp_idea

```
      -login bob@email.com
      -upp_idea "App de Carona Solidária"
        Vote registered for 'App de Carona Solidária'.
```

Fonte: Elaborado pelo autor (2026).

Se bob tentar votar novamente na mesma ideia, o sistema bloqueia:

Quadro 14 – Erro de Voto Duplicado no Sistema

```
      -upp_idea "App de Carona Solidária"
        Error: You have already voted for 'App de Carona Solidária'. Use -
down_idea to remove your vote.
```

Fonte: Elaborado pelo autor (2026).

Para remover o voto, basta usar -down_idea:

Quadro 15 – Remoção de Voto via Comando -down_idea

```
      -down_idea "App de Carona Solidária"
        Vote removed from 'App de Carona Solidária'.
```

Fonte: Elaborado pelo autor (2026).

8.4.4 Modificando sua ideia — privilégios LAMP

Como autora da ideia, alice tem acesso exclusivo aos comandos de modificação. Nenhum outro usuário consegue alterar ou excluir ideias que não criou.

Para atualizar a categoria:

Quadro 16 – Alteração de Categoria via Comando -change_category

```
      -login alice@email.com
      -change_category "App de Carona Solidária" mobilidade-urbana
        Category of 'App de Carona Solidária' updated to 'mobilidade-
urbana'.
```

Fonte: Elaborado pelo autor (2026).

Para renomear:

Quadro 17 – Renomeação de Ideia via Comando -change_name

```
      -change_name "App de Carona Solidária" "CaronaApp"
        Name updated: 'App de Carona Solidária' -> 'CaronaApp'.
```

Fonte: Elaborado pelo autor (2026).

Se bob tentar modificar a ideia de alice, o sistema recusa:

Quadro 18 – Erro de Permissão para Modificação de Ideia

```
      -login bob@email.com
      -change_name "CaronaApp" "HackNome"
        Error: Only the author of 'CaronaApp' can modify or delete it.
```

Fonte: Elaborado pelo autor (2026).

8.4.5 Consultando campos individuais

Para obter um campo específico de uma ideia sem exibir todos os dados, os comandos take_* retornam apenas o valor solicitado — úteis para scripts e automações:

Quadro 19 – Consulta de Campos Individuais via Comandos take_*

```
-take_name "CaronaApp"
  CaronaApp

-take_category "CaronaApp"
  mobilidade-urbana

-take_description "CaronaApp"
  Aplicativo para conectar motoristas e passageiros...
```

Fonte: Elaborado pelo autor (2026).

8.5 Encerrando a Sessão

Para sair do sistema sem fechar o terminal, usa-se -logout:

Quadro 20 – Encerramento de Sessão via Comando -logout

```
-logout
  'alice' logged out.
```

Fonte: Elaborado pelo autor (2026).

O sistema permanece em execução, pronto para um novo login. Para encerrar completamente, basta digitar Q:

Quadro 21 – Encerramento Completo do Sistema

```
Q
 Session closed for 'alice'. Goodbye.
```

Fonte: Elaborado pelo autor (2026).

8.6 Comandos Disponíveis

Os comandos do IdeaHall são implementados com Picocli em estrutura declarativa de sub-comandos, organizados em quatro categorias funcionais conforme o nível de acesso exigido (PICOCLI, 2026).

8.6.1 Comandos de autenticação — disponíveis a qualquer usuário, sem sessão prévia:

- -register <email> <username> — cadastra uma nova conta no sistema. A senha é solicitada interativamente após a execução do comando, com confirmação, e armazenada como hash BCrypt no banco de dados. Email e username devem ser únicos.
- -login <email> — autentica o usuário na sessão atual. A senha é solicitada interativamente e verificada contra o hash armazenado. Em caso de sucesso, o SessionContext passa a identificar o usuário para todos os comandos subsequentes da sessão.
- -logout — encerra a sessão do usuário atual, limpando o SessionContext. O sistema permanece em execução no modo contínuo, permitindo novo login.

8.6.2 Comandos de leitura — disponíveis a qualquer usuário autenticado:

- -show_idea <idea_name> — exibe todos os campos de uma ideia: nome, categoria, autor, contador de votos e descrição.
- -take_name <idea_name> — retorna o nome da ideia especificada.
- -take_category <idea_name> — retorna a categoria da ideia especificada.
- -take_description <idea_name> — retorna a descrição da ideia especificada.

8.6.3 Comandos de escrita — disponíveis a qualquer usuário autenticado:

- -create_idea <name> <category> <description> — publica uma nova ideia na plataforma. O autor é registrado automaticamente como o usuário da sessão

atual, sem necessidade de informar explicitamente. O contador de votos é inicializado em zero.

