Cinemais - Desenvolvimento de Plataforma de Gestão de Cinemas

Enzo Micael Silva Gaeta de Moraes; Jovanny Leite Marques da Silva; Lucas Galatro da Costa; Luiz Gustavo da Silva Vasconcellos;
Orientador: Elienai Lacerda Neves.

Resumo: Este trabalho descreve o desenvolvimento e implementação do sistema Cinemais, uma aplicação voltada para a indústria cinematográfica. O projeto utilizou metodologias de pesquisa bibliográfica e descritiva, embasando as escolhas de tecnologias, a utilização de metodologias ágeis e princípios de DevOps, se baseando em uma abordagem colaborativa e eficiente. O sistema foi construído com tecnologias modernas como Node.Js, TypeScript, Firebase e React, destacando-se por sua personalização e adaptabilidade, permitindo que diferentes cinemas ajustem as funcionalidades conforme suas necessidades. O processo de publicação foi automatizado, garantindo a qualidade do código e a consistência do banco de dados em ambientes de teste e produção, considerando a implementação de práticas de segurança, como a autenticação e o controle de acesso, protegendo os dados sensíveis dos usuários. A implementação desse projeto demonstra a importância da união entre a implementação técnica do sistema e dos processos e princípios DevOps para entregas assertivas, possibilitando mudanças rápidas com segurança.

Palavras-chave: Cinema, DevOps, Desenvolvimento Ágil, React, Express.

Cinemais - Development of Cinema Management Platform

Abstract: This work describes the development and implementation of the Cinemais system, an application aimed at the film industry. The project used bibliographic and descriptive research methodologies, supporting technology choices, the use of agile methodologies and DevOps principles, based on a collaborative and efficient approach. The system was built with modern technologies such as Node.Js, TypeScript, Firebase and React, standing out for its customization and adaptability, allowing different cinemas to adjust the functionalities according to their needs. The publication process was automated, ensuring code quality and database consistency in test and production environments, considering the implementation of security practices, such as authentication and access control, protecting users' sensitive data. The implementation of this project demonstrates the importance of combining the technical implementation of the system and DevOps processes and principles for assertive deliveries, enabling rapid changes with security.

Keywords: Cine, DevOps, Agile Development, React, Express.

1. Introdução

A gestão eficiente de complexos cinematográficos é uma tarefa desafiadora que envolve uma série de aspectos, desde a programação de sessões até a venda de ingressos. Segundo Laurindo et al. (2001), o uso de tecnologias é uma forma das empresas obterem vantagens competitivas, implicando no ganho de produtividade e competitividade do negócio.

Nesse cenário, o sistema Cinemais emerge como uma solução abrangente, concebida como uma central de administração de sistemas para cinemas.Este sistema permite que cinemas registrados gerenciem suas informações, incluindo detalhes sobre salas, capacidade de assentos, filmes em cartaz e a venda de ingressos. Além disso, a integração de todos os cinemas cadastrados no sistema possibilita a disponibilização dessas informações em nosso site e aplicativo, oferecendo aos clientes a conveniência de buscar sessões de filmes em cartaz e realizar a compra de ingressos, incluindo a escolha de assentos disponíveis.

Neste contexto, este artigo explora em detalhes o sistema Cinemais, examinando suas funcionalidades e o papel que desempenha na otimização da gestão de cinemas. Além disso, abordaremos a hierarquia de perfis de usuário dentro do sistema, destacando como esses perfis desempenham um papel fundamental na operação eficaz dos cinemas.

Para fundamentar o processo de desenvolvimento do projeto, foram utilizadas as metodologias de pesquisas bibliográficas e descritivas, definindo uma base concreta de conhecimento para a implementação do projeto, produzindo materiais e discussões com base na natureza aplicada desses conhecimentos.

A seguir, apresentamos uma visão geral dos objetivos deste trabalho, sua justificativa e a estrutura que será seguida para a análise e discussão do sistema Cinemais, demonstrando como esta pesquisa contribuirá para a compreensão e aplicação eficaz desta ferramenta na indústria cinematográfica.

1.1 Justificativa

A indústria cinematográfica é uma das formas de entretenimento mais populares e influentes em todo o mundo, desempenhando um papel significativo na cultura contemporânea. Com a crescente complexidade na administração de complexos cinematográficos, a eficiência na gestão torna-se crucial para oferecer experiências de alta qualidade aos espectadores e garantir a sustentabilidade econômica das operações.

O sistema Cinemais foi desenvolvido com o propósito de abordar esses desafios, proporcionando uma solução abrangente para a administração de cinemas. No entanto, a justificativa para esta pesquisa vai além da mera descrição das funcionalidades do sistema. Ela se baseia nos seguintes motivos:

Relevância da Indústria Cinematográfica: A indústria cinematográfica continua a desempenhar um papel central na cultura e no entretenimento contemporâneos. Portanto, a melhoria da gestão de cinemas é relevante tanto para a indústria quanto para o público em geral.

Necessidade de Eficiência Operacional: A gestão eficiente de complexos cinematográficos é essencial para otimizar custos, garantir a satisfação dos espectadores e manter a competitividade no mercado.

Desenvolvimento do Sistema Cinemais: O sistema Cinemais, desenvolvido internamente, é uma resposta direta à necessidade de uma solução personalizada e inovadora para a gestão de complexos cinematográficos. Esta ferramenta foi criada para atender de maneira abrangente às demandas específicas do projeto, destacando nossa capacidade de inovação e personalização na administração de complexos cinematográficos.

Potencial de Impacto: Ao desenvolver o Cinemais, o projeto pode beneficiar diretamente empresas do ramo cinematográfico, capacitando-os a maximizar o uso e os benefícios dessa ferramenta personalizada. Além disso, ao compartilhar nossas descobertas e experiências, podemos inspirar outros desenvolvedores a explorar soluções internas para desafios específicos.

1.2 Objetivos (Geral e específicos)

Diante desse contexto, o objetivo geral deste projeto é conceber, desenvolver e implementar um sistema de gestão abrangente destinado a empresas operantes no setor cinematográfico, proporcionando controle eficaz sobre uma variedade de aspectos críticos da operação de cinemas.

1.2.1 Objetivos Específicos

Para concretizar o objetivo geral do projeto, será necessário:

1. Analisar as melhores ferramentas e práticas do mercado de desenvolvimento de software
2. Demonstrar a importância da modelagem de dados no desenvolvimento de software
3. Buscar as melhores práticas para a implementação de projetos de software
4. Garantir a qualidade do projeto publicado
5. Discutir melhorias no projeto e processo de desenvolvimento

2. Revisão Bibliográfica

A elaboração deste projeto demandou uma análise abrangente de conteúdos existentes no campo de desenvolvimento de plataformas tecnológicas correlatas ao projeto apresentado. A revisão bibliográfica, como parte crucial do presente trabalho, foi conduzida para proporcionar uma base sólida para as escolhas metodológicas e técnicas apresentadas.

O processo de revisão bibliográfica foi iniciado mediante a identificação criteriosa de fontes relevantes, abrangendo periódicos científicos, conferências, livros, e outros recursos acadêmicos e técnicos, orientada pelas temáticas centrais do projeto, visando assegurar a confiabilidade das informações levantadas.

De acordo com Neto (2017), trabalhar de forma não bloqueante facilita a execução paralela e o aproveitamento de recursos, sendo um conceito central utilizado durante a pesquisa de materiais para o desenvolvimento do presente trabalho.

Ao longo deste documento serão compartilhadas as descobertas e análises que resultaram do processo de revisão da bibliografia estudada, sendo o alicerce que fundamenta as escolhas metodológicas e tecnológicas aqui apresentadas, enriquecendo o contexto em que o projeto se insere.

