PatternLab – Um Laboratório Virtual para a Prática de Padrões de Projeto

Trabalho de Conclusão de Curso

Rafael Campos da Paixão

Prof. Dr. Lauro Cássio Martins de Paula
Orientador

Instituto Federal da Bahia - IFBA
Curso de Análise e Desenvolvimento de Sistemas
Campus Salvador

Salvador, Bahia, Brasil
05 de fevereiro de 2026

Sumário

1 Introdução

1.1 Desenvolvimento de Software

A área de desenvolvimento de software é marcada por rápidas e constantes mudanças. A evolução tecnológica impactou, e continua impactando, esse campo de diversas formas. Inicialmente, o software era desenvolvido principalmente para fins militares; posteriormente, passou a ser utilizado em computadores domésticos. Com o tempo, esses computadores começaram a se conectar pela internet, evoluíram para smartphones, que hoje possuem mais poder computacional do que os primeiros computadores, e seguem em contínua transformação. O aumento no número de usuários resultou em uma demanda crescente por software voltado tanto a indivíduos quanto a organizações, e o mercado de desenvolvimento de software é o responsável por suprir essa necessidade.

Uma das estratégias criadas no cenário de desenvolvimento de software para suprir essa necessidade de forma rápida são as metodologias ágeis. Abordagens como Scrum[1] e Extreme Programming (XP)[2] são amplamente utilizadas no mercado para sustentar ciclos de entrega contínua de funcionalidades em diversos sistemas. Ambas fazem parte do Manifesto Ágil[3], documento que define os princípios que norteiam essas metodologias. Um desses princípios estabelece que “contínua atenção à excelência técnica e ao bom design aumenta a agilidade” . Dessa forma, evidencia-se a preocupação dos autores em destacar a importância de manter um código de qualidade e com bom design, de modo a facilitar o ciclo de desenvolvimento.

O cenário atual de desenvolvimento de software é caracterizado por sistemas de alta complexidade, ciclos acelerados de desenvolvimento, e pela constante manutenção e evolução de código. Nesse contexto, a qualidade dos códigos escritos representa uma vantagem competitiva no mercado, pois, como citado anteriormente com base no manifesto ágil, códigos bem escritos e de fácil manutenibilidade aceleram o tempo de desenvolvimento à entrega de novas funcionalidades e a sustentabilidade da base de código a longo prazo. Portanto, é de fundamental importância que o ensino das melhores convenções de escrita de código faça parte da formação dos desenvolvedores de software.

1.2 Padrões de projeto

Dentro das boas práticas de escrita de código, encontram-se os Padrões de Projeto apresentados pela *Gang of Four* no clássico livro *”Design Patterns: Elements of Reusable Object-Oriented Software”* [4]. Eles são um conjunto de soluções para problemas recorrentes na engenharia de software, independentemente do domínio, tecnologia ou plataforma. O conhecimento desses padrões por parte dos desenvolvedores torna a resolução de problemas corriqueiros mais eficiente e sofisticada, além de criar um vocabulário comum que facilita a manutenção de códigos, a colaboração e a comunicação entre programadores.

Os padrões de projeto catalogados no livro podem ser classificados em três categorias. Os padrões criacionais, que incluem Abstract Factory, Builder, Factory Method, Prototype e Singleton, concentram-se no quando e como os objetos são criados no sistema. Esses

padrões surgem da necessidade de tornar um sistema independente de como seus objetos são criados, compostos e representados, promovendo maior flexibilidade ao substituir instanciações diretas por mecanismos que encapsulam a lógica de criação e desacoplam o código que usa os objetos do código que os produz.

Os padrões estruturais, como Adapter, Bridge, Composite, Decorator, Facade, Flyweight e Proxy, abordam a forma como classes e objetos são compostos para formar estruturas maiores. Sua principal preocupação é garantir que, ao combinar componentes independentes, o resultado seja uma estrutura coesa e de fácil manutenção. Muitos desses padrões se apoiam em herança e composição para criar abstrações que simplificam o relacionamento entre partes do sistema, permitindo que componentes com interfaces incompatíveis colaborem ou que subsistemas complexos sejam acessados de forma transparente.

Por fim, os padrões comportamentais, que englobam Chain of Responsibility, Command, Iterator, Observer, Strategy, Template Method, entre outros, tratam dos algoritmos e da atribuição de responsabilidades entre objetos. Diferentemente das outras categorias, o foco aqui não está apenas na estrutura, mas no fluxo de controle e na comunicação entre os componentes em tempo de execução. Esses padrões buscam descrever como objetos que, individualmente, seriam incapazes de realizar uma tarefa de forma satisfatória, passam a colaborar de maneira organizada, reduzindo o acoplamento e tornando os comportamentos do sistema mais fáceis de estender e modificar.

Nem todos os padrões de projeto fazem parte do cotidiano dos programadores. No entanto, o conhecimento sobre eles pode facilitar a resolução de problemas recorrentes e contribuir para a elaboração de um código com melhor qualidade de design. Além disso, muitas tecnologias modernas utilizam esses padrões de forma implícita em sua implementação. Um exemplo é o padrão Composite, amplamente empregado em bibliotecas para construção de interfaces de usuário, permitindo a composição hierárquica de componentes e o gerenciamento de suas ações de maneira unificada.

1.3 Inteligência Artificial

Inteligência Artificial (IA) é um campo de estudo que remonta à década de 1950, tendo como principal objetivo desenvolver máquinas capazes de imitar atividades cognitivas humanas [5]. Desde então, diversas revoluções tecnológicas impulsionaram esse campo, como o aumento do poder computacional e a enorme quantidade de dados gerada pela internet. Esses avanços favoreceram o desenvolvimento de diferentes subáreas da IA, possibilitando a criação de sistemas especialistas — como sistemas de suporte ao diagnóstico médico —, de aprendizado de máquina — como sistemas de detecção de fraudes e recomendação de produtos —, de redes neurais — como assistentes virtuais — e, mais recentemente, das IAs generativas.

Nesse contexto, o projeto atual se apoia justamente nesse avanço mais recente: o crescente desenvolvimento e a popularização das IAs generativas. Essa tecnologia vem sendo amplamente utilizada em diversos contextos e apresenta um ritmo constante de inovações e tendências. Seja por meio de Retrieval-Augmented Generation (RAG), integração entre mo-

delos tradicionais de busca de dados e IAs generativas[6], Model Context Protocols (MCP), padrão aberto de conexão de IAs com fontes externas[7], ou da integração com provedores de modelos, as IAs generativas oferecem uma ampla gama de casos de uso.

O motivo da escolha da IA generativa é a sua grande habilidade em duas áreas-chave para este trabalho: a geração de textos, usada para criar problemas parecidos com os do mundo real, e a análise de código-fonte. Juntas, essas duas capacidades permitem construir a ferramenta proposta.

1.4 Declaração do Problema

Apesar dos benefícios, o aprendizado e o estudo dos padrões de projeto impõem desafios específicos. O aprendizado da maior parte das tecnologias e conceitos da área de computação exige bastante prática pelo estudante, se tratando de padrões de projeto, a literatura e materiais de ensino muitas vezes se utilizam de cenários muito simples que não traduzem bem a realidade enfrentada pelos desenvolvedores, onde muitas vezes existem cenários ambíguos e sem um caminho fácil de escolha de padrões. É importante que o aprendizado de padrões de projeto simule problemas mais relacionados ao mundo real, facilitando a transição do conhecimento teórico para a prática.

Adicionalmente, ao contrário de outros tópicos como a lógica de programação, onde a prática geralmente leva o estudante a cenários de certo ou errado, os padrões de projeto são implementações que mesmo incorretas podem levar programas funcionais, dessa forma criando a falsa sensação de aprendizado. Isso pode ser prejudicial para um futuro desenvolvedor, visto que a escrita de códigos com um mau design pode levar a muitos problemas no ciclo de desenvolvimento de um software. Esse desafio pode ser resolvido com um ciclo de feedback mais eficiente, por meio de uma análise automatizada das implementações feitas.

Diante desses obstáculos foi observada uma oportunidade de criação de uma ferramenta que ofereça aos estudantes um ambiente que resolva esses dois desafios. Esse ambiente deve contar com a geração de problemas alinhados ao mundo real e aos padrões de projeto, e com a integração para a análise em tempo real de códigos implementados por estudantes. Esse ambiente virtual se comportará como um laboratório, onde será possível realizar a prática do aprendizado teórico de padrões de projeto em um ambiente controlado que simula problemas reais e com o ciclo de feedback mais rápido9.

1.5 Resumo do Trabalho

Diante do cenário apresentado, este trabalho propõe o desenvolvimento do PatternLab, um laboratório virtual para a prática dos 23 padrões de projeto catalogados pela Gang of Four. A ferramenta foi construída para endereçar diretamente os dois desafios identificados: a escassez de cenários de prática alinhados ao mundo real e a ausência de um ciclo de feedback eficiente sobre as implementações realizadas pelos estudantes. Para resolver esses desafios, o PatternLab integra as capacidades de geração de texto e análise de código das IAs Generativas em suas duas funcionalidades centrais. A primeira é a geração dinâmica de desafios: ao solicitar um problema, o sistema seleciona aleatoriamente um tema e um

padrão de projeto e solicita à API da OpenAI a criação de um cenário contextualizado, sem mencionar diretamente o padrão esperado, exigindo do estudante a identificação e a aplicação do conhecimento. A segunda é a análise automatizada de submissões: após o estudante implementar sua solução em um editor de código integrado, o sistema a envia para avaliação pela IA, que retorna uma pontuação de 0 a 100, pontos positivos, sugestões de melhoria e um feedback geral sobre a implementação. A solução foi construída como uma aplicação web completa, com um backend desenvolvido em Kotlin e Spring Boot, um banco de dados MySQL para persistência das informações e um frontend em TypeScript e ReactJS. A arquitetura adota separação em camadas, isolando as regras de negócio das integrações externas, o que garante facilidade de manutenção e a possibilidade de substituição do provedor de IA no futuro. A qualidade do código foi mantida ao longo do desenvolvimento por meio de testes automatizados com JUnit e análise estática contínua com SonarQube, com critério mínimo de 80% de cobertura de testes. O sistema foi implantado em ambiente de produção utilizando a plataforma Railway e está disponível publicamente via navegador web. Os capítulos seguintes detalham cada aspecto do projeto: as tecnologias utilizadas, a proposta e os diagramas de design, os requisitos funcionais e não-funcionais, as estratégias de qualidade adotadas e o manual de uso da aplicação.

2 Objetivos

Diante do problema evidenciado no ensino de padrões de projeto e da importância que esse componente tem na formação de desenvolvedores qualificados, o presente trabalho busca os seguintes objetivos.

2.1 Objetivo Geral

Desenvolver o PatternLab, uma aplicação web voltada para estudantes e desenvolvedores em formação que desejam aprimorar o conhecimento nos padrões de projeto da Gang of Four. A plataforma se diferencia das abordagens tradicionais de ensino ao utilizar Inteligência Artificial Generativa para produzir cenários de prática contextualizados ao mundo real e para oferecer análise automatizada das implementações submetidas, fornecendo feedback preciso sobre a qualidade do código produzido.

2.2 Objetivos Específicos

1. **Projetar a arquitetura do sistema**, especificando os módulos necessários e suas responsabilidades, com foco na modularidade e mantenabilidade do sistema.

2. **Implementar a gestão de usuários**, permitindo que usuários se registrem no sistema e tenham acesso as funcionalidades do sistema de forma segura.

3. **Desenvolver o módulo de geração e analise**, utilizando uma integração com alguma Inteligência Artificial Generativa para a criação de cenários de problemas realistas e para a análise das submissões de código dos estudantes.

4. **Desenvolver o ambiente virtual**, estabelecendo um ambiente onde os estudantes possam interagir com os problemas gerados, submeter suas soluções e receber o feedback analítico da IA. Garantindo uma interface e experiência positiva e responsiva para os usuários.