- -upp_idea <idea_name> — registra o interesse do usuário autenticado em uma ideia. Cada usuário pode votar em uma ideia no máximo uma vez — tentativas de voto duplicado são bloqueadas tanto na camada de serviço quanto por constraint de chave primária composta no banco. O contador de votos da ideia é incrementado atomicamente.
- -down_idea <idea_name> — remove o voto previamente registrado pelo usuário em uma ideia. Retorna erro se o usuário não tiver votado na ideia. O contador de votos é decrementado atomicamente.

8.6.4 Comandos LAMP — disponíveis exclusivamente ao autor da ideia

O conceito de LAMP no IdeaHall não representa um papel global no sistema, mas uma relação de propriedade entre usuário e ideia. O autor de uma ideia — registrado no campo author_email no momento da criação — é o único com permissão para executar os comandos abaixo sobre ela. A verificação ocorre em tempo de execução por comparação entre o email do usuário autenticado na sessão e o author_email armazenado no banco.

- -change_name <idea_name> <new_name> — renomeia a ideia especificada.
- -change_category <idea_name> <new_category> — altera a categoria da ideia especificada.
- -change_description <idea_name> <new_description> — altera a descrição da ideia especificada.
- -delete_idea <idea_name> — remove permanentemente a ideia e todos os votos associados a ela do banco de dados. A remoção dos votos é executada previamente à exclusão da ideia para respeitar as constraints de chave estrangeira.

8.7 Referência Rápida de Comandos

Tabela 04 – Referência de Todos os Comandos Disponíveis no Sistema

Comando	Argumentos	Quem pode usar?
-register	<email> <username>	Qualquer pessoa
-login	<email>	Qualquer pessoa
-logout		Usuário autenticado
-create_idea	<name> <category> <description>	Usuário autenticado
-show_idea	<idea_name>	Usuário autenticado
-take_name	<idea_name>	Usuário autenticado
-take_category	<idea_name>	Usuário autenticado
-take_description	<idea_name>	Usuário autenticado
-upp_idea	<idea_name>	Usuário autenticado
-down_idea	<idea_name>	Usuário autenticado
-change_name	<idea_name> <new_name>	Autor da ideia (LAMP)
-change_category	<idea_name> <new_category>	Autor da ideia (LAMP)
-change_description	<idea_name> <new_description>	Autor da ideia (LAMP)
-delete_idea	<idea_name>	Autor da ideia (LAMP)
-dry_run		Qualquer pessoa
-help	<command>	Qualquer pessoa

Fonte: Elaborado pelo autor (2026).

8.8 Site de Documentação

Todas as informações apresentadas neste capítulo — instalação, dependências, modos de execução, comandos disponíveis e especificações técnicas da ferramenta — estão igualmente disponíveis no site de documentação oficial do IdeaHall, desenvolvido e entregue como parte integrante deste trabalho. O site reúne

o conteúdo técnico da plataforma em formato navegável e acessível, servindo como referência centralizada para usuários e desenvolvedores que venham a utilizar ou contribuir com o projeto. Recomenda-se sua consulta como complemento a este documento, especialmente para informações que possam ser atualizadas ao longo do ciclo de desenvolvimento futuro da ferramenta.

9 INOVAÇÕES TECNOLÓGICAS

O desenvolvimento do IdeaHall não ocorre em um vácuo histórico. Ele emerge de um conjunto de transformações no ecossistema tecnológico que, ao longo das últimas décadas, redefiniram o que é possível construir com equipes pequenas, recursos limitados e conhecimento técnico aplicado com direção.

Como discutido ao longo deste trabalho, o avanço da economia digital reduziu progressivamente as barreiras de entrada para novos produtos de software. Infraestrutura de nuvem acessível, frameworks maduros de código aberto e plataformas de distribuição global tornaram viável o que antes exigia estruturas corporativas de grande porte. Esse cenário é o mesmo que tornou possível a concepção do IdeaHall por uma equipe de três pessoas em formação técnica — e é também o cenário que o IdeaHall pretende tornar mais navegável para as equipes que vierem depois.

O que há de novo, no entanto, não está nas tecnologias que o compõem. Java, Spring Boot, PostgreSQL e GraalVM são ferramentas consolidadas, amplamente documentadas e utilizadas pela indústria. A inovação do IdeaHall não é tecnológica no sentido da pilha técnica — é conceitual no sentido do problema que escolheu resolver.