2.1 Node.js e Typescript

Seguindo essa visão, o Node.js é projetado para ser não bloqueador e orientado a eventos, o que o torna altamente eficiente na manipulação de operações. De acordo com Marcolino (2021), essa aplicação trata-se de um construtor de aplicações JavaScript V8, tendo sido desenvolvido pela Google e possuindo um gerenciador de pacotes nativo — o NPM (node package manager) —, onde várias bibliotecas estão disponíveis. Node.js revoluciona o cenário de desenvolvimento de aplicativos web, permitindo que os desenvolvedores utilizem a mesma linguagem, tanto no front-end quanto no back-end, reduzindo a curva de aprendizado necessária para seu uso.

Em relação à pertinência do Node.js na atualidade, o website StackOverflow (2023) estabeleceu, com base em uma enquete com participação de desenvolvedores ao redor do mundo, a ferramenta como sendo a mais utilizada por programadores para o desenvolvimento web. Assim, demonstrando ter uma excelente aceitação na percepção na comunidade de desenvolvedores.

Juntamente ao Node.js, o Typescript está presente em seu conjunto de ferramentas, sendo uma linguagem de programação fortemente tipada que se baseia no JavaScript. O código TypeScript é convertido para JavaScript, que pode ser executado em qualquer lugar onde o mesmo seja compatível, de acordo com Adriano (2021) o TypeScript pode ser compreendido como uma extensão do JavaScript, uma linguagem de programação que adiciona recursos de tipagem estática e funcionalidades de programação orientada a objetos ao JavaScript tradicional. Essa extensão foi desenvolvida para atender a uma demanda crescente por maior segurança e robustez no desenvolvimento de aplicativos web e de software em geral.

Uma pesquisa realizada pelo website StackOverflow (Maio, 2023) indica que, em 2023, TypeScript foi considerada a 5ª linguagem mais popular e a 3ª mais desejada pelos desenvolvedores, apontando sua atual relevância no contexto de desenvolvimento de softwares. Além de sua justificabilidade no ramo da programação, o TypeScript possibilita utilizar a mesma linguagem no front-end e no back-end, viabilizando a utilização em projetos já existentes que utilizam JavaScript.

2.2 Tecnologias Backend

Como complemento, o Express.js é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel. Mardan (2014) expressa um dos pontos-chave do Express.js: seu design modular. Ele permite que desenvolvedores usem somente os componentes que precisam para as aplicações e torna fácil a extensão do framework com um middleware personalizado. Isso torna o Express.js um framework altamente customizável que pode ser adaptado para atender às necessidades específicas de qualquer aplicação web.

Outra característica importante do Express.js, dessa vez citado por Brown (2014), é o processo de definir como uma aplicação web deve reagir a uma solicitação, ou seja, oferece um excelente suporte de rotas, fornecendo um sistema de roteamento simples e intuitivo, facilitando o tratamento de solicitações, criação de APIs e construção de páginas web dinâmicas.

2.2.1 PostgreSQL

Conforme descrito no website PostgreSQL (2023), ele é um poderoso sistema de banco de dados relacional de objeto de código aberto com mais de 35 anos de desenvolvimento ativo que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho.

2.2.2 Prisma

Conforme explica Buzzi (2022) Prisma é um ORM (Object-Relational Mapping) que ajuda os desenvolvedores a criar aplicações mais rapidamente e cometer menos erros com um kit de ferramentas de banco de dados open source, como PostgreSQL e MySQL. Além disso, o Prisma também suporta as seguintes linguagens: Javascript e Typescript.

Buzzi (2022) descreve o ORM em três camadas fundamentais: prisma client, um construtor de queries gerado de forma automática e typesafe para Typescript e Nodejs; o prisma migrate (sistema de migração); por fim, o prisma studio, que se trata de uma interface de usuário construída para visualização e edição os dados na database, sendo este o produto principal da tecnologia.

Todo o conceito do Prisma está no “schema” proporcionado por eles no desenrolar dessas três camadas, sendo o principal arquivo de configuração para o Prisma. Assim, sendo descrito como um novo tipo de ORM, possuindo seus fundamentos diferenciados dos modelos tradicionais, o Prisma, nos dias de hoje, está em sua terceira etapa de desenvolvimento, chamada de Prisma 3, com recursos mais avançados e consistentes, aponta Buzzi (2022).

2.2.3 Express

Também conhecido como Express.js, é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.

Mardan (2014) expressa um dos pontos-chave do Express.js: seu design modular. Ele permite que desenvolvedores usem somente os componentes que precisam para as aplicações e torna fácil a extensão do framework com um middleware personalizado. Isso

torna o Express.js um framework altamente customizável que pode ser adaptado para atender às necessidades específicas de qualquer aplicação web.

Outra característica importante do Express.js, dessa vez citado por Brown (2014), é o processo de definir como uma aplicação web deve reagir a uma solicitação, ou seja, oferece um excelente suporte de rotas. O Express.js fornece um sistema de roteamento simples e intuitivo, facilitando o tratamento de solicitações, criação de APIs e construção de páginas web dinâmicas.

2.3 Tecnologias Frontend

Segundo Marcolino (2021), o React permite a utilização variada de elementos do framework e é indicado para o desenvolvimento de single page applications. Sua principal característica é permitir a construção de interfaces altamente interativas e dinâmicas. A biblioteca se destaca por sua abordagem de componentização, que divide a interface em pequenos componentes reutilizáveis. Além disso, utiliza o conceito de Virtual DOM para otimizar as atualizações da interface, melhorando o desempenho da aplicação.

O React é uma biblioteca JavaScript para criar interfaces de usuário. A biblioteca facilita a criação de UIs interativas, atualizando e renderizando de forma eficiente apenas os componentes necessários na medida em que os dados mudam, sendo uma biblioteca JavaScript amplamente reconhecida no desenvolvimento web, especialmente na criação de interfaces de usuário. Document Object Model (DOM) é uma das principais ferramentas que o React lida.

Conforme explica Danielsson (2016), o DOM é uma estrutura em forma de árvore que representa todo o esqueleto de uma página web e seus estados usando elementos HTML. Kumar e Singh (2016) adicionam, mostrando que o intuito através do React é criar uma cópia, o Virtual DOM, e fazer as correções e alterações dos dados e atualizar os elementos de interface através da mesma, ao invés de manipular o DOM diretamente, evitando um trabalho maçante.

De acordo com o website NPM Trends (2023), o React é a biblioteca mais baixada no mundo dentro do último ano (Outubro/2022 – Outubro/2023), comparado com suas principais concorrentes Angular e Vue.

2.4 Métodos de Desenvolvimento Ágeis

No processo de desenvolvimento do sistema Cinemais, adotamos uma abordagem ágil. O norte dos métodos ágeis são os indivíduos e suas comunicações, produto funcional, colaboração com o cliente e resposta rápida a mudanças, aponta Sabbagh (2014).

O desenvolvimento ágil é conhecido por sua flexibilidade e capacidade de se adaptar a mudanças nos requisitos do projeto. Para o contexto do Cinemais, adotamos o Scrum que, segundo Kiniberg (2009), é um método que preza por uma equipe reduzida, gastando um pequeno tempo e construindo uma pequena parte do sistema, porém integrando-as regularmente para que no final do processo se componha por inteiro.

Schwaber e Sutherland (2013) descrevem-na como um framework em que os desenvolvedores lidam e resolvem situações de enorme complexidade através de entregas recorrentes de alto valor. O método é tratado como leve, intuitivo e tem grande dificuldade de ser dominado, ou seja, dificuldade para executar com total conformidade às regras do Scrum.

Na abordagem ágil, Sabbagh (2014) cita que a comunicação contínua e a colaboração das equipes são indispensáveis. A multifunção de todos os integrantes do projeto torna o trabalho cada vez mais coeso e, consequentemente, responde de maneira mais flexível às mudanças dos requisitos do projeto, permitindo uma rápida adaptação a demandas novas.