5. **Implementar o espaço de acompanhamento e revisão**, possibilitando ao usuário o acesso a um área em que seja possível revisar problemas solucionados, e uma área onde seja possível acompanhar o progresso no uso da plataforma.

6. **Validar a usabilidade e eficácia da aplicação**, garantindo por meio de testes com usuários e com a coleta de feedbacks de uso que a plataforma está de fato facilitando o processo de aprendizado de padrões de projeto.

3 Tecnologias

Este capítulo detalha as ferramentas e tecnologias utilizadas para a construção do projeto. A escolha das tecnologias foi feita com base na maturidade das tecnologias, no suporte da comunidade de desenvolvedores, no alinhamento com os requisitos do projeto e na familiaridade do autor. Desta forma, o desenvolvimento da aplicação teve um ganho de eficiência no tempo de desenvolvimento. As ferramentas listadas aqui são divididas em duas partes: o backend, que cuida da persistência de dados e das regras de negócio, e o frontend, que é a parte de apresentação e experiência do usuário.

3.1 Kotlin

Kotlin[8] é uma linguagem de programação desenvolvida pela JetBrains. Trata-se de uma linguagem compilada que roda na Java Virtual Machine (JVM)[8], o que lhe garante acesso à biblioteca padrão de classes do Java. Por isso, sua sintaxe é semelhante à do Java, porém mais concisa e direta. A linguagem oferece funcionalidades que permitem adotar tanto o paradigma funcional quanto o orientado a objetos, além de contar com tipagem estática, verificação de valores nulos e diversos padrões de projeto já implementados nativamente[9][10]. Com Kotlin, é possível desenvolver aplicações mobile multiplataforma, bem como sistemas de backend. Neste projeto, o Kotlin foi utilizado para implementar o backend da aplicação, uma escolha justificada pelo conhecimento prévio do autor na linguagem.

3.2 Spring Framework

O Spring Framework[11] é um conjunto de módulos que facilita a criação de aplicações em Java. Com o Spring, desenvolvedores podem construir sistemas seguindo diferentes tipos de arquitetura e utilizar linguagens que rodam na JVM, como Kotlin e Groovy. Ao utilizar esses módulos, é possível focar nos casos de uso da aplicação sem se preocupar com funcionalidades específicas da arquitetura adotada.[11] Os módulos do Spring permitem a criação de microsserviços, aplicações web, reativas, em nuvem, serverless e orientadas a eventos. Além disso, os desenvolvedores podem combinar esses módulos de forma flexível no sistema em desenvolvimento, permitindo, por exemplo, a criação de uma aplicação com características de microsserviços e arquitetura orientada a eventos. No projeto, o Spring é utilizado para criar o backend de uma aplicação web, utilizamos o modúlo de comunicação com banco de dados[12], o Spring JPA, e o módulo de autenticação e autorização[13], o módulo Spring Security.

3.3 MySql

O MySQL[14] é um Sistema de Gerenciamento de Banco de Dados (SGBD) de código aberto, desenvolvido pela Oracle, que utiliza SQL, linguagem específica para interagir com SGBDs[14], para a criação e manipulação de dados. O SGBD possui casos de uso em diversos tipos de aplicações, como redes sociais, e-commerces e soluções do tipo Software as a Service (SaaS). Trata-se de uma alternativa acessível, leve, performática e com um conjunto de funcionalidades completo.[15] Além disso, é amplamente utilizado no setor de desenvolvi-

mento de software, o que facilita a resolução de problemas. Seu uso no projeto se justifica pelas características mencionadas e pelo conhecimento prévio do autor.

3.4 TypeScript

Com a popularização e o crescimento na criação de aplicações web, o JavaScript, linguagem utilizada para implementar lógicas em páginas HTML dentro dos navegadores, passou a ser cada vez mais utilizado. Seu uso se expandiu de tal forma que a linguagem deixou de estar restrita ao frontend, sendo hoje amplamente empregada também no backend de diversas aplicações. Embora seja uma linguagem completa e com diversas funcionalidades, sua característica de não possuir checagem estática de tipos de dados nem sempre atende às exigências do contexto moderno da programação.[16] Diante disso, o TypeScript foi criado com o objetivo de permitir aos desenvolvedores utilizar a sintaxe do JavaScript aliada à checagem de tipos estática. Seu uso no projeto se justifica pela ampla adoção da linguagem pela comunidade, em substituição ao JavaScript.

3.5 ReactJS

ReactJS é uma biblioteca de componentes utilizada para a criação de interfaces de usuário com a linguagem JavaScript. Criada pelo Facebook em 2011, tornou-se desde então uma das principais ferramentas para o desenvolvimento de interfaces em aplicações web. Com o React, é possível dividir a interface em componentes reutilizáveis que controlam seus próprios estados e podem interagir entre si, formando interfaces complexas[17]. O React foi utilizado neste projeto devido ao conhecimento prévio do autor na ferramenta e à ampla comunidade de desenvolvedores, o que facilita tanto a resolução de problemas quanto o desenvolvimento de novas funcionalidades.

3.6 Conclusão

Em resumo, o conjunto de ferramentas utilizado no projeto forma a base de uma aplicação web moderna. A utilização do Kotlin e Spring Framework possibilitou a criação de um backend que atende aos requisitos do projeto com ferramentas seguras e de bom desempenho. O MySQL ofereceu uma solução de persistência de dados confiável. Já o TypeScript e o React permitiram a criação de uma interface de usuário interativa e performática, baseada em componentes. A união entre a camada de persistência e regras de negócio com a camada de apresentação resultou em uma aplicação completa, que atende a todos os requisitos definidos.

4 Proposta

Conforme citado no capítulo introdutório, os Padrões de Projeto introduzidos pela Gang of Four [4] formam um catálogo de soluções reutilizáveis que contribui para a criação de bases de código mais sofisticadas, de fácil manutenção e evolução. No entanto, o ensino desses padrões nem sempre se traduz bem para situações reais do dia a dia de desenvolvedores, e também carece de um ciclo de feedback rápido durante a prática. Este capítulo apresenta como as Inteligências Artificiais Generativas têm potencial para resolver essas lacunas e de que forma elas estão sendo integradas ao projeto.

4.1 Relação entre IA Generativa e Padrões de Projeto

A solução proposta se fundamenta em duas capacidades centrais das IAs Generativas: a geração de texto e a análise de código-fonte. A relação entre a IA e os Padrões de Projeto é aplicada de forma prática para resolver as lacunas de aprendizado identificadas. O sistema utiliza a IA para criar cenários de problemas dinâmicos para avaliar as soluções implementadas pelos estudantes.

- **Geração de problemas contextualizados:** A principal função da IA nesta etapa é gerar desafios de forma dinâmica. O processo funciona da seguinte forma:

  1. **Seleção Aleatória:** Primeiramente, o sistema escolhe aleatoriamente um tema (como ”Sistema Bancário”ou ”E-commerce”) e um Padrão de Projeto de uma lista pré-definida.

  2. **Construção do Prompt:** Em seguida, essas informações são inseridas em um modelo de prompt estruturado, que contém as instruções para a IA.

  3. **Integração e Geração:** Por fim, o sistema envia este prompt a um provedor de IA Generativa via API. A IA, então, gera um desafio único que se encaixa no tema e cuja solução ideal é a aplicação do padrão de projeto selecionado.

  O modelo de prompt utilizado, apresentado a seguir, foi elaborado seguindo boas práticas de engenharia de prompts [18], como a definição de um papel para a IA (personificação) e a limitação do escopo da resposta, para garantir a qualidade dos desafios gerados.

  > You are a university-level professor specialized in software design patterns. Your task is to create challenging exercises that help students identify which design pattern applies to a given scenario.
  > – Generate a real-world problem where the `${pattern.name}` pattern would be the most appropriate solution.
  > – The scenario must align conceptually with the characteristics of `${pattern.name}`: `${pattern.description}`.
  > – Ensure the scenario is strongly tied to the given theme: `${theme}`.
  >
  > **Instructions:**
  > – Write the entire description in **Brazilian Portuguese**, using formal and technical language.

  - Begin by establishing the context of the system and its core functionalities.
  - Then, describe a specific technical challenge faced by the engineering or project team.
  - Do **not** mention or hint directly at the name of the design pattern.
  - Provide **no more than two subtle, non-obvious clues** that suggest the nature of the design problem.
  - Avoid formatting elements such as bold, italics, or highlighting.
  - Limit the entire description to a maximum of **four paragraphs**.
  - Ensure the final output is structured using the following JSON format:

```
{
"description": "[Descreva aqui o cenário completo do problema, em português]",
"title": "[Um resumo breve do desafio, também em português]"
}
```

  Important:
  - The problem must clearly reflect a scenario where the `${pattern.name}` would be an effective solution, based on its typical use cases and characteristics.
  - The goal is to help students infer the correct design pattern based on the context, not to recognize it by keywords or overt hints.

- **Análise de código implementado**: De forma semelhante à geração de desafios, o sistema utiliza uma IA Generativa para analisar o código implementado pelo estudante. A IA avalia a implementação do padrão de projeto e retorna uma análise estruturada contendo: uma pontuação de 0 a 100, uma lista de pontos positivos, sugestões de melhoria e um feedback geral. Tecnicamente, o processo ocorre por meio de uma integração via API com um provedor de IA, para o qual o seguinte prompt é enviado:

  > You are an expert software engineer and instructor specialized in software design patterns.
  > Your task is to evaluate a student's proposed solution to a design pattern problem.
  > **Inputs:**
  > - Problem Title: `${challenge.title}`
  > - Problem Description: `${challenge.description}`
  > - Expected Pattern: `${pattern.name}`
  > - Pattern Description: `${pattern.description}`
  > - Language (can be pseudocode or real code): `${submission.language}`
  > - Code Submission: `${submission.code}`
  >
  > **Evaluation Criteria:**
  > 1. Correct and appropriate implementation of the `${pattern.name}` design pattern.
  > 2. Code structure, clarity, and adherence to best practices (even if it's pseudocode).
  > 3. Whether the solution effectively addresses the problem scenario.

Scoring Guide: Use the following scoring guide when assigning a score (0 to 100):

- 0–24: Poor — Fundamental errors or incorrect solution.
- 25–49: Fair — Attempt made but with major issues.
- 50–74: Good — Mostly correct but with some flaws.
- 75–100: Excellent — Correct, clean, and effective.

Instructions:

- Your evaluation should be critical but constructive, as if guiding a student.
- Focus on technical aspects, explaining any conceptual mistakes or misapplications of the pattern.
- If the code is pseudocode, evaluate based on logic, structure, and correctness of the pattern usage.
- Do not include markdown or formatting indicators in the output.

Output format (plain JSON only, no code fences):

```
{
"score": [A number between 0 and 100],
"feedback": [A list of general observations about the solution],
"strengths": [A list of specific technical strengths],
"improvements": [A list of clear suggestions or areas for improvement]
}
```

Essas duas integrações com um provedor de IA generativa nos permite dar a capacidade de geração de problemas e analise de respostas ao sistema, com essas integrações feitas, o backend oferece uma API com duas funcionalidades que utilizam essas integrações e fazem a persistência das informações vindas da IA.

A consistência das respostas da IA no idioma português brasileiro e outras diretrizes são garantidas pela combinação de dois fatores. O primeiro é a estrutura do prompt utilizado, que segue boas práticas de engenharia de prompts ao definir explicitamente as instruções de idioma, formato de saída e restrições de conteúdo por meio de diretrizes claras de o que fazer e o que evitar, reduzindo significativamente a margem para respostas fora do esperado.