Peter Thiel, em Zero to One, argumenta que a inovação genuína não consiste em fazer melhor o que já existe, mas em criar algo que ainda não existe (THIEL; MASTERS, 2014). Plataformas que organizam código existem. Plataformas que organizam dúvidas técnicas existem. Plataformas que anunciam produtos em lançamento existem. O que não existia — e o que o IdeaHall propõe — é uma infraestrutura dedicada especificamente à etapa anterior a todas essas: a organização colaborativa e estruturada de problemas tecnológicos não resolvidos, validados coletivamente por quem os experimenta, e acessíveis a quem tem capacidade técnica de resolvê-los.

Essa distinção é precisa e deliberada. O IdeaHall não é um ambiente de desenvolvimento. Não é uma plataforma de lançamento. Não é um fórum de discussão técnica. É um mecanismo para a etapa que Eric Ries, em The Lean Startup, trata como anterior ao próprio ciclo de construção, avaliação e aprendizado — a etapa em que a equipe precisa ter identificado uma hipótese de mercado válida antes de

escrever a primeira linha de código (RIES, 2012). O IdeaHall estrutura exatamente essa etapa, que a literatura de produto reconhece como crítica, mas que nenhuma ferramenta existente havia se proposto a resolver de forma dedicada e acessível a times técnicos pequenos.

A segunda inovação conceitual está no modelo de propriedade intelectual adotado. Diferentemente de plataformas onde o registro de uma ideia implica alguma forma de precedência ou exclusividade, o IdeaHall opera sobre o princípio oposto: ideias publicadas são contribuições ao bem comum da comunidade técnica, sem direito de exclusividade para quem as registrou primeiro. Essa decisão não é ingenuidade jurídica — é uma escolha de design fundamentada na compreensão de que sistemas de precedência autoral em plataformas colaborativas geram desincentivo ao registro e à implementação, destruindo exatamente o valor que a plataforma pretende criar. A inovação aqui está em tratar o problema da propriedade intelectual não como uma questão a ser resolvida por restrição, mas como uma questão a ser contornada por design.

A terceira inovação está na arquitetura de expansão. O IdeaHall foi concebido desde o início não como um CLI isolado, mas como um núcleo de API sobre o qual múltiplas interfaces serão construídas — web, mobile, desktop — cada uma servindo um perfil de usuário distinto com o mesmo repositório central de dados. Isso significa que a ferramenta entregue neste trabalho não é o produto: é a fundação sobre a qual o produto será construído. Essa distinção entre núcleo e interface, entre backend permanente e clientes intercambiáveis, é uma decisão arquitetural que reflete uma compreensão madura do ciclo de vida de plataformas tecnológicas — e que posiciona o IdeaHall para crescer sem necessidade de reestruturação fundamental a cada nova extensão.

O modelo de sustentabilidade previsto — com patrocínio de ideias e participação da plataforma nos fundos gerados — introduz ainda uma dinâmica econômica inédita nesse tipo de ferramenta: a possibilidade de que problemas validados coletivamente sejam financiados para se tornarem soluções reais, com a plataforma atuando como intermediária confiável entre quem financia e quem desenvolve. Esse mecanismo, quando implementado, fechará um ciclo que começa na identificação do problema e termina na entrega da solução — com o IdeaHall presente em todas as etapas.

O que este trabalho entrega, portanto, não é apenas um sistema funcional. É a prova de conceito de uma abordagem diferente para um problema que afeta toda equipe técnica pequena que já se perguntou, diante de um terminal aberto e habilidades prontas para usar: o que vale a pena construir? O IdeaHall é a resposta estruturada a essa pergunta — e sua existência, ainda que em versão inicial, demonstra que a pergunta tinha solução.

10 RESPONSIVIDADE

A responsividade compreende a capacidade de um sistema ou interface de adaptar sua apresentação e comportamento ao contexto de uso do usuário — dispositivo, tamanho de tela, resolução e capacidades de entrada disponíveis — sem perda de funcionalidade. No IdeaHall, essa preocupação se manifesta de forma distinta em cada componente da plataforma.

10.1 Responsividade no Sistema CLI

A saída do IdeaHall foi estruturada para ser legível dentro de uma largura de 80 colunas — padrão histórico de terminais, garantido em qualquer ambiente de execução, de emuladores de desktop a sessões SSH remotas.

O sistema detecta a presença ou ausência de um terminal interativo via System.console() e adapta seu comportamento conforme o ambiente: em terminais reais, a leitura de senhas é feita via Console.readPassword(), que suprime o eco dos caracteres digitados; em ambientes sem terminal interativo — como IDEs ou pipelines de CI/CD — o sistema realiza fallback para o Scanner compartilhado, mantendo o funcionamento correto sem exigir configuração adicional.