2.5 Integração Contínua e Entrega Contínua (CI/CD) com Azure DevOps

Para adentrarmos nesse parâmetro, precisamos entender primeiramente o que é DevOps. Segundo Ebert (2016), o termo DevOps representa o desenvolvimento e a implementação cada vez mais rápida e flexível, juntando duas partes da tecnologia da informação que são historicamente distantes. A implementação de práticas ágeis no desenvolvimento de software fez surgir o DevOps. Dessa forma, acabam compartilhando uma linha de pensamento conhecido, o Lean. Esse pensamento tem como principal foco a eliminação de itens dispensáveis ou processos desnecessários, com intuito de torná-los mais ágeis e, consequentemente, evitar alguns desperdício de recursos, completa Ravichandran (2016).

O DevOps possui um ciclo, conhecido como ciclo DevOps. Em seu livro, Bass (2015) destaca sete processos: Planejamento, Desenvolvimento, Verificação, Teste, Implementação, Operação e Monitoramento. Para implementação do DevOps, podemos destacar duas práticas: Integração Contínua e Entrega Contínua.

A Integração Contínua é intitulada como uma prática destinada ao desenvolvimento, que preza pela integração dos trabalhos várias vezes ao dia, estes sendo realizados pelos desenvolvedores de forma rápida e ágil, conforme explicam Fowler e Foemmel (2006). Antes da integração, é efetuado um processo de construção, em meio a testes automatizados, ou seja, acaba facilitando a detecção de erros na integração de forma mais eficiente, completam.

No caso da entrega Entrega Contínua, Chen (2015) explica o papel desse processo, que é solucionar problemas com a entrega dele para produção do cliente. Esses problemas são solucionados devido a sua capacidade da Entrega Contínua fornecer e resultar em um processo de entrega do software mais simplificado, repetível, confiável, previsível e automatizado.

Para gerenciar o código fonte, as tarefas a serem feitas, prazos e testes, Rossberg (2019) fala sobre o código que decidimos utilizar, sendo ele um controlador de projetos que conta com suporte para extensões, integrando com outros serviços populares, e podendo também desenvolver suas próprias extensões: o software Microsoft Azure DevOps. O software é adaptado para trabalhar com inúmeros tipos de serviços, no qual pode-se escolher partes de todos eles. Ele trabalha de maneira formidável com qualquer tipo de aplicação ou framework, desktop ou nuvem, sendo possível utilizar todos em conjunto para garantir uma solução com diversos serviços.

A integração contínua e a entrega contínua são práticas essenciais nos desenvolvimentos de softwares atuais. Elas envolvem a automação de testes, compilação e implantação para garantir que o código esteja em constante teste e que novas funcionalidades sejam entregues de maneira rápida, eficiente e segura.

2.6 Design System

Conforme definido por Kholmatova, o Design System consiste em:

> Um conjunto de padrões interconectados e práticas compartilhadas organizadas de modo coerente para se atingir os propósitos de produtos digitais. Padrões são os elementos que se repetem que nós combinamos para criar uma interface: coisas como fluxo de usuário, interações, botões, campos de texto, ícones, cores, tipografia, micro cópia. Práticas são como nós escolhemos criar, capturar, compartilhar e usar esses padrões, particularmente quando trabalhamos em equipe (Kholmatova, 2017).

Tendo isso em mente, utilizar itens que tenham um padrão definido traz coerência para a interface que foi criada, melhorando a interação do usuário que a utilizar. Esse mesmo processo ocorre durante a produção de novos itens, possibilitando que diferentes

componentes individuais que já existem sejam utilizados para desenvolver novos fluxos completos, facilitando a criação de mudanças confiáveis e longevas.

2.7 Firebase e BaaS

Batschinski (2016) definiu o Firebase como um serviço de computação em nuvem que serve como middleware. O mesmo fornece aos desenvolvedores uma forma para conectar suas aplicações mobile e web a serviços na nuvem a partir de APIs e SDKs, completa o autor. Nada mais é que uma plataforma digital utilizada para simplificar o desenvolvimento de aplicações mobile e web, essa solução é conhecida como Backend As A Service (BaaS).

Não é necessário o desenvolvimento dessa solução manualmente, visto que o BaaS é uma ferramenta que proporciona o backend e a infraestrutura de uma aplicação de maneira fácil e intuitiva, demonstra Andrade (2020) em seu artigo. Ele oferece armazenamento, autenticação, escalabilidade, notificação e muitas outras funcionalidades.

Alguns dos diversos recursos do Firebase (2023) são: realtime database, google analytics, cloud firestore, authentication e cloud storage Firebase. Para nosso projeto utilizamos o Authentication, que tem o papel de cumprir a autenticação segura e simplificada de login para o usuário, seja esse login feito desde senhas, até login por redes sociais, como o facebook e o X ou plataformas de trabalho, como o github e o gmail.

2.8 Visual Studio Code

O Visual Studio Code (ou VSCode) é um editor de código criado pela Microsoft (2023), que segundo a mesma, teve o objetivo de apresentar de forma mais simples e prático o ambiente de desenvolvimento semelhante a Integrated Development Environment (IDE) Visual Studio. Conforme aponta Rask (2021), o foco do VSCode é o desenvolvimento de aplicações web usando javascript, typescript e Node.js. Além do mais, tem presente uma grande biblioteca de extensões criadas pela própria comunidade, permitindo que o editor seja usado para desenvolvimento de aplicações com as mais diversas linguagens.

Rask (2021) também descreve as características do VSCode, sendo eles um núcleo desenvolvido em Node.js e uma UI desenvolvida em HTML e CSS, utilizando o Framework Electron, para fazer build do seu executável para os sistemas operacionais Windows, Linux e Mac. Além disso, a Microsoft disponibilizou um código fonte aberto, com um mecanismo de integração de código-fonte externo via extensões, que usam um ecossistema Node.js.

3. Materiais e Métodos

A seleção dos materiais e métodos adotados no projeto foram escolhidos com foco na otimização do projeto em todas as partes do ciclo de vida de um software, desde sua concepção até a publicação para uso.

3.1 Materiais

Com base na bibliografia consultada e estudada para o embasamento da construção do projeto, foram escolhidas para aplicação as tecnologias que trazem a maior vantagem de experiência para o desenvolvimento, aproveitando o conhecimento já obtido para ser aplicado em diferentes plataformas, bem como a possibilidade de manter o sistema desenvolvido coerente e seguro.

3.1.1 Linguagens, Extensões e Frameworks

No desenvolvimento do portal Cinemais+, foram empregadas linguagens, extensões e frameworks fundamentais para a construção da plataforma. O backend do sistema foi

desenvolvido em Node.js, aproveitando a flexibilidade e escalabilidade oferecidas por essa tecnologia. O Node.js foi utilizado para executar a lógica de negócios, gerenciar o acesso ao banco de dados, implementar autenticação e outras funcionalidades essenciais.

Além disso, para aprimorar a segurança e a experiência do desenvolvedor, o TypeScript foi adotado tanto no backend quanto no frontend. O TypeScript proporcionou benefícios significativos para a qualidade e a manutenção do código, tornando o sistema mais seguro e escalável. A capacidade de utilizar recursos de tipagem estática e programação orientada a objetos foi crucial para criar um sistema robusto e confiável.

3.1.2 Frameworks Específicos

No frontend, o React foi escolhido para criar a interface do sistema, proporcionando uma experiência de usuário envolvente e responsiva. Por meio da componentização e da capacidade de reagir às interações do usuário, o React desempenhou um papel fundamental na criação de páginas web dinâmicas, facilitando a navegação, a seleção de filmes e a compra de ingressos para os clientes do Cinemais.

Além disso, foi utilizada a biblioteca de modelos ShadCN UI, baseada em Radix UI e Tailwind CSS. Essa biblioteca ofereceu um conjunto de componentes React prontos para uso, totalmente acessíveis e personalizáveis. A escolha do ShadCN UI foi guiada pela sua alta qualidade, praticidade no acesso à biblioteca e capacidade de personalização dos componentes, contribuindo significativamente para o desenvolvimento do Cinemais.

Essas tecnologias e ferramentas desempenharam papéis cruciais no desenvolvimento do Cinemais+, garantindo a eficiência, segurança e qualidade do sistema.