O segundo fator é o modelo de integração adotado: cada requisição ao provedor de IA é enviada de forma isolada, sem histórico de mensagens anteriores, o que significa que cada geração de desafio ou análise de submissão ocorre em uma conversa completamente nova. Esse comportamento elimina o risco de deriva de contexto, fenômeno comum em interações multi-turno onde instruções iniciais podem ser gradualmente ignoradas pelo modelo ao longo de uma conversa prolongada. Dessa forma, o prompt completo é sempre enviado integralmente a cada requisição, garantindo que as instruções de idioma e formato sejam respeitadas de maneira consistente em todas as interações com o sistema.

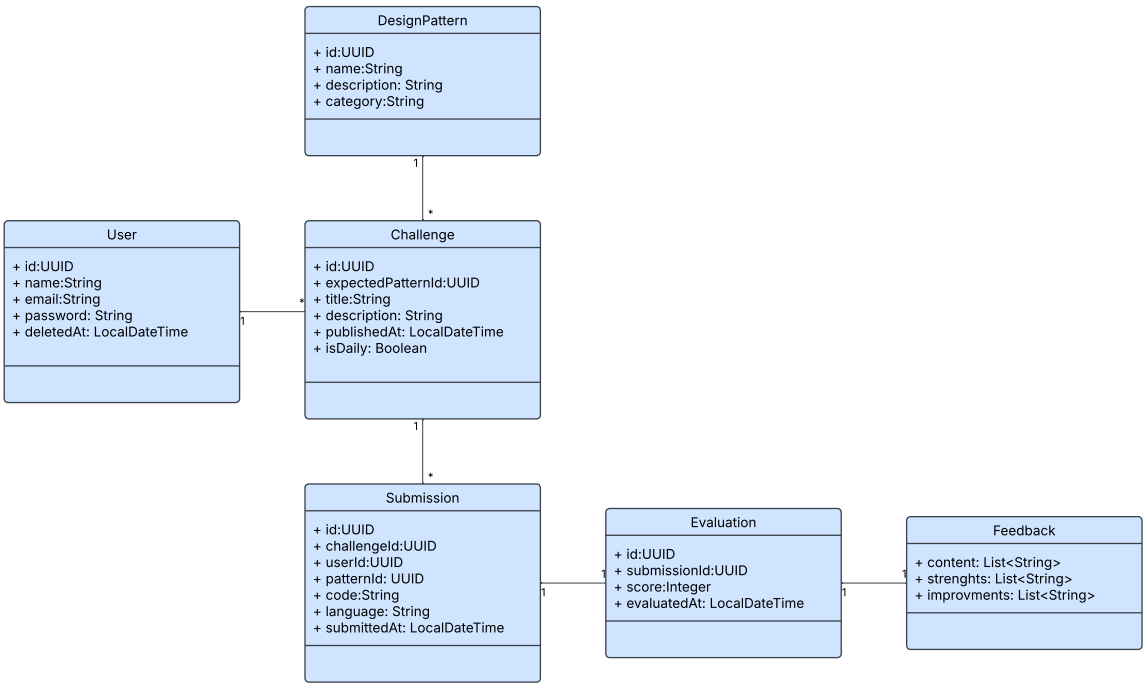
Com as informações armazenadas, podemos levar ao usuário o desafio e o feedback de resposta em um formato ideal no frontend. A seguir, no capítulo de design, é demonstrado como essas funcionalidades são integradas e mapeadas na aplicação por meio de diagramas.

4.2 Design

Nesse capítulo são exibidos diversos designs da aplicação, como o modelo das principais entidades, fluxos críticos como a geração de desafios e análise de submissões, os casos de uso, os componentes da aplicação, modelo de banco de dados e uma visão arquitetural do sistema.

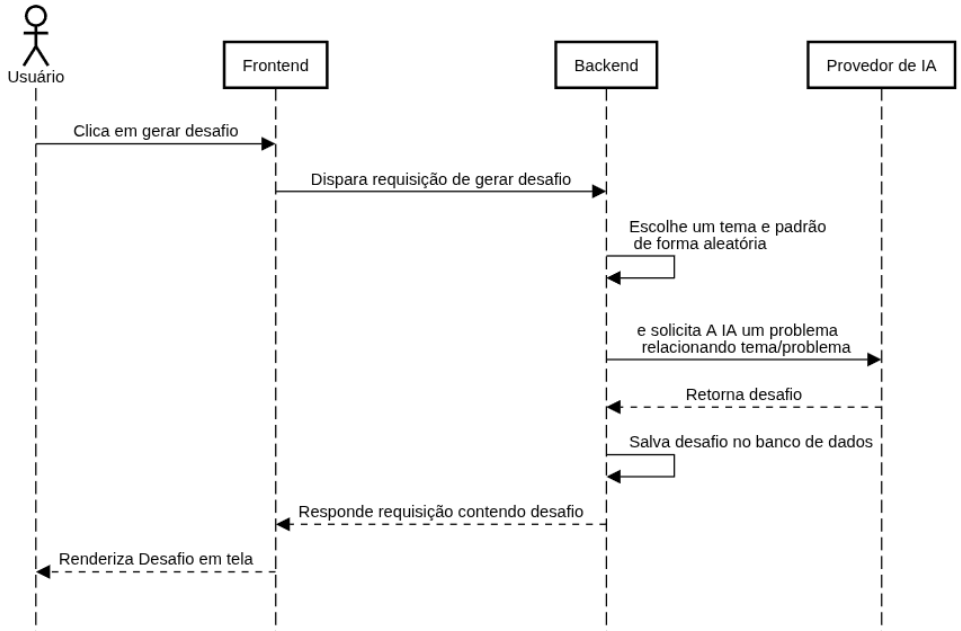
4.2.1 Projeto UML

1. **Diagrama de Classes (Entidades do Domínio)**: O diagrama abaixo ilustra as entidades de domínio da aplicação. O Usuário representa um estudante que utiliza o sistema, ele pode ter vários desafios associados a seu perfil, cada desafio tem um padrão de projeto associado e pode possuir uma submissão, essas submissões possuem uma avaliação realizada pela IA, e, a partir da análise, são identificados pontos positivos, pontos de melhoria e um feedback geral do código submetido. Todas essas informações foram mapeadas para as classes abaixo.



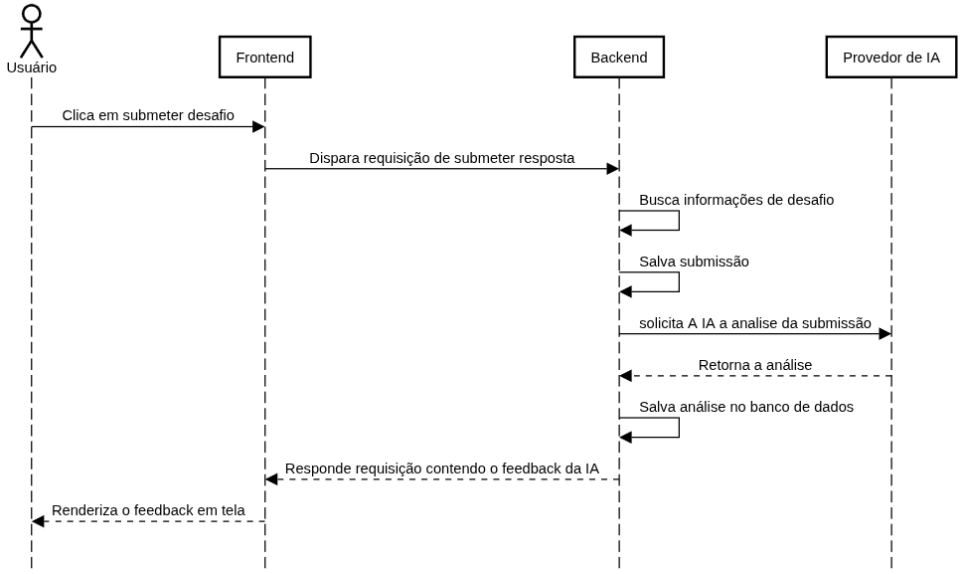
2. **Diagrama de Sequência (Geração de desafios)**: O diagrama sequencial abaixo ilustra o processo de geração de desafios, a partir do momento em que ele solicita a geração de um desafio.

Diagrama de Sequência - Geração de desafios



3. **Diagrama de Sequência (Análise de Submissões)**: O diagrama sequencial abaixo ilustra o processo de análise de submissões do usuário, a partir do momento em que ele realiza a submissão da resposta do desafio.

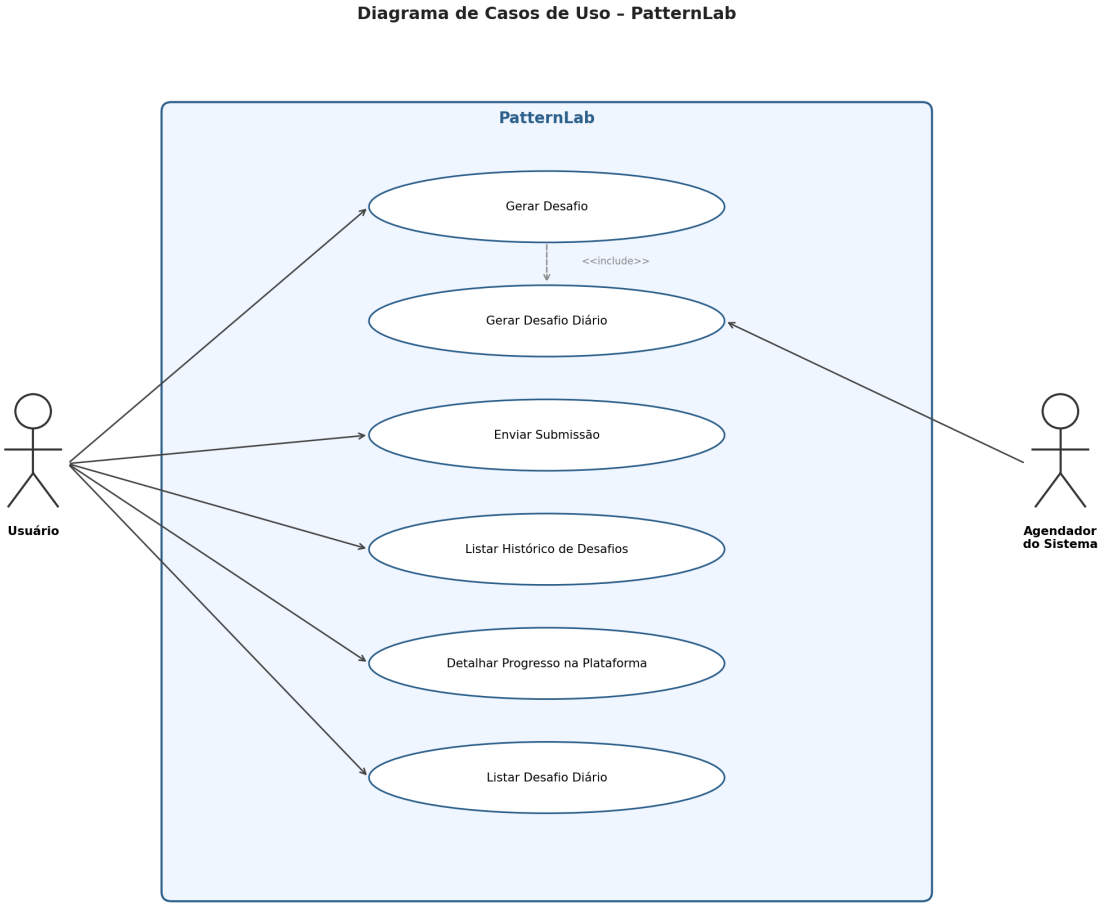
Diagrama de Sequência - Análise de submissões



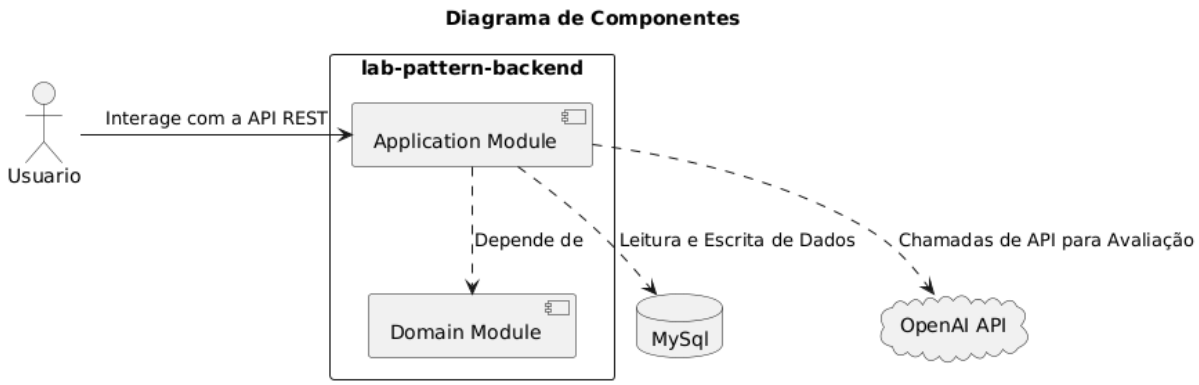
Vale ressaltar que os diagramas de sequência apresentados utilizam rótulos representativos das camadas da aplicação em vez de instâncias explícitas das classes do domínio. Essa escolha foi feita por questões de legibilidade, uma vez que a inclusão de todas as instâncias envolvidas nos fluxos tornaria os diagramas visualmente densos, comprometendo sua clareza. As classes e responsabilidades de cada camada estão detalhadas no diagrama de classes apresentado anteriormente.