A distinção entre modo single-shot e modo contínuo -dry_run também constitui uma forma de responsividade comportamental: o primeiro oferece ciclo de vida curto e exit codes semânticos para contextos de automação; o segundo preserva o estado de autenticação entre comandos para uso humano interativo.

10.2 Responsividade no Site de Documentação

O site de documentação foi desenvolvido seguindo o modelo Mobile First, com layout progressivamente expandido via media queries CSS em três faixas: abaixo de 768px, o menu lateral é substituído por menu colapsável e o conteúdo ocupa largura total; entre 768px e 1024px, o menu lateral é exibido de forma compacta em layout de duas colunas; acima de 1024px, o layout completo é apresentado com largura máxima de conteúdo definida para preservar legibilidade em monitores de alta resolução.

Imagens e diagramas utilizam max-width: 100% e formato SVG sempre que possível, garantindo escala sem perda de qualidade em qualquer densidade de pixels. Tipografia e espaçamentos são definidos em unidades relativas — rem e em — respeitando as preferências de tamanho de fonte do sistema operacional do usuário. Elementos de navegação em dispositivos de toque possuem área mínima de 44×44px conforme o critério WCAG 2.5.5.

10.3 Considerações Gerais

A responsividade não é uma camada aplicada ao final do desenvolvimento — é uma decisão de arquitetura. No IdeaHall, essa perspectiva se reflete tanto na adaptação do CLI ao ambiente de execução quanto na estrutura progressiva do site, planejada para suportar a expansão futura da plataforma sem exigir reestruturação da base já construída.

11 ACESSIBILIDADE DIGITAL

A acessibilidade digital abrange as práticas e decisões de desenvolvimento que tornam sistemas utilizáveis pelo maior número possível de pessoas. No IdeaHall, essa preocupação se aplica a duas frentes: o sistema CLI e o site de documentação, cada um com diretrizes adequadas à sua natureza de interface.

11.1 Acessibilidade no CLI

Por não possuir elementos visuais, interfaces de terminal eliminam certas barreiras comuns em aplicações web, mas introduzem outras, relacionadas à clareza da linguagem e ao comportamento diante de erros. O IdeaHall adota as seguintes práticas:

- Linguagem clara: mensagens de erro e confirmação escritas em linguagem natural, substituindo códigos genéricos por textos descritivos e orientações de ação.
- Saída estruturada: campos nomeados e alinhados de forma consistente, facilitando a leitura por ferramentas assistivas de terminal.
- Feedback imediato: toda operação produz uma resposta explícita de sucesso ou falha, seguindo as heurísticas de usabilidade de Nielsen (NIELSEN, 1994).
- Exit codes semânticos: valores diferenciados (0, 1, 2, 3) para cada tipo de resultado, essenciais para automação via scripts e pipelines.
- Compatibilidade com leitores de tela: ausência de sequências ANSI de cor ou animação, mantendo a saída em texto puro.

11.2 Acessibilidade no Site de Documentação

O site foi desenvolvido em conformidade com as diretrizes WCAG 2.1 nível AA — padrão exigido pela Lei Brasileira de Inclusão nº 13.146/2015 (BRASIL, 2015) — seguindo os quatro princípios POUR:

- Perceptível: contraste mínimo de 4,5:1, sem uso de cor como único diferenciador, e texto alternativo em todas as imagens.
- Operável: navegação completa por teclado, sem armadilhas de foco, com skip links para acesso direto ao conteúdo principal.
- Compreensível: idioma declarado no atributo lang, linguagem clara e, em formulários futuros, mensagens de erro descritivas.
- Robusto: HTML semanticamente correto, uso criterioso de ARIA e validação com WAVE, Lighthouse, NVDA e VoiceOver.