3.2 Métodos

Como se trata de um projeto prático, o presente trabalho se trata de uma pesquisa com natureza aplicada, que, segundo Prodanov e Freitas (2013), é utilizada quando o objetivo é construir materiais e conhecimentos para que sejam aplicados e possibilitando a solução de problemas particulares.

Em relação ao aspecto técnico presente no projeto, que são as ferramentas para que seja possível implementar o conceito exposto no presente trabalho, foi utilizada uma abordagem de pesquisa bibliográfica “elaborada a partir de material já publicado, constituído principalmente de: livros, revistas, publicações em periódicos e artigos científicos, jornais, boletins, monografias, dissertações, teses, material cartográfico, internet” (PRODANOV; FREITAS, 2013, p.54), possibilitando o embasamento das escolhas feitas e a expansão do conhecimento em tópicos não conhecidos.

3.2.1 Modelagem da Aplicação

De acordo com Pressman (2011), a modelagem de sofware refere-se ao processo de construir modelos abstratos que fazem a representação do software que será desenvolvido, abrangendo diferentes aspectos do sistema, como sua estrutura estática, o comportamento que ele terá, a interface proposta para o usuário e outros aspectos que sejam relevantes.

Ele expressa que esses modelos precisam cumprir determinadas tarefas:

- Representar o fluxo do sistema
- Apresentar sua arquitetura
- Determinar as funções existentes
- Expressar características desejadas pelos usuários

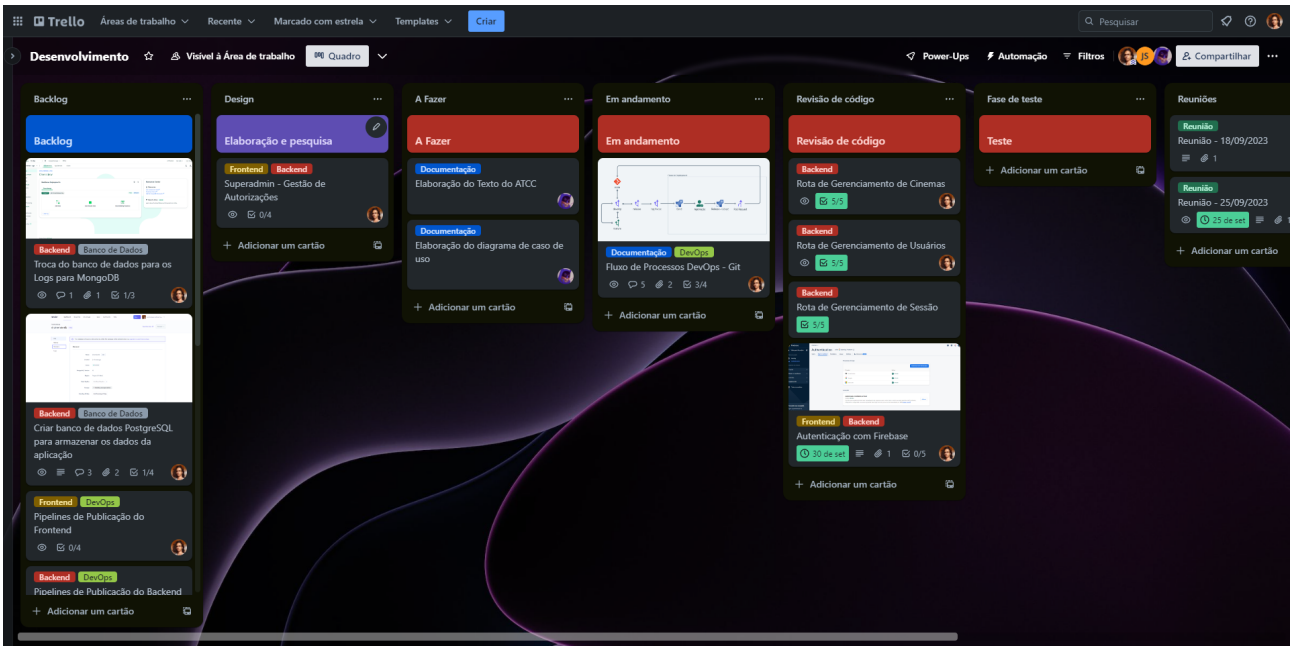
De forma resumida, é possível entender que a modelagem consiste no desenvolvimento de modelos para consolidar a compreensão do que será de fato construído, elaborando-se um esboço do software que representa a visão macro.

3.2.2 Quadro com Itens de Trabalho

Segundo Anderson (2010) o método Kanban apresenta em seu método a utilização de um quadro para a definição das atividades necessárias e a verificação dos estados atuais dos itens, sendo sua principal ferramenta para o acompanhamento do processo de trabalho desenvolvido.

Com base nos itens levantados durante a modelagem da aplicação, foram definidos os itens necessários para o desenvolvimento do projeto, criando um quadro Kanban (demonstrado na Figura 1) para facilitar a visualização das tarefas e o progresso individualizado em cada uma.

Figura 1 – Quadro kanban com os itens de trabalho



Fonte: Elaborado pelo autor

3.2.3 Gerenciamento de Configuração

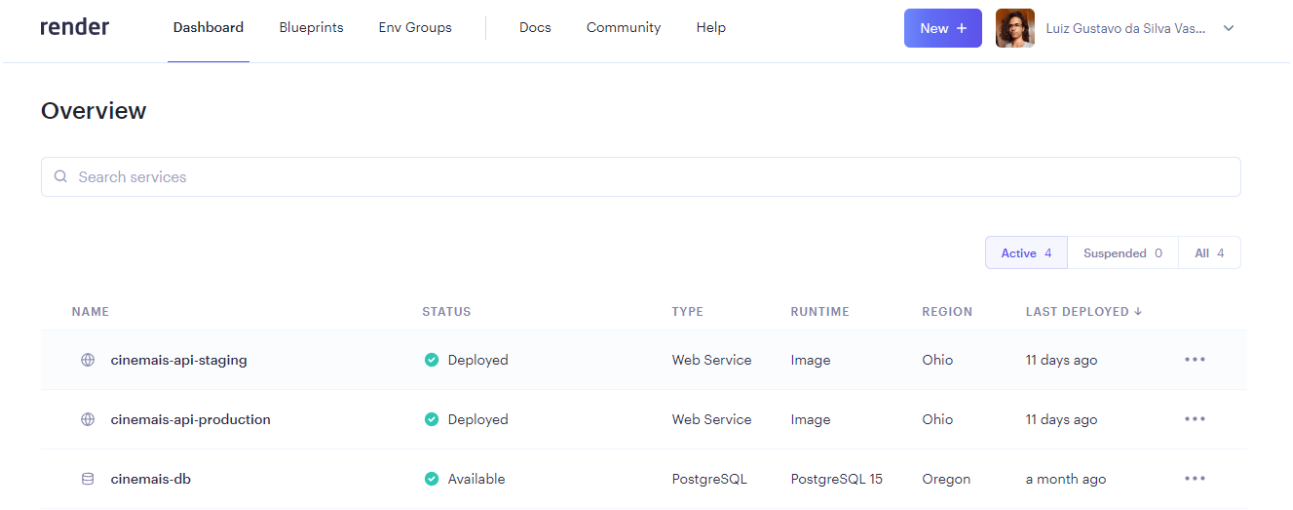
O gerenciamento de configuração forneceu um papel essencial na manutenção da integridade do sistema proposto. Utilizamos a ferramenta GIT para ter um controle de versão, visando mapear todas as alterações no código-fonte. De acordo com Bitbucket (2022) o GIT é intitulado como um DVCS (Sistema de Controle de Versão Distribuído), ao contrário dos antigos sistemas de controle de versão como SVN (conhecido também como Subversion), onde apenas um local armazenava o histórico de mudança do software. No caso do GIT, as máquinas dos programadores envolvidos no mesmo projeto, contém o histórico de alterações do software, incluindo todo o código fonte do projeto, servindo assim cada cópia como um repositório em si.