4. **Diagrama de Casos de Uso**: O diagrama de casos de uso da figura abaixo ilustra os atores presentes no sistema, os casos de uso e suas relações. O usuário tem acesso

as funcionalidades de gerar desafios, enviar submissões, listar o histórico de desafios realizados, detalhar informações de progresso e listar o desafio diário. Já o agendador do sistema é responsável por gerar diariamente um desafio compartilhado entre todos os usuários. Por fim, como descrito na proposta, o provedor de IA é responsável por gerar desafios de usuários e desafios diários, assim como receber e analisar submissões.

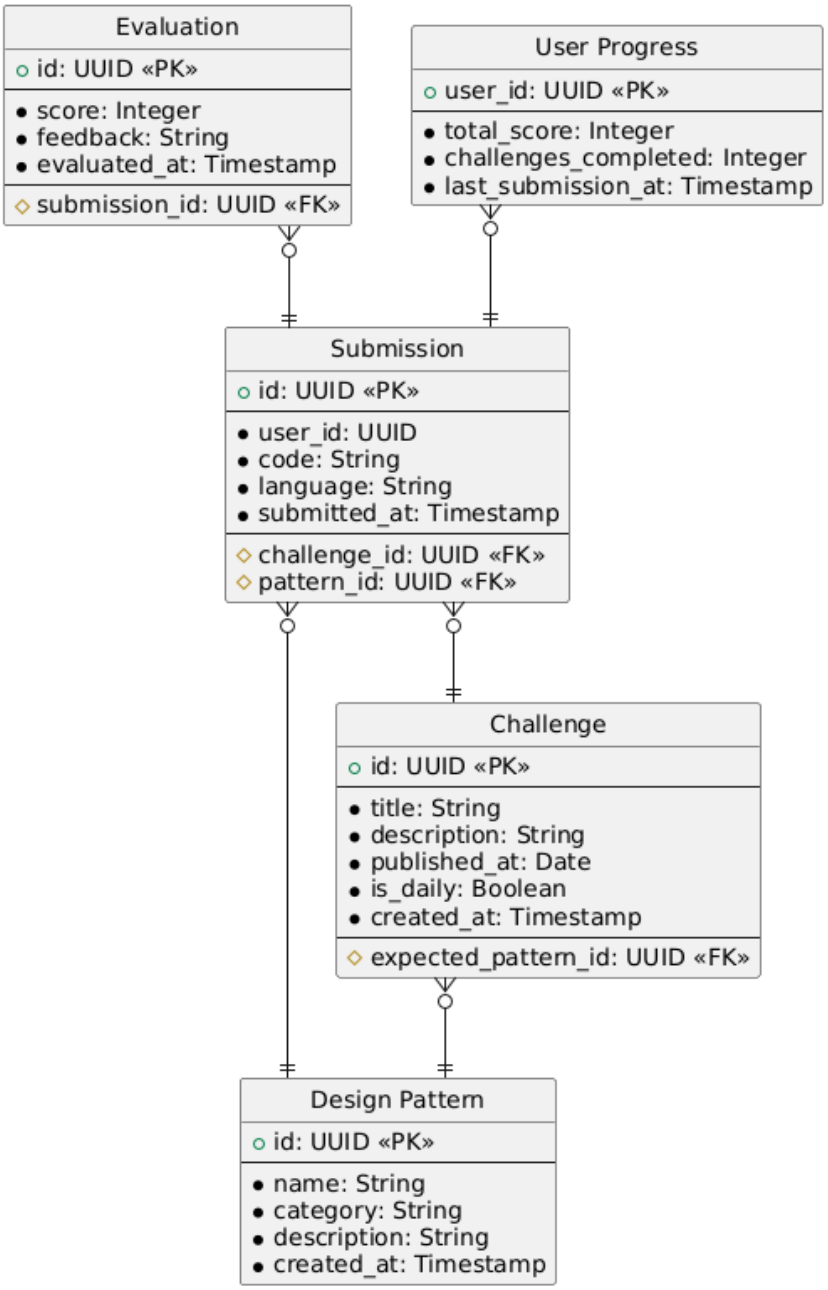


5. **Diagrama de Componentes**: O diagrama de componentes abaixo ilustra os componentes principais da aplicação backend, podemos considerar neste cenário o usuário como o frontend da aplicação, ele interage com a API Rest. O backend da aplicação é composto por dois módulos, o módulo de *application* contém todas as integrações externas, como interações com banco de dados, serviços externos e pontos de entrada do sistema, já o módulo *domain* contém todos os casos de uso e entidades existentes do sistema, por não possuir nenhum código baseado em dependências externas e trabalhar apenas com interfaces que são implementadas pelos componentes do módulo application, os elementos do módulo de domínio podem trabalhar com qualquer implementação de dependência externas futuras, assim facilitando, por exemplo, a mudança do provedor de IA ou banco de dados em um momento futuro.



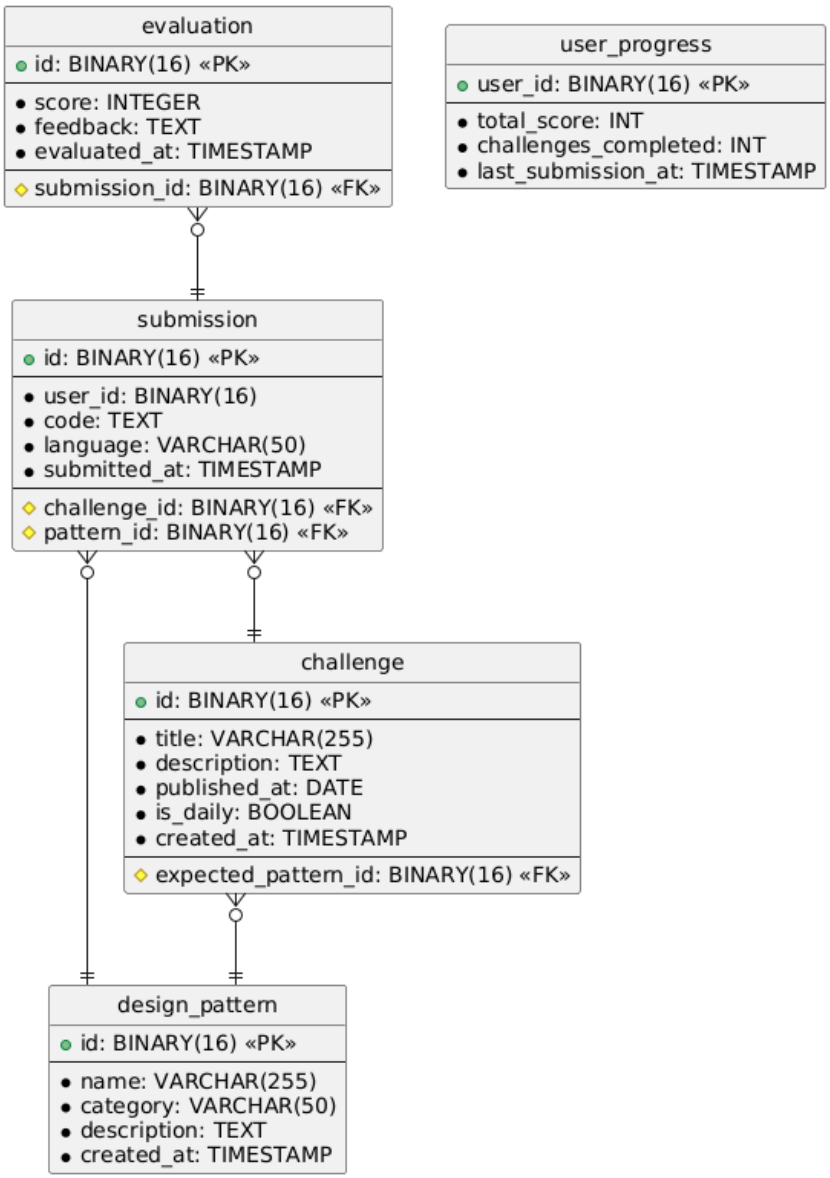
4.2.2 Modelo de Banco de Dados Lógico

O modelo lógico foi criado para representar as principais informações e regras de negócio do sistema, de acordo com os requisitos levantados. Nele, foram definidas as entidades essenciais da aplicação, como Desafio, Submissão e Avaliação, e seus respectivos atributos e relacionamentos. Para desenvolver este modelo, foi utilizada a notação de Entidade-Relacionamento (ER), que permite visualizar como os dados se conectam de forma abstrata, antes de se decidir sobre a tecnologia específica do banco de dados.