11.3 Considerações Gerais

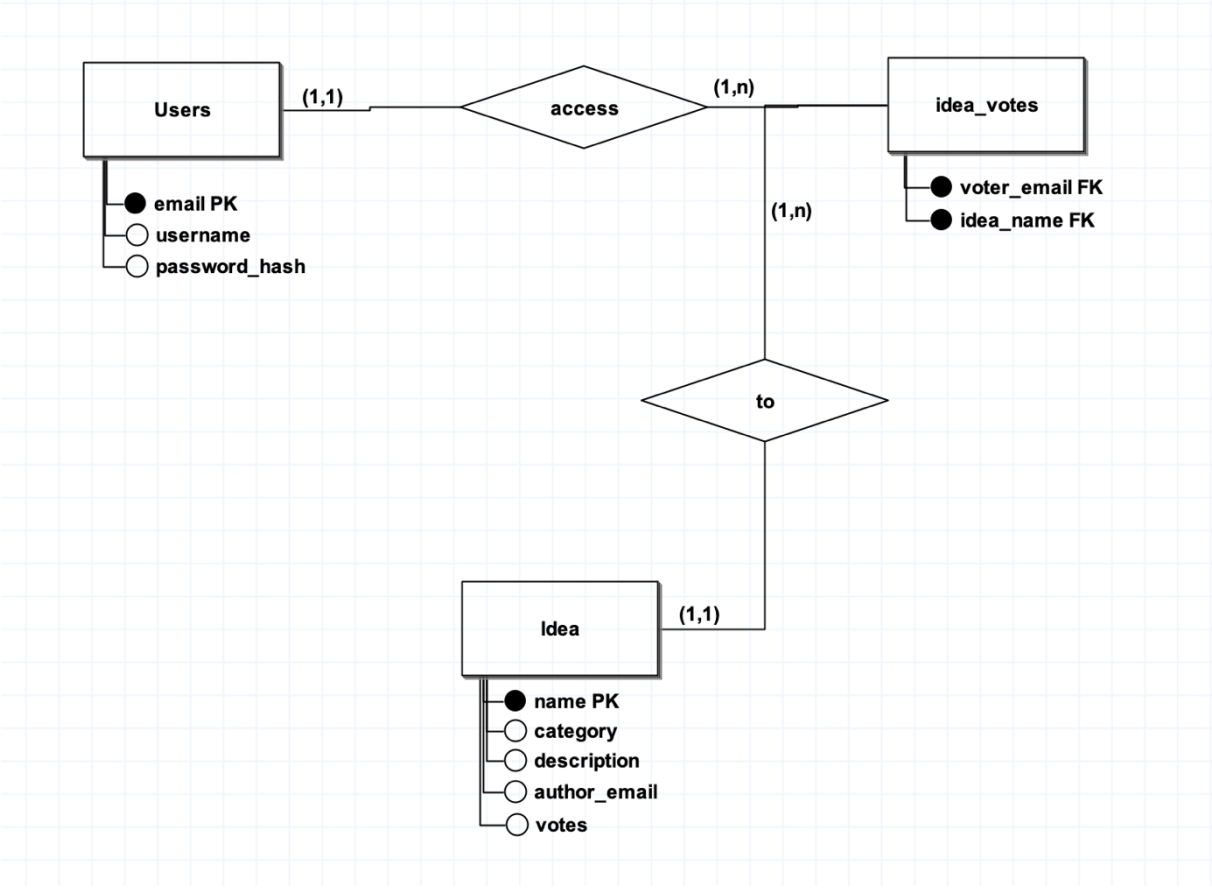
Acessibilidade foi tratada como dimensão de qualidade desde o início do projeto, e não como adição posterior. A estrutura adotada garante continuidade quando o site for expandido para a versão web, ampliando o alcance da plataforma e reforçando seu compromisso com responsabilidade técnica e social.

12 ANÁLISE DE BANCO DE DADOS

O modelo de dados do sistema CLI é organizado em três entidades centrais que se relacionam para suportar todas as operações previstas: a entidade de usuário, que armazena credenciais e metadados de identificação; a entidade de ideia, que armazena o conteúdo estruturado de cada ideia registrada; e a entidade de interação, que registra as validações — UPs — que usuários conferem a ideias de outros. A comunicação entre essas entidades segue um modelo relacional convencional, com chaves estrangeiras garantindo integridade referencial e índices nas colunas de consulta frequente garantindo desempenho adequado.

Para melhor visualização desse banco de dados, ele pode ser também visualizado no padrão brModelo, da seguinte forma:

Figura 01 – Diagrama do banco de dados do IdeaHall



Fonte: Elaborado pelo autor (2026).

13 RESULTADOS FINAIS

O principal resultado deste trabalho é o IdeaHall — uma ferramenta de linha de comando funcional, compilável e executável, que implementa de forma operacional a proposta descrita nos capítulos iniciais. O sistema entregue é composto por dois artefatos complementares: o CLI e o site de documentação.

O CLI foi desenvolvido em Java 21 com Spring Boot 3.2.5, Picocli 4.7.5, Spring Data JPA, BCrypt para criptografia de senhas e PostgreSQL como sistema de persistência (SPRING, 2026; PICOCLI, 2026; POSTGRESQL, 2026). O projeto é compilado via Maven e empacotado como um arquivo JAR executável sem dependências externas além da JVM e do banco de dados. O conjunto de funcionalidades entregues inclui:

- Sistema de autenticação com cadastro de usuários, login por sessão com senha criptografada via BCrypt e logout explícito. Nenhuma senha é armazenada ou transmitida em texto puro em nenhuma etapa do ciclo de vida da aplicação.
- Publicação e consulta de ideias com campos de nome, categoria, descrição e autoria registrada automaticamente pelo usuário da sessão. A consulta por campos individuais via comandos take_* permite integração com scripts e automações externas.
- Sistema de votos com controle de duplicidade garantido em dois níveis independentes: verificação na camada de serviço antes de qualquer operação de escrita, e constraint de chave primária composta na tabela idea_votes no banco de dados, que rejeita inserções duplicadas independentemente da camada de aplicação.
- Controle de acesso por autoria — o conceito de LAMP — implementado como uma relação entre usuário e ideia: somente o autor de uma ideia pode modificá-la ou excluí-la, verificado em tempo de execução por comparação de emails entre a sessão ativa e o campo author_email persistido no banco.
- Dois modos de execução: single-shot para uso programático e modo contínuo - dry_run para uso interativo, com estado de autenticação preservado entre comandos consecutivos dentro da mesma sessão.

- O site de documentação foi desenvolvido como artefato HTML único, sem dependências de frameworks externos, com design responsivo para dispositivos móveis, navegação lateral com highlight automático da seção ativa, blocos de código copiáveis e conformidade com os critérios WCAG 2.1 AA de acessibilidade.

14 CONSIDERAÇÕES FINAIS

O desenvolvimento do IdeaHall representa uma resposta concreta a uma lacuna observada no ecossistema de criação de software: a ausência de um mecanismo estruturado que permita a desenvolvedores validar a viabilidade e a demanda de seus projetos antes mesmo de seu início. A partir do sistema proposto, o grupo descrito nos capítulos iniciais deste trabalho passa a dispor de uma ferramenta capaz de subsidiar a tomada de decisão com base em evidências coletadas diretamente de potenciais usuários — algo que, até então, inexistia de forma integrada no mercado.

Um aspecto de particular relevância está na formação de um ciclo de valor mútuo entre desenvolvedores e usuários finais. Ao escolher uma ideia com base nas interações registradas na plataforma, o desenvolvedor já parte de um conjunto pré-existente de interessados naquela solução, reduzindo a margem de incerteza e o risco de abandono por falta de propósito — problema recorrente no desenvolvimento de software. Esse modelo simbiótico representa uma ruptura com o paradigma tradicional de desenvolvimento, no qual a validação de mercado ocorre, geralmente, apenas após investimento substancial de tempo e recursos, frequentemente resultando em produtos que não encontram adoção significativa.

14.1 Impacto Potencial no Ecossistema Tecnológico

A longo prazo, caso o IdeaHall alcance adoção em escala, seus efeitos sobre o ecossistema de desenvolvimento de software podem ser consideráveis. A plataforma atua, essencialmente, como um mecanismo de alocação de esforço: ao sinalizar quais problemas possuem demanda real e quantificável, ela tende a direcionar o trabalho de desenvolvedores para soluções com maior probabilidade de impacto efetivo, em detrimento de projetos construídos sobre premissas não validadas.

No contexto da crescente integração de inteligência artificial nos fluxos de desenvolvimento — com ferramentas de geração de código, prototipagem automatizada e assistência técnica tornando-se cada vez mais acessíveis —, a capacidade de identificar o que construir torna-se tão crítica quanto a capacidade de

como construir. O IdeaHall endereça precisamente essa dimensão: a inteligência coletiva da plataforma, expressa nas interações e sinalizações de interesse dos usuários, funciona como um sistema de priorização distribuída, potencialmente capaz de encurtar o ciclo entre a identificação de uma necessidade social ou técnica e a entrega de uma solução funcional.

Em um horizonte mais amplo, essa dinâmica pode contribuir para a democratização do desenvolvimento de software: ao reduzir o risco percebido associado a novos projetos, a plataforma remove uma barreira significativa para desenvolvedores independentes e equipes de pequeno porte que, sem acesso a estruturas formais de pesquisa de mercado, frequentemente optam por não investir em ideias inovadoras por incerteza quanto à sua recepção.

14.2 Desafios Previstos para a Implementação

A trajetória entre o estado atual do projeto e sua plena operacionalização no mercado envolve um conjunto de desafios técnicos, estratégicos e operacionais que merecem análise objetiva.

Desafio técnico — dependência de CLI: Na versão atual, a interação com o sistema requer o uso de uma interface de linha de comando, o que restringe o perfil de usuário a desenvolvedores com familiaridade técnica suficiente para operar nesse ambiente. Essa limitação é incompatível com o objetivo de alcançar uma base de usuários diversificada, que inclua também profissionais de outras áreas com ideias a validar. A remoção dessa barreira de entrada constitui, portanto, uma prioridade técnica de primeira ordem no roadmap de desenvolvimento.