Tendo isso em vista, foram implementadas práticas de gerenciamento de configuração para documentar os requisitos do sistema, permitindo uma fácil recuperação e restauração em caso de possíveis falhas, garantindo a consistência do ambiente de validação e produção.

3.2.4 Processo de Publicação

Foram utilizadas ferramentas e serviços em nuvem para garantir a disponibilidade e acesso ao sistema projetado, iniciando pelo processo de criação dos recursos de banco de dados e para a API na plataforma “Render”, que tem seu dashboard exibido na Figura 2.

Figura 2 – Dashboard da plataforma “Render” com recursos criados



Fonte: Elaborado pelo autor

Para contemplar a realização dos testes para cada um dos requisitos implementados no projeto sem afetar a experiência do usuário que está utilizando o sistema, foram definidos dois recursos para a API com diferentes finalidades:

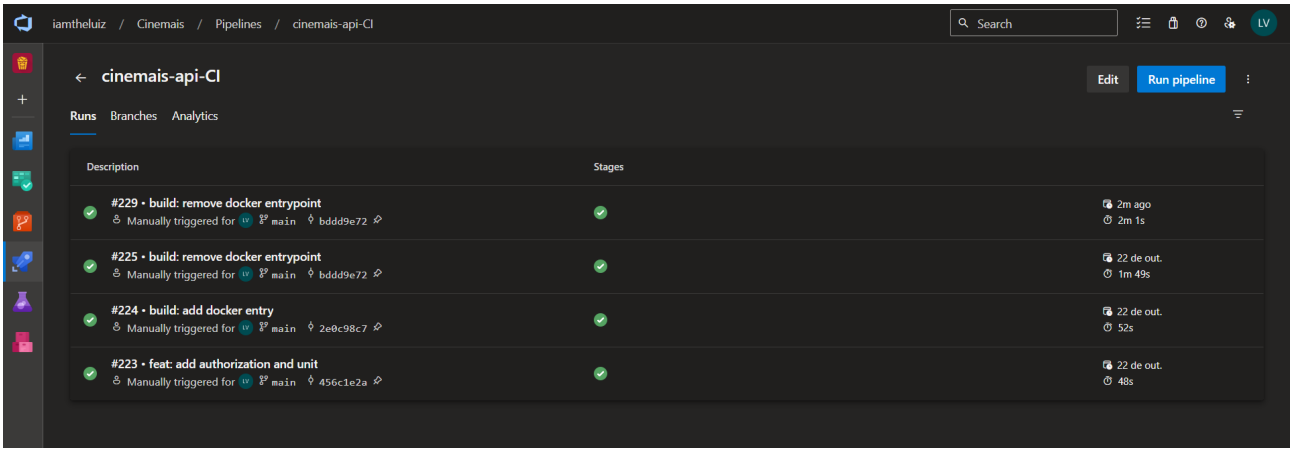
- **cinemais-api-staging**: Representa a API utilizada para os testes e validação do sistema, não sendo o ambiente para uso pelo usuário final do projeto
- **cinemais-api-production**: Define a API utilizada pelo usuário final, refletindo a versão estável do sistema, que foi validada para uso

Essa separação possibilita a realização de testes fidedignos ao ambiente final de publicação, garantindo que problemas ou falhas não diagnosticadas localmente, durante o desenvolvimento, sejam encontradas no ambiente intermediário.

Com os recursos necessários para a publicação do backend do projeto, foi criado no Azure DevOps um processo de esteiras automatizadas que faz a compilação da API e também a sua publicação nos ambientes provisionados.

Assim sendo, foram criadas duas esteiras para a publicação do backend do projeto, a primeira faz a construção da aplicação, sendo denominada “pipeline de build” (Figura 3).

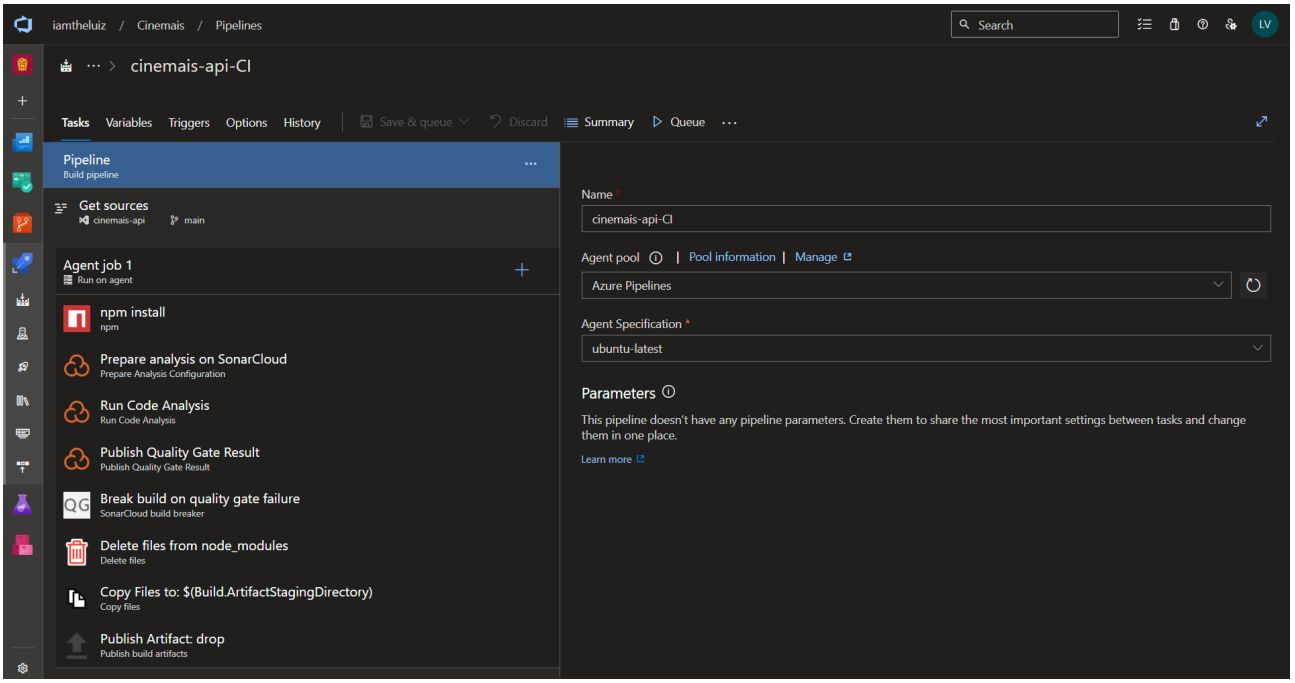
Figura 3 – Pipeline de build “cinemais-api-CI”



Fonte: Elaborado pelo autor

Esse pipeline utiliza um roteiro de tarefas que foi definido conforme as necessidades da aplicação para efetuar a instalação das dependências necessárias para o projeto, fazer a análise de segurança do código implementado e a criação de um arquivo com todo o conteúdo do projeto preparado para publicação, também chamado de “artefato” (Figura 4).