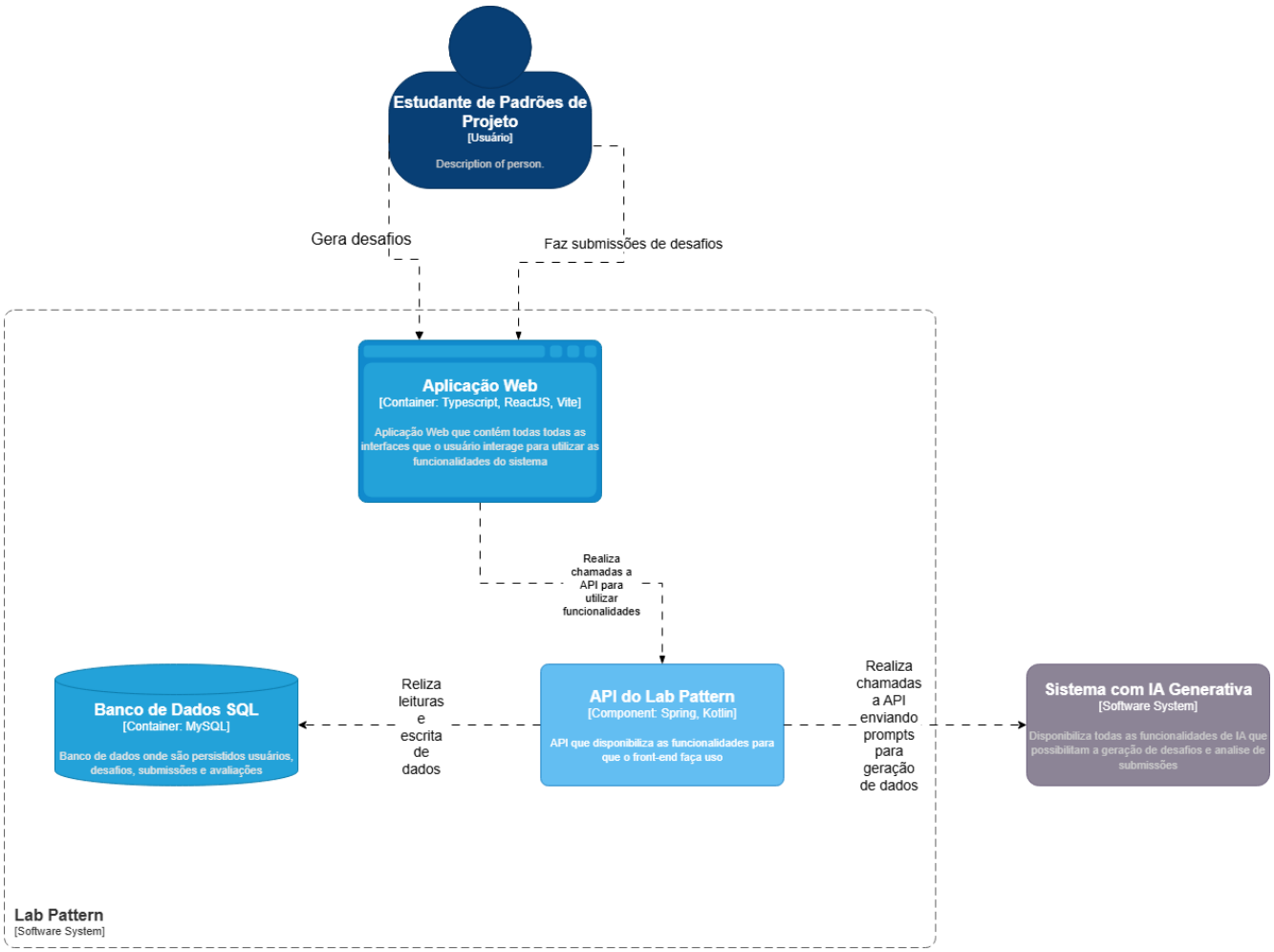
4.2.3 Modelo de Banco de Dados Físico

O modelo físico foi derivado do modelo lógico. Seu objetivo é traduzir a estrutura abstrata para uma implementação real e otimizada para o SGBD MySQL. Nesta etapa, as entidades do modelo ER foram convertidas em tabelas e seus atributos foram mapeados para colunas com os tipos de dados específicos do MySQL (como VARCHAR, INT, TIMESTAMP, etc.). Além disso, foram definidas as chaves primárias e estrangeiras para garantir a integridade e o correto relacionamento entre as tabelas. O esquema a seguir representa a estrutura final do banco de dados implementado.



4.2.4 Visão arquitetural

A arquitetura utilizada pelo sistema pode ser classificada como uma arquitetura monolítica modular, na qual todos os recursos visuais estão concentrados em um projeto frontend(módulo frontend) desenvolvido com ReactJS[17]. As regras de negócio e a camada de persistência de dados encontram-se em um serviço backend(módulo backend), implementado com Kotlin[8] e Spring Boot[11]. O backend da aplicação utiliza um banco de dados MySQL[15] e integra-se a uma API externa da OpenAI. A seguir, é apresentada a representação visual do sistema, utilizando o diagrama de contêiner do modelo C4:



O backend da aplicação adota uma arquitetura em camadas. Nesse modelo, as regras de negócio são isoladas de componentes externos, como classes responsáveis pela conexão com o banco de dados, pontos de entrada da API e integrações com serviços externos. Essa abordagem favorece a característica desejada de permitir, com facilidade, a substituição do provedor de IA generativa no futuro.

O frontend da aplicação segue um padrão comum em projetos de frontend, com a separação de responsabilidades em diretórios específicos. Por exemplo, o diretório components armazena os componentes utilizados nas interfaces, enquanto o diretório services contém as integrações com serviços externos — neste caso, a API da aplicação.

4.3 Conclusão

Nesse capítulo foi possível demonstrar quais as estratégias utilizadas para permitir que a aplicação tenha a capacidade de gerar desafios e analisar submissões, duas características centrais da proposta de sistema. Por meio dos diagramas é possível entender como as principais partes do domínio do sistema foram mapeadas entre classes, seus casos de uso, e como a integração com um provedor de IA funciona. A Visão arquitetural demonstra como os componentes se integram para entregar um sistema que simula um laboratório completo para a prática de padrões de projeto.

5 Requisitos

5.1 Requisitos Funcionais

Acesso à prática de desafios

- **Como** estudante, **quero** acessar uma página de prática, **para que** eu possa gerar um desafio aleatório de design pattern.
- **Como** estudante, **quero** gerar um novo desafio na página de prática, **para que** eu possa praticar um padrão sem precisar escolher um manualmente.

Desafio diário

- **Como** estudante, **quero** acessar a página do desafio diário, **para que** eu possa praticar o mesmo desafio que os outros estudantes naquele dia.
- **Como** sistema, **quero** atualizar automaticamente o desafio diário a cada 24 horas, **para que** todos os usuários vejam um novo desafio diariamente.

Visualização e submissão de desafios

- **Como** estudante, **quero** visualizar a descrição e os requisitos de um desafio, **para que** eu entenda o que precisa ser implementado.
- **Como** estudante, **quero** submeter minha resposta para um desafio, **para que** eu receba um feedback sobre minha implementação.
- **Como** estudante, **quero** poder selecionar o design pattern do desafio, **para que** eu possa me lembrar do que tenho que implementar.
- **Como** estudante, **quero** poder selecionar mais de uma linguagem de programação, **para que** eu possa utilizar uma linguagem que estou familiarizado na implementação.

Avaliação automática e feedback

- **Como** sistema, **quero** analisar a resposta submetida pelo estudante, **para que** eu possa verificar se o padrão foi aplicado corretamente.
- **Como** estudante, **quero** receber um feedback claro e imediato, **para que** eu saiba o que acertei e o que posso melhorar.
- **Como** estudante, **quero** ser direcionado para uma tela única de feedback após enviar uma resposta, **para que** eu tenha uma experiência consistente ao revisar meu desempenho, independente de onde o desafio foi acessado.

Histórico de desafios

- **Como** estudante, **quero** acessar uma página com os desafios que já concluí, **para que** eu possa revisar meu histórico de prática.
- **Como** estudante, **quero** visualizar o feedback recebido em cada desafio concluído, **para que** eu possa aprender com meus erros e acertos anteriores.
- **Como** estudante, **quero** ver a minha solução submetida ao lado do feedback, **para que** eu entenda claramente o que foi avaliado.

Dashboard inicial com dados de uso

- **Como** estudante, **quero** ver um resumo dos meus dados de uso ao entrar no sistema, **para que** eu acompanhe meu progresso de forma rápida.
- **Como** estudante, **quero** visualizar quantos desafios já concluí, **para que** eu saiba o quanto pratiquei.
- **Como** estudante, **quero** ver minha streak atual (dias seguidos praticando), **para que** eu me motive a manter uma rotina de estudos.
- **Como** estudante, **quero** visualizar minha média de nota nas submissões, **para que** eu avalie meu desempenho geral.

5.2 Requisitos Não-Funcionais

Usabilidade

- O sistema deve apresentar uma interface intuitiva para estudantes, com navegação clara entre páginas.
- O feedback deve ser exibido em até 20 segundos após a submissão do desafio.

Desempenho

- A geração do desafio aleatório deve ocorrer em no máximo 20 segundos.
- O sistema deve suportar até 100 usuários simultâneos sem degradação perceptível da performance.

Segurança

- As credenciais dos usuários devem ser armazenadas de forma criptografada.
- O acesso às páginas de desafios e feedback deve ser restrito a usuários autenticados.

Manutenibilidade

- O código deve ser modularizado em camadas para facilitar a substituição do motor de avaliação.
- Deve ser possível adicionar novos tipos de desafios sem alteração significativa no frontend.

Portabilidade

- O sistema deve ser compatível com os navegadores Chrome, Firefox e Edge nas versões mais recentes.
- Deve ser possível acessar o sistema em dispositivos móveis e desktops com responsividade adequada.

Confiabilidade

- O sistema deve garantir que o desafio diário seja atualizado corretamente a cada 24 horas.
- Submissões dos usuários não devem ser perdidas mesmo em caso de falha temporária de rede.

Escalabilidade

- A arquitetura deve permitir expansão para múltiplos servidores para suportar aumento do número de usuários.
- A base de dados deve suportar crescimento do volume de desafios e respostas sem perda de desempenho.

6 Qualidade

6.1 Visão Geral

Manter a qualidade de sistema é uma tarefa que demanda esforços em várias frentes, o projeto visa manter a qualidade em duas frentes, uma é por meio de testes automatizados. Esses testes podem ser rodados com apenas um comando e garantem que nenhuma mudança feita no código leve a aplicação a um estado de falha. A outra frente é evitar que códigos com problemas possam ser enviados para a versão mais recente do sistema. Para isso, utilizamos duas ferramentas para trabalhar nessas frentes.

6.2 Trabalhando com o JUnit

A primeira ferramenta utilizada é o JUnit[19], uma biblioteca amplamente utilizada na linguagem Java para a criação e execução de testes automatizados. No projeto, o JUnit é empregado tanto para testes unitários quanto para testes de integração. Os testes unitários validam o comportamento de unidades específicas de código, como serviços de geração de desafios e serviços que interagem com dependências externas, como banco de dados e APIs. Já os testes de integração verificam o funcionamento completo de um fluxo, iniciando pelos endpoints da API e avaliando a interação entre os diversos componentes do sistema. Para isso, são utilizados scripts SQL que configuram cenários reais no banco de dados, simulando situações comuns da aplicação [19].

6.3 Testes Unitários

O projeto contém testes unitários cobrindo as classes de caso de uso e as classes de serviços. As classes de caso de uso são testadas via chamada direta do método que executa a regra de negócio contida nela, antes de chamar esse método, instanciamos objetos contendo resultados esperados e mocks, que são resultados pré definidos para chamadas a classes de caso de uso ou interfaces que compõem a lógica da classe que está sendo testada. O trecho de código abaixo é um dos testes unitários feitos sobre a classe de geração de desafios

```
34          fun `should generate challenge successfully`() {
35              val pattern = DesignPattern(
36                  id = UUID.randomUUID(),
37                  name = "Strategy",
38                  category = "Behavioral",
39                  description = "Strategy pattern description"
40              )
41              val aiResponse = AiQuestionDto(
42                  title = "Shopping Cart Discount",
43                  description = "Implement a shopping cart with different discount strategies"
44              )
45              val challenge = Challenge(
46                  id = UUID.randomUUID(),
47                  expectedPatternId = pattern.id,
48                  title = aiResponse.title,
49                  description = aiResponse.description,
50                  createdAt = LocalDateTime.now(),
51                  publishedAt = LocalDateTime.now()
52              )
53
54              every { getAllPatternsGateway.execute() } returns Result.success(listOf(pattern))
55              every { getAiQuestionGateway.execute(pattern, any()) } returns Result.success(aiResponse)
56              every {
57                  storeChallengeGateway.execute(match {
58                      it.expectedPatternId == pattern.id &&
59                              it.title == aiResponse.title &&
60                              it.description == aiResponse.description
61                  })
62              } returns Result.success(challenge)
63
64              val result = useCase.execute()
65
66              Assertions.assertTrue(result.isSuccess)
67              Assertions.assertEquals(challenge, result.getOrNull())
68              verify {
69                  getAllPatternsGateway.execute()
70                  getAiQuestionGateway.execute(pattern, any())
71                  storeChallengeGateway.execute(any())
72              }
73          }
```

O trecho da linha 35 a linha 52 pré-define o resultado das chamadas de interfaces utilizadas na lógica do caso de uso, o trecho da linha 54 a 61 configura esse mock, a linha 64 executa a lógica e armazena o resultado na constante result, após isso fazemos as asserções sobre os resultados e verificamos que as interfaces que realizam alguns passos da lógica foram chamadas

6.4 Testando a Integração com IA Generativa

A integração com IA generativa constitui um componente crítico do sistema, exigindo uma abordagem de testes em duas camadas. Testes unitários com execução contínua que utilizam mocks para simular respostas da API, validando o comportamento da aplicação frente

a diferentes cenários sem gerar custos. Testes de validação da LLM com execução seletiva, que submetem um golden dataset contendo casos representativos (exemplos canônicos de padrões, casos ambíguos e casos negativos) à API real do provedor, validando se as análises produzidas correspondem aos resultados esperados. Dado o caráter custoso e mais lento de requisições a APIs externas, o segundo perfil é executado seletivamente em momentos críticos: alterações em parâmetros de configuração da IA (temperatura, prompt, tokens), mudanças de provedor ou versão do modelo, antes de deploys para produção. Esta abordagem híbrida permite feedback rápido durante o desenvolvimento mantendo garantias de qualidade da integração com IA sem comprometer velocidade ou incorrer em custos desnecessários.