Desafio de escalabilidade: À medida que o volume de ideias e interações cresce, surgem questões não triviais de arquitetura: como garantir a qualidade e a relevância das sugestões exibidas a cada usuário? Como evitar que ideias com maior visibilidade inicial monopolizem a atenção em detrimento de propostas igualmente relevantes, mas de menor alcance? Esses problemas de ranqueamento e curadoria algorítmica demandarão soluções robustas conforme a plataforma escala.

Desafio de moderação e integridade dos dados: Uma plataforma baseada em sinalizações de interesse é, por natureza, suscetível a manipulações — sejam elas intencionais, como a inflação artificial de interesse em determinada ideia, ou não

intencionais, como vieses de amostragem decorrentes de uma base de usuários inicialmente homogênea. A implementação de mecanismos de verificação e moderação será essencial para preservar a confiabilidade dos dados produzidos pela plataforma.

Desafio de monetização: A definição de um modelo de negócio sustentável que não comprometa a proposta de valor central da plataforma — a colaboração aberta entre desenvolvedores e usuários — é uma questão estratégica relevante. Modelos baseados em acesso premium, parcerias institucionais ou serviços analíticos são caminhos possíveis, cada um com implicações distintas sobre a dinâmica da comunidade.

O problema do cold start: Talvez o desafio mais imediato seja o período inicial em que a escassez de usuários ativos compromete a experiência oferecida pelo sistema, cuja dinâmica é intrinsecamente dependente do volume de interações. Esse fenômeno é análogo ao observado historicamente em plataformas como YouTube e Reddit, onde a ausência de conteúdo inviabilizava a atração de novos usuários, e vice-versa — um ciclo de retroalimentação negativa (RIES, 2012). A equipe avalia que a natureza do IdeaHall é particularmente favorável à superação desse obstáculo em horizonte relativamente curto, dado que o público-alvo inicial — desenvolvedores — apresenta motivação intrínseca para engajar com uma plataforma que beneficia diretamente seu processo criativo. Esse segmento tende a atuar como vetor primário de dispersão orgânica da plataforma no mercado.

14.3 Da Versão Acadêmica à Versão Comercial

É importante situar adequadamente o escopo do presente trabalho: o IdeaHall, em sua forma atual, foi concebido e desenvolvido no contexto de um curso técnico, com os recursos, o tempo e as restrições inerentes a esse ambiente. O produto entregue neste TCC representa, portanto, uma prova de conceito funcional — suficiente para demonstrar a viabilidade técnica da proposta e validar suas premissas fundamentais, mas não a versão final que o time idealiza para o mercado.

A equipe de desenvolvimento tem como objetivo lançar futuramente uma versão comercial do IdeaHall, substancialmente expandida em relação ao protótipo acadêmico. Essa versão contempla, entre outras iniciativas:

- Interface gráfica completa: substituição da interface CLI por uma aplicação com experiência de usuário elaborada, acessível a perfis não técnicos, com fluxos intuitivos de cadastro de ideas, exploração de projetos e sinalização de interesse.
- Porte para as principais plataformas: disponibilização do sistema como aplicação web, com versões nativas para iOS e Android, garantindo acesso ubíquo independentemente do dispositivo ou sistema operacional do usuário.
- Integração com ecossistemas de desenvolvimento: conexões com plataformas como GitHub, GitLab e gerenciadores de projetos amplamente adotados, de modo a reduzir a fricção entre a validação de uma ideia no IdeaHall e o início efeito de seu desenvolvimento.
- Camada analítica: painéis de dados que ofereçam ao desenvolvedor uma visão quantitativa do interesse gerado por sua ideia ao longo do tempo, subsidiando decisões de priorização com base em métricas objetivas.

Essa transição do contexto acadêmico para o comercial não é trivial. Ela implica não apenas a ampliação do escopo técnico, mas também a estruturação de processos de suporte, a definição de políticas de privacidade e uso de dados, e a construção de uma estratégia de go-to-market consistente. O time reconhece esses desafios e os compreende como etapas naturais da maturação de um produto de software.

14.4 Perspectivas Futuras

Para que o cenário descrito acima se concretize, é imprescindível que o projeto avance conforme o roadmap estabelecido e que a adoção da plataforma alcance o volume crítico necessário para ativar sua dinâmica colaborativa. A equipe demonstra comprometimento com essa trajetória e confiança nos fundamentos que tornam o IdeaHall uma contribuição relevante tanto para a engenharia de software quanto para o ecossistema tecnológico de forma mais ampla.