Figura 4 – Tarefas presentes no pipeline “cinemais-api-CI”



Fonte: Elaborado pelo autor

O processo definido consiste no uso das seguintes tarefas:

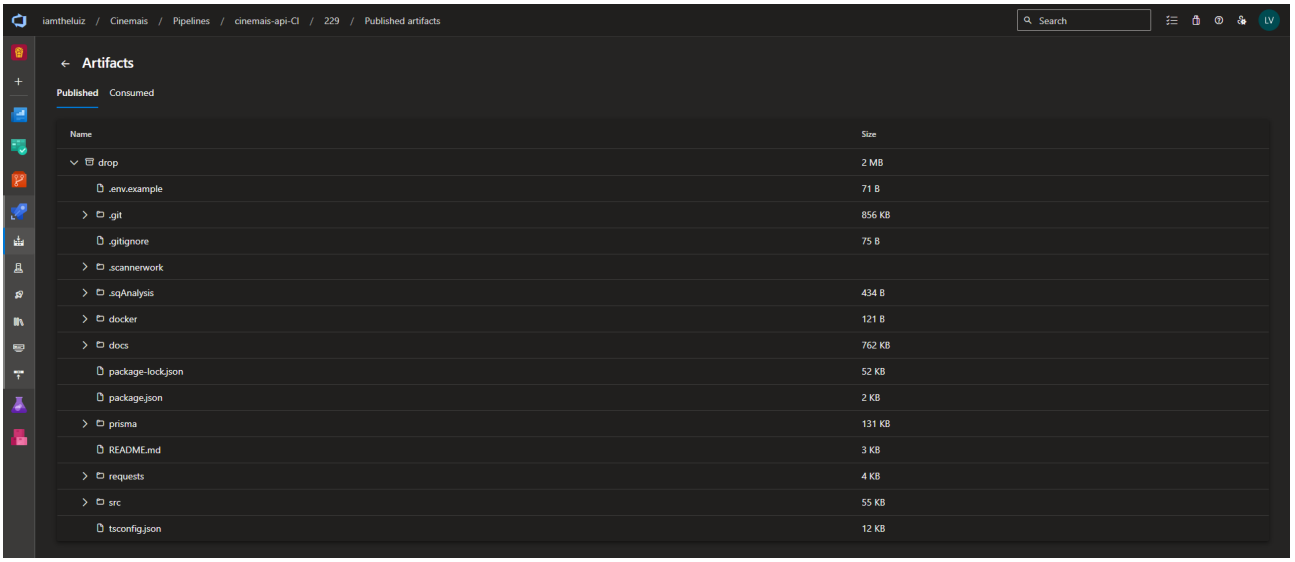
- **npm install**: Esta tarefa executa o comando `npm install`, que é usado para instalar todas as dependências do projeto Node.js definidas no arquivo `package.json`.
    - É uma etapa essencial para garantir que todas as dependências do projeto sejam baixadas e instaladas corretamente antes de prosseguir com outras etapas de compilação e análise.

- **Prepare analysis on SonarCloud**: Esta tarefa prepara os dados e as configurações necessárias para a análise do código-fonte no SonarCloud, um serviço de análise estática de código que “pode ajudar a detectar e erradicar bugs de segurança comuns” (GOMES et al., 2009, p. 15).
    - Antes de executar a análise do código, é necessário preparar o ambiente e configurar as regras de análise estática, o que é feito nesta etapa.
- **Run Code Analysis**: Esta tarefa executa a análise do código-fonte para identificar problemas, padrões de código e outras métricas de qualidade.
    - A análise estática do código ajuda a identificar possíveis bugs, vulnerabilidades e problemas de estilo, permitindo melhorias na qualidade do código.
- **Publish Quality Gate Result**: Esta tarefa publica os resultados da análise de qualidade do código no SonarCloud.
    - Após a análise do código, os resultados são enviados para o SonarCloud, onde são processados e exibidos para que os desenvolvedores possam revisar as métricas de qualidade e tomar medidas para melhorar o código, se necessário.\
- **Break build on quality gate failure**: Esta tarefa interrompe a construção (build) se os resultados da análise de qualidade no SonarCloud não atenderem aos critérios definidos (por exemplo, se o código não atender a determinadas métricas de qualidade).
    - Garante que apenas código de alta qualidade seja promovido para os próximos estágios do pipeline, ajudando a manter um alto padrão de qualidade no projeto.
- **Delete files from node_modules**: Esta tarefa exclui os arquivos e pastas no diretório `node_modules`, que contém as dependências do Node.js instaladas anteriormente.
    - É útil para economizar espaço em disco e evitar possíveis conflitos de dependência, especialmente se você estiver criando um pacote ou artefato que não requer as dependências do projeto.
- **Copy Files to: $(Build.ArtifactStagingDirectory)**: Esta tarefa copia arquivos e pastas do projeto para o diretório de artefatos de compilação (`$(Build.ArtifactStagingDirectory)`), que é usado para armazenar os artefatos gerados durante a construção.
    - Prepara os artefatos compilados e outras informações necessárias para os estágios subsequentes do pipeline, facilitando a implantação e distribuição do software.

Essas tarefas juntas formam um pipeline de build robusto e eficiente, garantindo que o código seja compilado, analisado e avaliado quanto à qualidade antes de ser implantado ou distribuído.

Quando a execução do pipeline é concluída, é gerado um artefato com o conteúdo da aplicação (Figura 5):

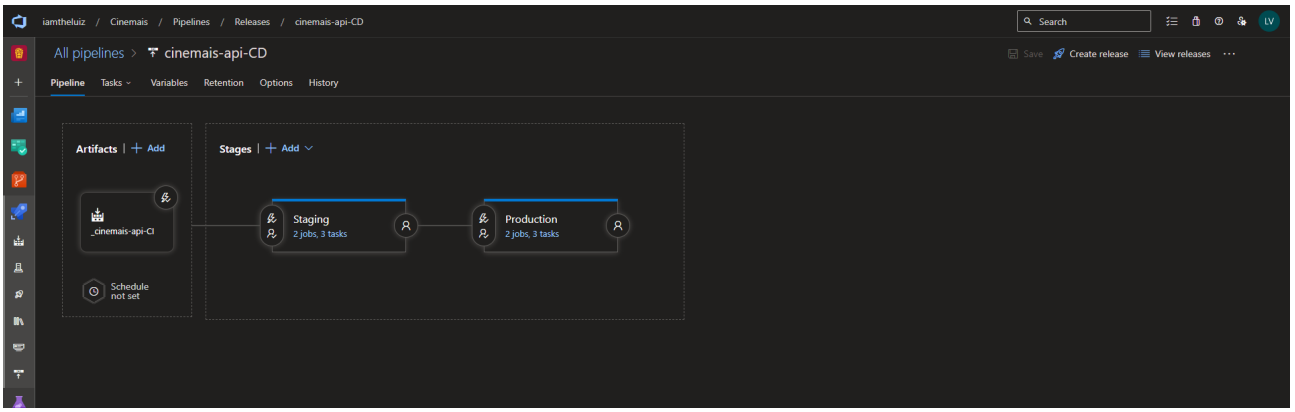
Figura 5 – Artefato gerado pelo pipeline “cinemais-api-CI”



Fonte: Elaborado pelo autor

Esse artefato é utilizado por uma esteira de publicação do projeto, que recebe o nome de “pipeline de release” (Figura 6):

Figura 6 – Pipeline de release “cinemais-api-CD”



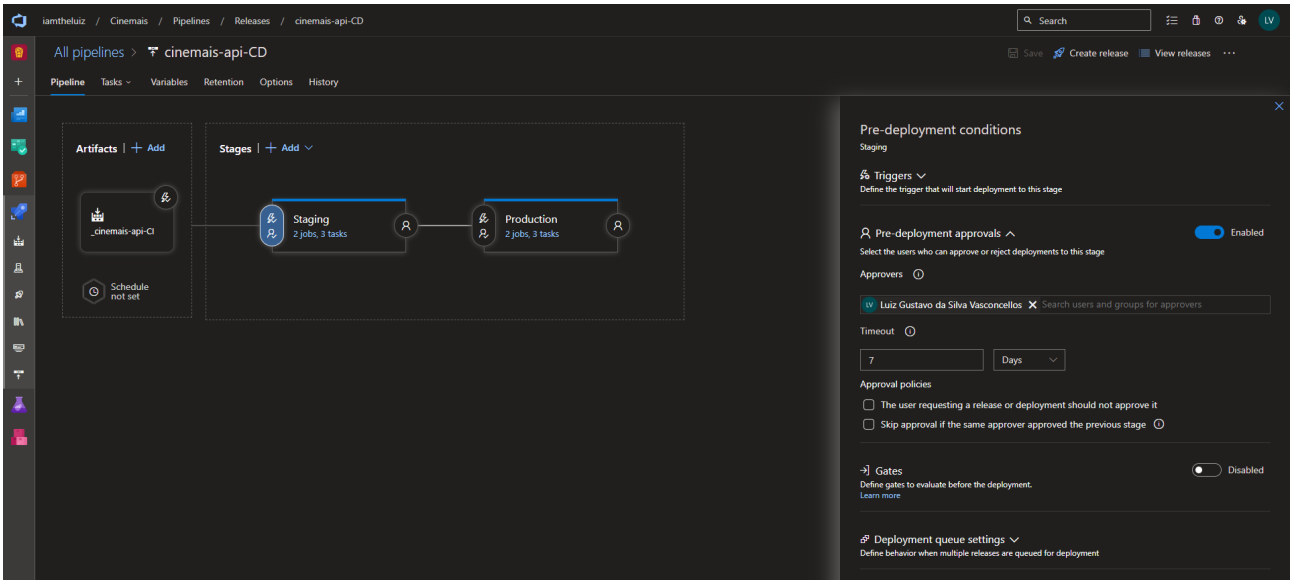
Fonte: Elaborado pelo autor

O pipeline se divide nos dois ambientes em que a API pode ser publicada:

- **Staging**: Ambiente de testes e validação do projeto, sendo uma réplica do ambiente final
- **Production**: Ambiente final para o projeto, que é apresentado ao usuário final

Ao ser feita a execução da esteira, existe uma configuração de aprovação que deve ser feita antes de ser liberada a atualização de cada um dos ambientes, garantindo que ela só será executada no momento correto (Figura 7):

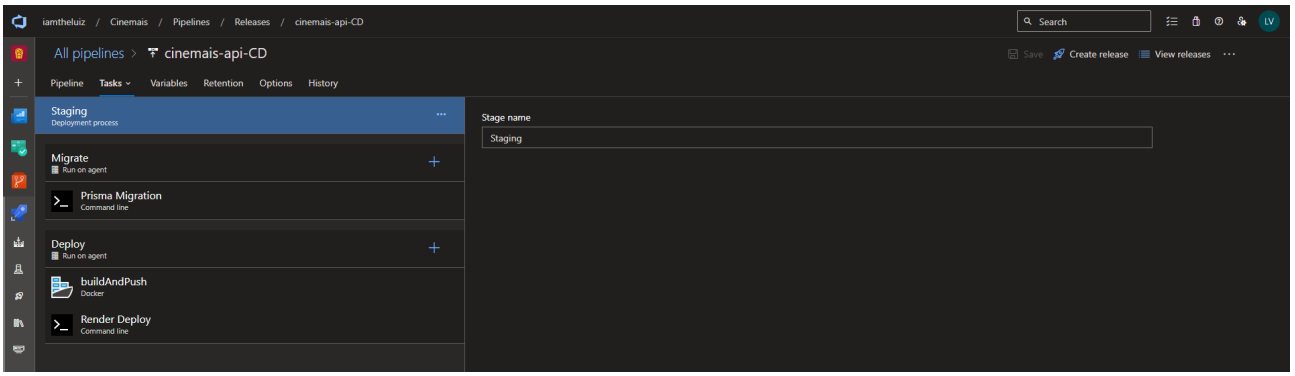
Figura 7 – Aprovação para publicação no pipeline “cinemais-api-CD”



Fonte: Elaborado pelo autor

Dessa forma, após ser aprovada a publicação, são executadas as tarefas do pipeline (Figura 8):

Figura 8 – Tarefas presentes no pipeline “cinemais-api-CD”



Fonte: Elaborado pelo autor

Dessa forma, após ser aprovada a publicação, são executadas as tarefas do pipeline:

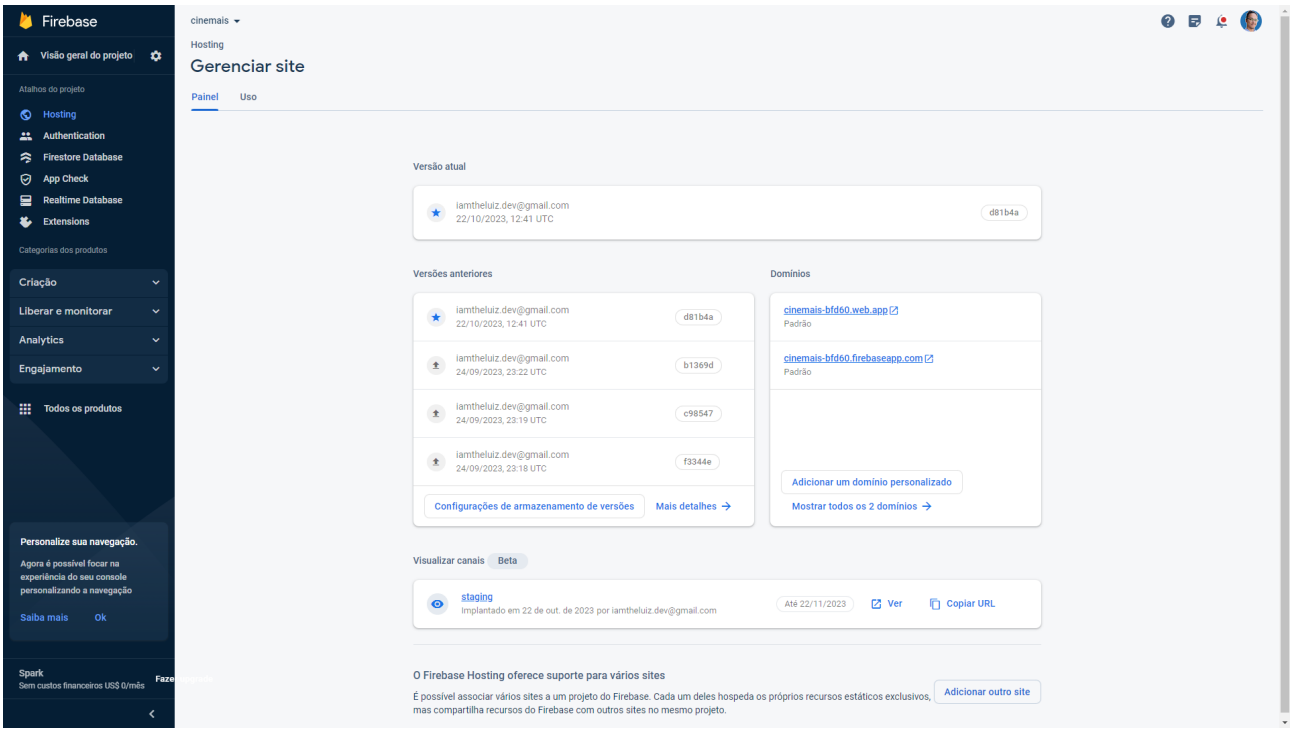
- **Prisma Migration**: A função desta tarefa é executar migrações do banco de dados usando o Prisma ORM. Isso inclui gerar migrações, aplicar essas migrações ao banco de dados e semear o banco de dados com dados iniciais, se necessário.
    - Migrações de banco de dados são cruciais para manter a consistência do banco de dados ao longo das versões do aplicativo. Garante que o esquema do banco de dados esteja sempre alinhado com a estrutura esperada pela aplicação.
- **buildAndPush**: Esta tarefa é responsável por construir uma imagem Docker do aplicativo e enviá-la para um registro de contêiner. Ela utiliza variáveis de ambiente para configurar o ambiente da aplicação durante a construção.
    - Construir e empurrar a imagem Docker é uma etapa crítica para a implantação do aplicativo em um ambiente de produção. A imagem Docker contém todos os componentes necessários para executar o aplicativo, garantindo consistência entre diferentes ambientes.

- **Render Deploy**: Esta tarefa executa um script usando wget para chamar um gancho de implantação (deploy hook) em um URL específico. Isso pode ser usado para notificar outros sistemas ou realizar ações específicas após a implantação bem-sucedida.
    - O gancho de implantação é útil para realizar ações adicionais, como atualizar caches, notificar equipes ou sistemas externos sobre a nova versão do aplicativo, ou realizar outras operações personalizadas após a implantação.