6.5 Utilizando o SonarQube

A segunda ferramenta é o SonarQube[20], uma plataforma de análise contínua de código que avalia a qualidade do software com base em métricas como cobertura de testes, duplicação de código, vulnerabilidades e falhas técnicas. O SonarQube está configurado no repositório do projeto de forma que todo código submetido seja analisado automaticamente. Além disso, foi estabelecido um critério mínimo de 80% de cobertura de testes: caso esse percentual não seja atingido, o código não pode ser integrado à versão principal do sistema. Essa política ajuda a manter um alto padrão de confiabilidade e manutenibilidade ao longo do desenvolvimento [20].

6.6 Mantendo a Qualidade

Com o uso dessas duas ferramentas, foi possível manter a qualidade do código ao longo do desenvolvimento do projeto. O fluxo de trabalho adotado segue os seguintes passos:

1. Implementar o código das novas funcionalidades;
2. Escrever testes unitários e/ou de integração;
3. Executar os testes automatizados para garantir que nenhuma funcionalidade existente foi comprometida;
4. Subir o código em uma nova versão utilizando o Git como sistema de controle de versão;
5. Criar uma *Pull Request* (solicitação de integração do código à versão principal do sistema) no GitHub;
6. Revisar o código e verificar se o SonarQube não identificou nenhuma falha ou violação de qualidade;
7. Integrar o código à versão principal do sistema.

Esse processo, aliado à automação e às ferramentas de análise estática e de testes, contribui diretamente para a confiabilidade, manutenibilidade e robustez da aplicação.

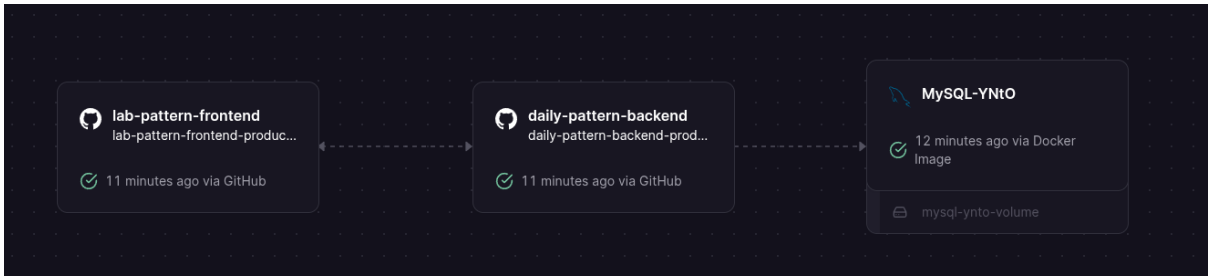
7 Implantação e Manual de Uso

Esse capítulo cobre a descrição de como o projeto foi implantado, seus requisitos para funcionamento, como ele foi disponibilizado com a ferramenta Railway[21] e seu manual de uso.

7.1 Projeto de implantação

O projeto foi implantado utilizando a ferramenta Railway[21], uma plataforma em nuvem que facilita o processo de implantação, fornecendo serviços automatizados para a execução de projetos a partir de repositórios de código armazenados no GitHub, além da possibilidade de instanciar serviços de banco de dados ou imagens Docker[22].

Para utilizar a ferramenta, foi necessário importar os repositórios do projeto para o ambiente do Railway, neste caso, o backend e o frontend da aplicação. Em seguida, criou-se uma instância do banco de dados MySQL dentro do mesmo ambiente. Em cada implantação, as variáveis de ambiente foram configuradas de modo que o frontend apontasse para a API disponibilizada pelo backend e que o backend utilizasse o banco de dados MySQL. A Figura abaixo ilustra como o ambiente do Railway ficou após a conclusão da implantação:



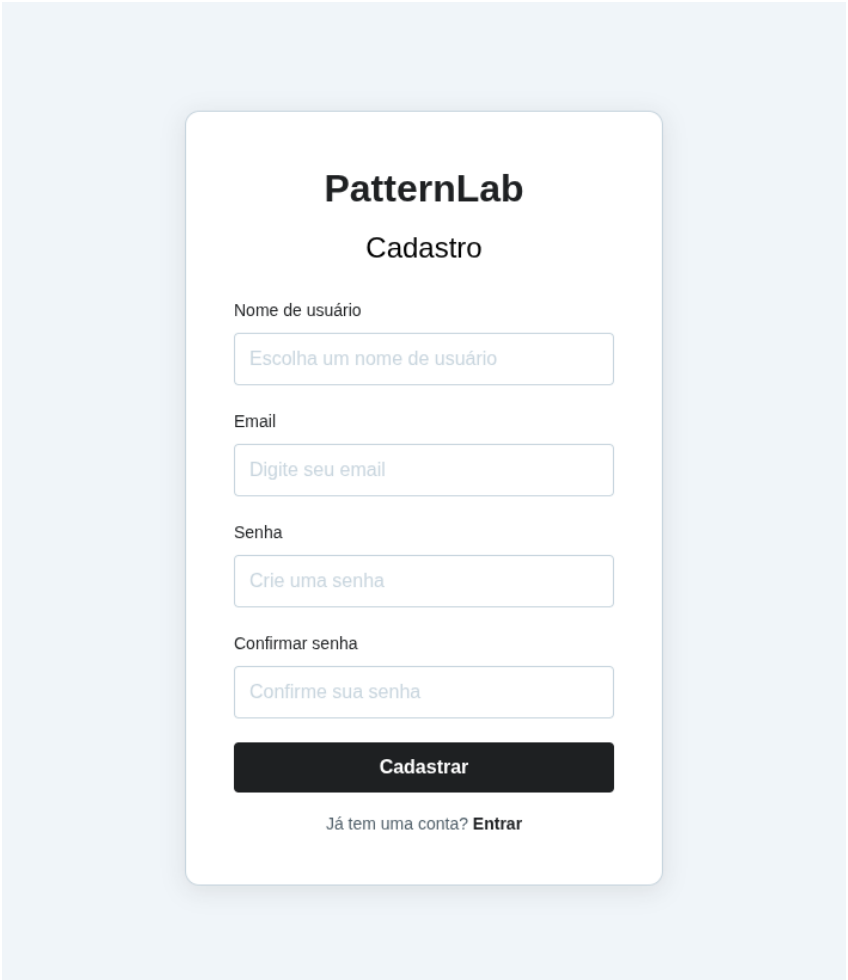
Após a implantação, o último passo foi disponibilizar o sistema web em um domínio público. Essa etapa também foi realizada por meio do Railway, que possui a funcionalidade de gerar um domínio e tornar a aplicação acessível de forma pública, permitindo que os usuários acessem o sistema via navegador web. Com isso, a aplicação passou a estar disponível para uso. Na próxima seção, é apresentado o manual do usuário, contendo as funcionalidades da aplicação.

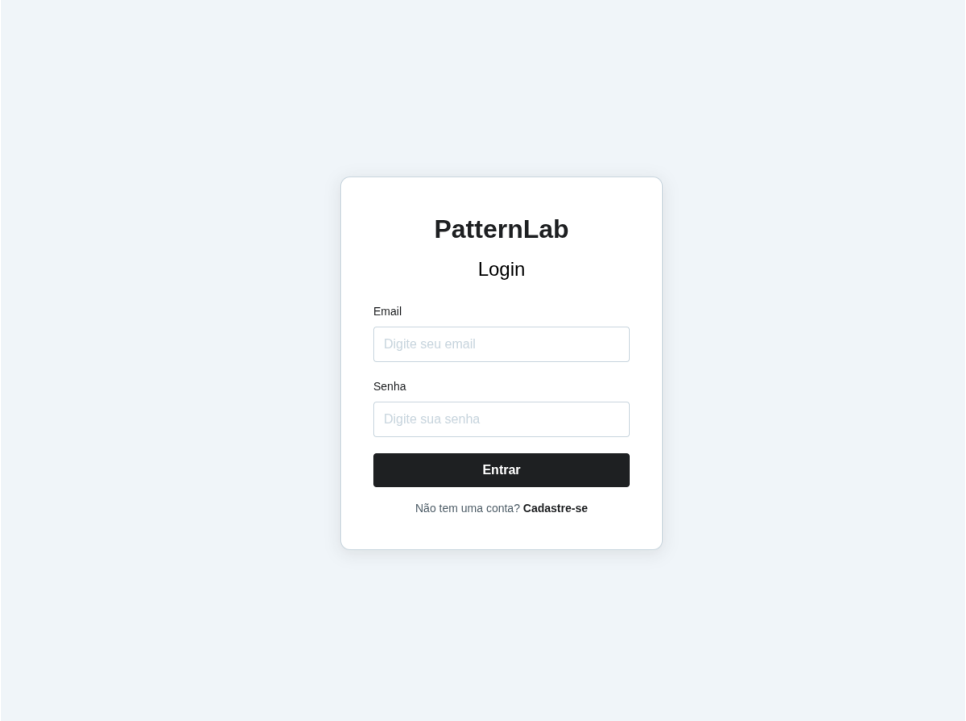
7.2 Manual do Usuário

Abaixo estão as principais as demonstrações das interfaces para manuseio do sistema.