O IdeaHall, em sua essência, propõe uma reconfiguração da relação entre quem desenvolve e quem usa software — tornando-a mais simétrica, mais informada e, consequentemente, mais eficiente. Se essa proposta encontrar a adoção esperada, o projeto deixa de ser apenas uma ferramenta e passa a ser uma infraestrutura de

colaboração para a inovação tecnológica: um resultado que, do ponto de vista dos seus criadores, justifica plenamente o investimento realizado e motiva a continuidade do trabalho que se inicia aqui.

REFERÊNCIAS

APACHE. Maven Documentation. Disponível em: https://maven.apache.org. Acesso em: 14 mar. 2026.

BRASIL. Lei nº 13.146, de 6 de julho de 2015. Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência. Brasília: Presidência da República, 2015. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm. Acesso em: 20 fev. 2026.

BROWN, Tim. Design thinking: uma metodologia poderosa para decretar o fim das velhas ideias. Rio de Janeiro: Elsevier, 2010.

CHRISTENSEN, Clayton M. et al. Competing Against Luck: the story of innovation and customer choice. New York: HarperCollins, 2016.

CLOUDFLARE. Cloudflare Pages Pricing. Disponível em: https://www.cloudflare.com/plans/developer-platform/. Acesso em: 09 abr. 2026.

EM NOTÍCIA. Transformação digital e tecnologias de ponta representam alta em investimentos. Disponível em: https://emnoticia.com.br/270228-transformacao-digital-e-tecnologias-de-ponta-representam-alta-em-investimentos/. Acesso em: 10 mar. 2026.

GITHUB. GitHub Pages. Disponível em: https://pages.github.com. Acesso em: 02 abr. 2026.

GRAALVM. GraalVM Native Image Documentation. Disponível em: https://www.graalvm.org/latest/reference-manual/native-image/. Acesso em: 16 mar. 2026.

GRAALVM NATIVE BUILD TOOLS. Native Build Tools Documentation. Disponível em: https://graalvm.github.io/native-build-tools. Acesso em: 16 mar. 2026.

INDÚSTRIA 4.0. Mercado indústria global 4.0 chegará a US$ 219,8 bilhões. Disponível em: https://www.industria40.ind.br/noticias/21478-mercado-industria-global-40-chegara-us-2198-. Acesso em: 10 mar. 2026.

INSTITUTO FEDERAL DE SANTA CATARINA. Cursos superiores de tecnologia dobram número de matrículas em 10 anos. Publicado em: 16 jan. 2019. Disponível em: https://www.ifsc.edu.br/en/web/noticias/w/cursos-superiores-de-tecnologia-dobram-numero-de-matriculas-em-10-anos. Acesso em: 16 mar. 2026.

NETLIFY. Pricing. Disponível em: https://www.netlify.com/pricing/. Acesso em: 15 abr. 2026.

NIELSEN, Jakob. Usability Engineering. San Francisco: Morgan Kaufmann, 1994.

ORGANIZAÇÃO DAS NAÇÕES UNIDAS. World Economic Situation and Prospects 2021. Nova York: United Nations, 2021. Disponível em: https://www.un.org/development/desa/dpad/publication/world-economic-situation-and-prospects-2021/. Acesso em: 23 mar. 2026.

PICOCLI. Picocli — a mighty tiny command line interface. Disponível em: https://picocli.info. Acesso em: 19 fev. 2026.

POSTGRESQL. PostgreSQL Documentation. Disponível em: https://www.postgresql.org/docs. Acesso em: 16 mar. 2026.

RAILWAY. Pricing. Disponível em: https://railway.com/pricing. Acesso em: 12 mar. 2026.

RENDER. Deploy for Free. Disponível em: https://render.com/docs/free. Acesso em: 18 mar. 2026.

RIES, Eric. A startup enxuta. São Paulo: Leya, 2012.

SPRING. Spring Boot Reference Documentation. Disponível em: https://docs.spring.io/spring-boot/docs/current/reference/html/. Acesso em: 16 mar. 2026.

SUPABASE. Pricing & Fees. Disponível em: https://supabase.com/pricing. Acesso em: 05 mar. 2026.

THIEL, Peter; MASTERS, Blake. Zero to One: notes on startups, or how to build the future. New York: Crown Business, 2014.

UOL NOTÍCIAS. IA poderia impactar 40% dos empregos em todo o mundo, segundo a ONU. Disponível em: https://noticias.uol.com.br/ultimas-noticias/afp/2025/04/03/ia-poderia-impactar-40-dos-empregos-em-todo-o-mundo-segundo-a-onu.htm. Acesso em: 10 mar. 2026.

VERCEL. Pricing. Disponível em: https://vercel.com/pricing. Acesso em: 22 abr. 2026.