7.2.1 Login e Cadastro

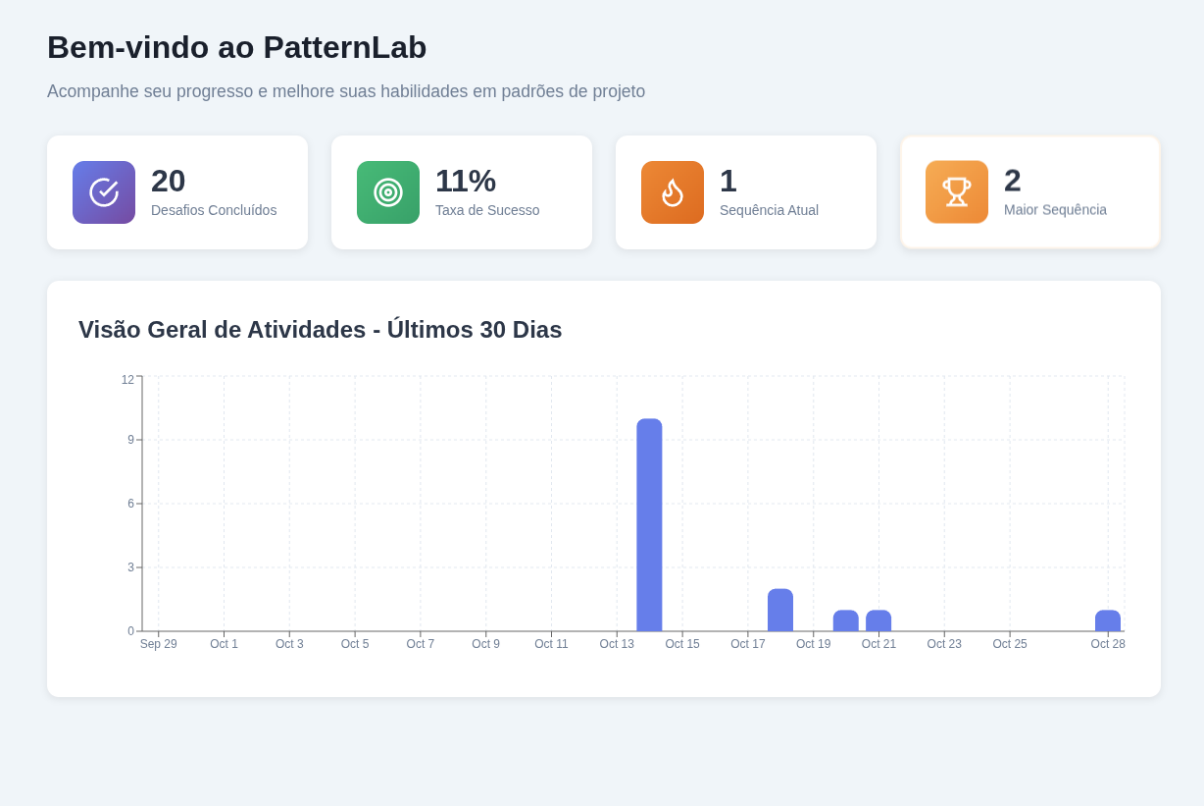
Essas duas interfaces são o ponto de entrada no sistema, o usuário deve se cadastrar com os dados necessários para o primeiro acesso. Após o primeiro acesso, é necessário que o usuário entre no sistema com os dados cadastrados.


PatternLab
Cadastro
Nome de usuário
Escolha um nome de usuário
Email
Digite seu email
Senha
Crie uma senha
Confirmar senha
Confirme sua senha
Cadastrar
Já tem uma conta? Entrar


PatternLab
Login
Email
Digite seu email
Senha
Digite sua senha
Entrar
Não tem uma conta? Cadastre-se

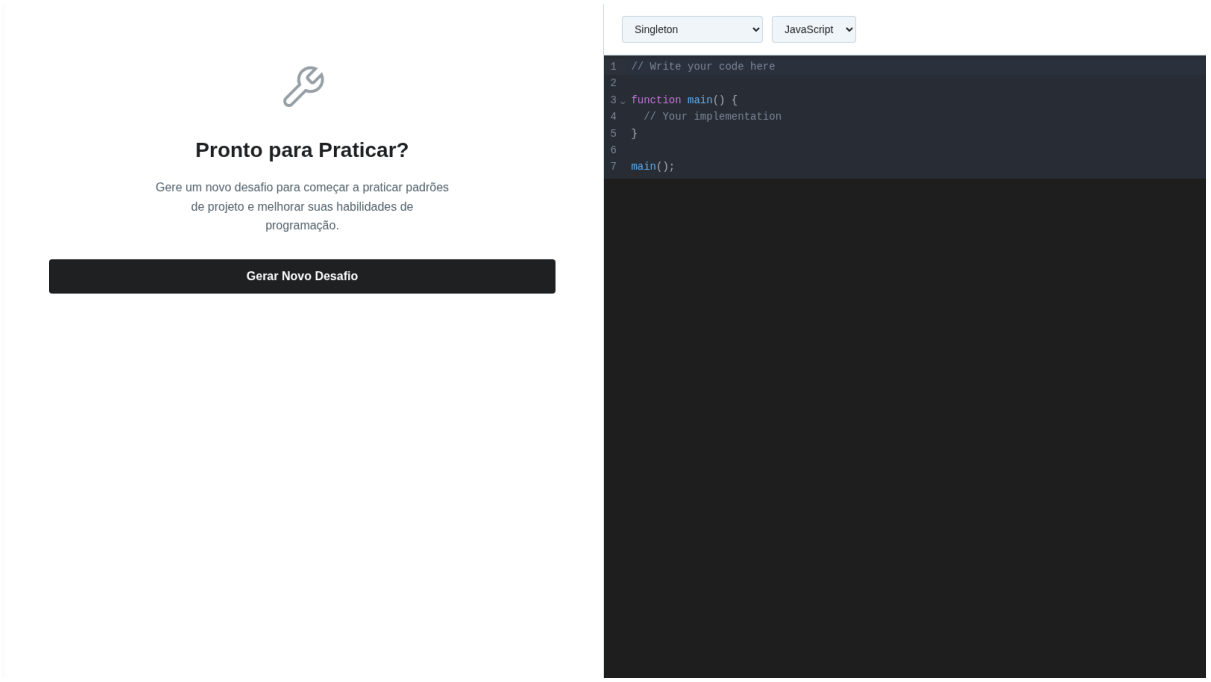
7.2.2 Início

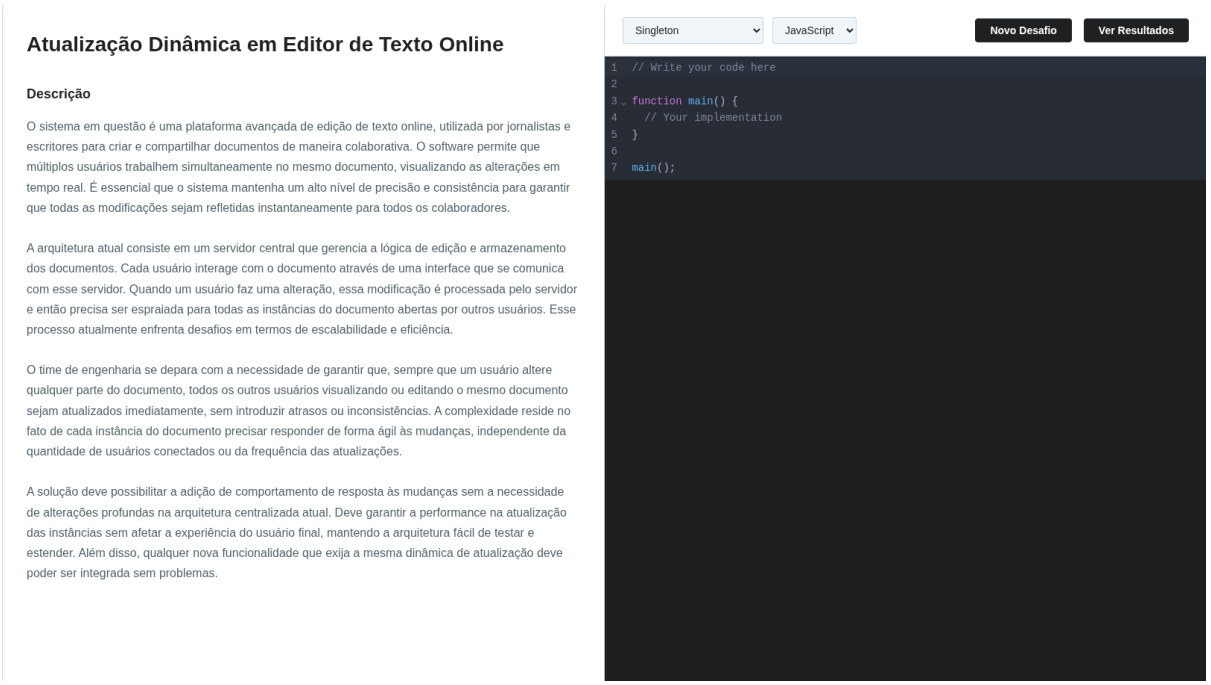
O Início é a interface que o usuário tem acesso logo após autenticado no sistema. Nela são exibidos alguns dados do uso do sistema e um gráfico exibindo a quantidade de desafios resolvidos nos últimos 30 dias.



7.2.3 Prática

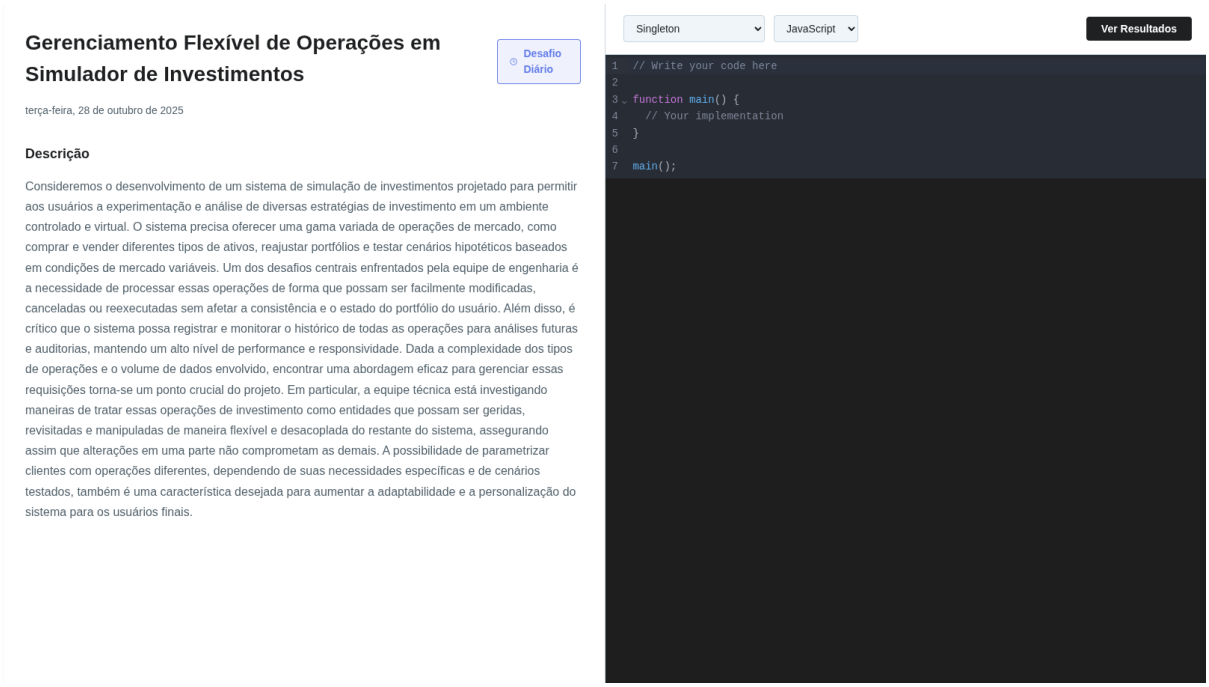
Essa é a página onde o usuário pode praticar de fato padrões de projeto, quando o usuário não tem nenhum desafio gerado, o sistema exibe um botão para gerar um desafio aleatório escolhendo um tema e padrão para praticar aleatoriamente. Após carregado o desafio, o sistema exibe na esquerda a descrição do problema e na direita um componente de editor de código para que o usuário implemente algo real ou um pseudo-código com a solução.





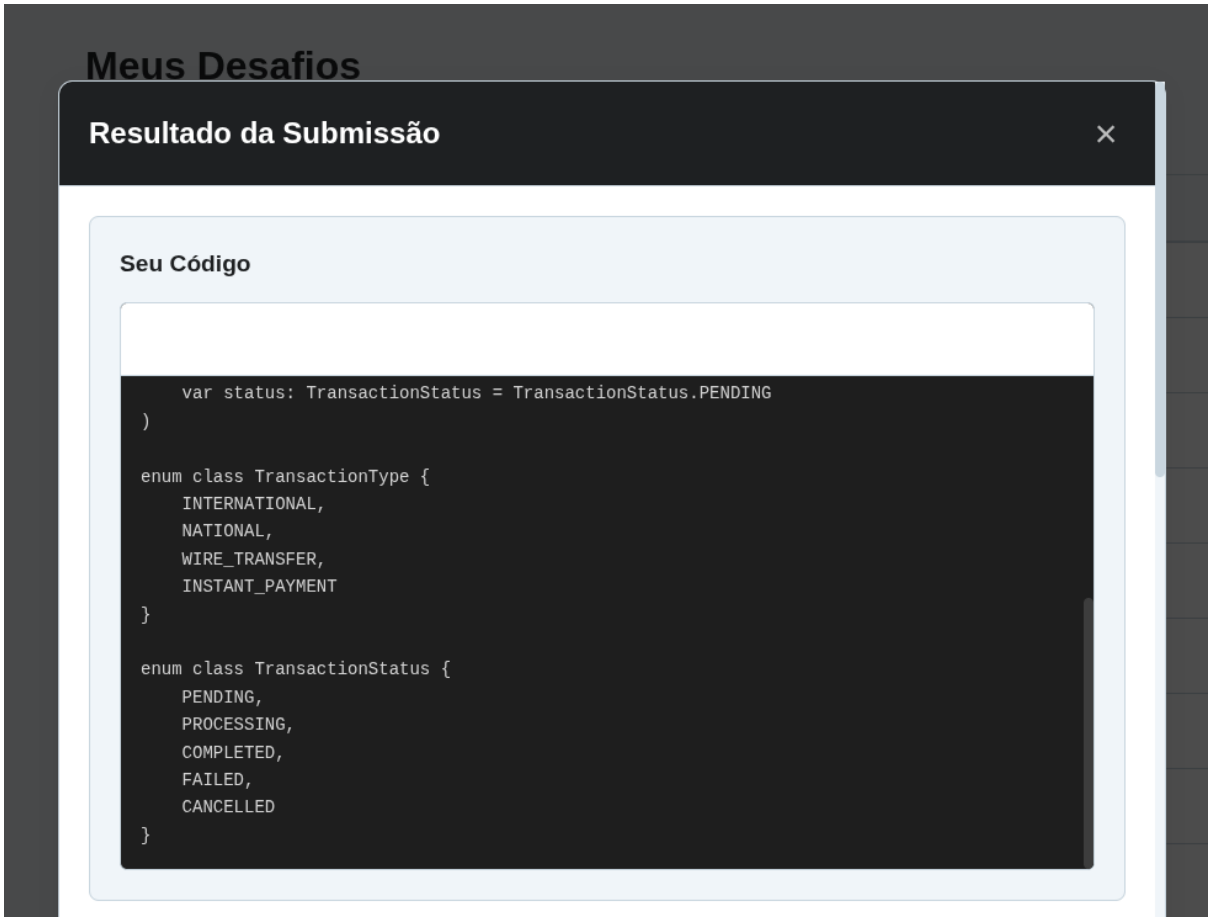
7.2.4 Desafio Diário

A interface de desafio diário é similar a de prática. A maior diferença é que o desafio é sempre gerado pelo sistema diariamente, então não existem ações para gerar desafios novos e, uma vez resolvido pelo usuário, o desafio continua em tela até que um novo desafio seja gerado.



7.2.5 Resultado de submissão

Após a submissão de um desafio, o sistema exibe um modal com os resultados da análise do código escrito pelo usuário.



7.2.6 Desafio

A interface de desafios exibe todos os desafios já realizados pelo usuário no sistema de forma paginada, exibindo a data de publicação e o tipo de desafio, permitindo também que o usuário visualize o resultado ao clicar no desafio que deseja rever.

Meus Desafios

Veja todos os desafios que você completou

TÍTULO	TIPO	DATA DE PUBLICAÇÃO
Atualização Dinâmica em Editor de Texto Online	PRÁTICA	28/10/2025
Simulador de Entrevistas com Histórico de Estados	PRÁTICA	21/10/2025
Otimização de Sistema de Cadastro em Hackathons	PRÁTICA	20/10/2025
Adaptação Dinâmica na Avaliação de Desempenho Estudantil	PRÁTICA	18/10/2025
Integração de Sistemas Bancários Desatualizados	PRÁTICA	18/10/2025
Desenvolvimento de um sistema de interpretação de comandos para aplicativo financeiro	PRÁTICA	14/10/2025
Adaptação Dinâmica dos Métodos de Avaliação em uma Plataforma de Debates Online	PRÁTICA	14/10/2025
Gerenciamento de Estado em Sistema de Aprendizado de Idiomas	PRÁTICA	14/10/2025
Eficiência na gestão de objetos em aplicativo de hábitos pessoais	PRÁTICA	14/10/2025
Gerenciamento Flexível de Operações em Simulador de Investimentos	DIÁRIO	14/10/2025

Anterior Página 1 de 2 Próxima

8 Anexos

Este anexo apresenta os repositórios de código-fonte desenvolvidos no contexto deste trabalho. Todo o código está disponível publicamente, permitindo consulta, reprodução e evolução do projeto.

8.1 Repositório do Backend

O repositório do backend contém o código-fonte da API desenvolvida com Kotlin e Spring Boot, responsável pelas regras de negócio, integração com o banco de dados MySQL e integração com o provedor de IA Generativa.

⟨https://github.com/RafaelCamposs/lab-pattern-backend⟩

8.2 Repositório do Frontend

O repositório do frontend contém o código-fonte da aplicação web desenvolvida com TypeScript e ReactJS, responsável por todas as interfaces de interação do usuário com o sistema.

⟨https://github.com/RafaelCamposs/lab-pattern-frontend⟩

9 Agradecimentos

O desenvolvimento deste trabalho contou com a colaboração e o apoio de muitas pessoas, às quais expresso meus sinceros agradecimentos.

Ao meu orientador, Prof. Dr. Lauro Cássio Martins de Paula, agradeço pela dedicação, paciência e orientação ao longo de todo o processo de elaboração deste projeto. Sua orientação foi essencial para o amadurecimento da pesquisa e para a qualidade deste trabalho.

Aos meus pais, agradeço pelos inúmeros sacrifícios, pelo incentivo constante e por nunca medirem esforços para me proporcionar a melhor educação possível. Os ensinamentos e o apoio de vocês foram fundamentais para que eu chegasse até aqui. Sou eternamente grato por poder dar continuidade ao sonho que vocês começaram.

Por fim, agradeço aos professores e colegas do curso de Análise e Desenvolvimento de Sistemas, que compartilharam conhecimentos e experiências que levarei comigo por toda a vida.

Referências

1 Scrum.org. What is scrum? **Scrum.org Learning Series**, 2024. Acesso em: 17 set. 2025. Disponível em: ⟨https://www.scrum.org/learning-series/what-is-scrum/⟩.

2 Wells, Don. Extreme programming: A gentle introduction. **ExtremeProgramming.org**, 2009. Acesso em: 17 set. 2025. Disponível em: ⟨http://www.extremeprogramming.org/⟩.

3 The Agile Manifesto Authors. Princípios por trás do Manifesto Ágil. **Manifesto Ágil (site oficial)**, 2025. Tradução em Português Brasileiro dos 12 princípios do Manifesto Ágil. Disponível em: ⟨https://agilemanifesto.org/iso/ptbr/principles.html⟩.

4 GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. [S.l.]: Addison-Wesley, 1994. ISBN 978-0201633610.

5 IEEE Computer Society Team. The Evolution of AI: From Foundations to Future Prospects. **IEEE Computer Society - Tech News: Research**, Mar 2025. Disponível online. Disponível em: ⟨https://www.computer.org/publications/technews/research/evolutionofai⟩.

6 Google Cloud. What is Retrieval-Augmented Generation (RAG)? **Google Cloud Use Cases**, 2025. Acesso em `17 set. 2025`. Disponível em: ⟨https://cloud.google.com/use-cases/retrieval-augmented-generation?hl=pt-BR⟩.

7 Model Context Protocol Team. What is the Model Context Protocol (MCP)? **Model Context Protocol Documentation**, 2025. Acesso em 17 set. 2025. Disponível em: ⟨https://modelcontextprotocol.io/docs/getting-started/intro⟩.

8 Kotlin Team. K2 compiler migration guide – Kotlin Documentation. **Kotlin Documentation**, 2025. Guia oficial de migração para o compilador K2. Disponível em: ⟨https://kotlinlang.org/docs/k2-compiler-migration-guide.html⟩.

9 SOSHIN, A. **Kotlin Design Patterns and Best Practices: Elevate your Kotlin skills with classical and modern design patterns, coroutines, and microservices**. 3rd. ed. [S.l.]: Packt Publishing, 2024.

10 Kotlin Team. Object declarations and expressions – Kotlin Documentation. **Kotlin Documentation**, 2025. Seção “Object declarations and expressions” da documentação oficial do Kotlin. Disponível em: ⟨https://kotlinlang.org/docs/object-declarations.html⟩.

11 Spring Framework Project. **Spring Framework Reference Documentation: Overview**. [S.l.], 2025. Online; seção “Overview” da documentação oficial. Disponível em: ⟨https://docs.spring.io/spring-framework/reference/overview.html⟩.

12 Spring Data Team. Core Concepts – Spring Data JPA Reference Documentation. **Spring Data JPA Reference Documentation**, 2025. Seção “Core Concepts” da documentação oficial do Spring Data JPA. Disponível em: ⟨https://docs.spring.io/spring-data/jpa/reference/repositories/core-concepts.html⟩.

13 Spring Security Team. Getting Started – Spring Security Reference Documentation. **Spring Security Reference Documentation**, 2025. Seção “Getting Started” da documentação oficial do Spring Security. Disponível em: ⟨https://docs.spring.io/spring-security/reference/getting-spring-security.html⟩.

14 JONKER, A.; MUCCI, T. O que é linguagem de consulta estruturada (SQL)? **IBM Think**,

2025. Acesso em: 17 set. 2025. Disponível em: ⟨https://www.ibm.com/br-pt/think/topics/structured-query-language⟩.

15 ERICKSON, J. MySQL: Entendendo o que é e como é usado. **Oracle Brasil**, ago. 2024. Acesso em 30 jul. 2025. Disponível em: ⟨https://www.oracle.com/br/mysql/what-is-mysql/⟩.

16 TypeScript Team. The TypeScript Handbook: Introduction. **TypeScript Documentation**, 2025. Seção “Introduction” do Handbook oficial. Disponível em: ⟨https://www.typescriptlang.org/docs/handbook/intro.html⟩.

17 Facebook / Meta and React Community. React: The library for web and native user interfaces. **GitHub repository**, jul. 2025. Repositório oficial de código fonte do React no GitHub. Disponível em: ⟨https://github.com/facebook/react⟩.

18 BOONSTRA, L. **Prompt Engineering**. [S.l.], 2024. Whitepaper para o programa ”5-Day Gen AI Intensive”do Kaggle. Disponível em: ⟨https://www.kaggle.com/whitepaper-prompt-engineering⟩.

19 TEAM, J. **JUnit 5 User Guide**. 2024. ⟨https://junit.org/junit5/docs/current/user-guide/⟩. Acesso em: 04 ago. 2025.

20 SONARSOURCE. **SonarQube Documentation**. 2024. ⟨https://docs.sonarsource.com/sonarqube/⟩. Acesso em: 04 ago. 2025.

21 Railway Team. About Railway: Overview of the Railway Deployment Platform. **Railway Documentation**, 2025. Visão geral da plataforma Railway — incluindo implantação, CI/CD, observabilidade e escalabilidade. Disponível em: ⟨https://docs.railway.com/overview/about-railway⟩.

22 Docker Documentation Team. Building images – Docker Documentation. **Docker Docs**, 2025. Guia “Building images” na seção “Docker concepts” da documentação oficial. Disponível em: ⟨https://docs.docker.com/get-started/docker-concepts/building-images/⟩.