Essas tarefas juntas formam uma pipeline de release robusta, automatizando não apenas a implantação do aplicativo, mas também garantindo a consistência do banco de dados durante o processo de liberação.

De modo similar ao backend, o frontend possui recursos publicados em serviços disponibilizados pelo Firebase para a publicação de sites estáticos, utilizando apenas recursos como HTML, CSS e JavaScript (Figura 9).

Figura 9 – Tela de gerenciamento de hospedagem do Firebase

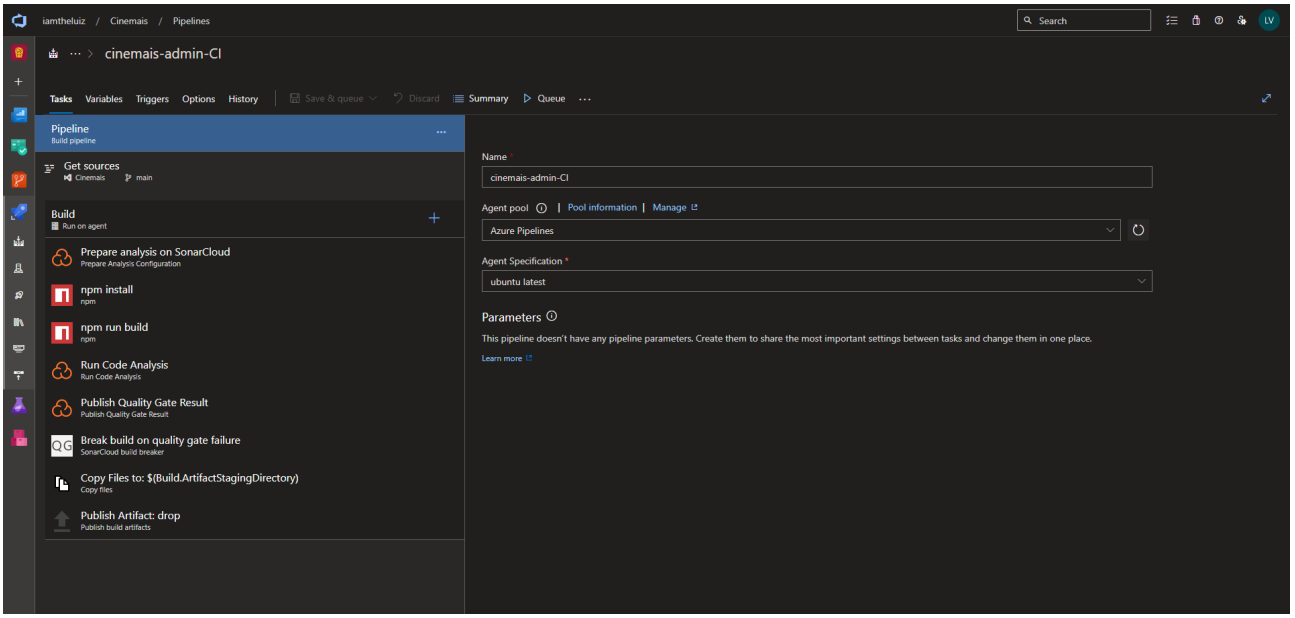


Fonte: Elaborado pelo autor

Utilizando essa ferramenta é possível publicar um site com React, já que seu conteúdo para publicação necessita apenas de recursos estáticos.

Após a criação dos recursos no Firebase, também foram estruturados os pipelines de build e release para o frontend do projeto (Figura 10):

Figura 10 – Tarefas presentes no pipeline “cinemais-admin-CI”



Fonte: Elaborado pelo autor

O processo definido consiste no uso das seguintes tarefas:

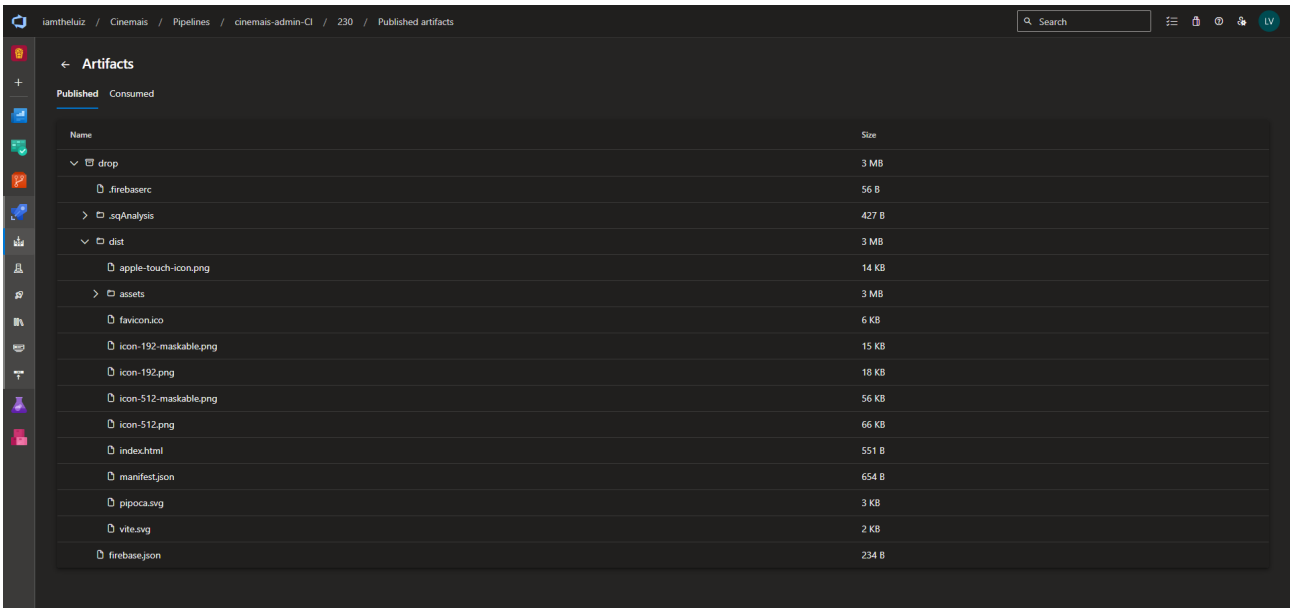
- **Prepare analysis on SonarCloud**: Esta tarefa prepara os resultados da análise estática de código para envio ao SonarCloud, uma plataforma que oferece insights sobre a qualidade do código.
    - Permite análise estática do código-fonte para identificar possíveis problemas, vulnerabilidades e padrões de codificação.
- **npm install**: Instala as dependências do projeto Node.js usando o npm (Node Package Manager).
    - Garante que todas as dependências necessárias para o projeto estejam instaladas e prontas para serem usadas durante a compilação.
- **npm run build**: Executa o script de build definido no arquivo package.json do projeto React.
    - Compila o código-fonte React em arquivos estáticos (HTML, CSS, JavaScript) prontos para serem implantados em um servidor web.
- **Run Code Analysis**: Executa a análise de código para identificar problemas e padrões de codificação usando ferramentas específicas.
    - Garante que o código atenda aos padrões de codificação da equipe, melhorando a consistência e a qualidade do código.
- **Publish Quality Gate Result**: Publica o resultado da análise de qualidade do código, como métricas de cobertura de código e conformidade com padrões de codificação, em um local acessível para a equipe.
    - Fornece transparência sobre a qualidade do código e permite que a equipe tome medidas para melhorar, se necessário.
- **Break build on quality gate failure**: Interrompe a compilação se a análise de qualidade do código não atender aos critérios definidos.
    - Garante que apenas código de alta qualidade seja incorporado ao projeto, ajudando a evitar a introdução de problemas no código-base.
- **Copy Files to: $(Build.ArtifactStagingDirectory)**: Copia os arquivos compilados e outros artefatos importantes para o diretório de preparação para artefatos na máquina de compilação.

    - Prepara os arquivos de build e outros artefatos para serem empacotados como um pacote de artefato.
- **Publish Artifact: drop**: Publica os artefatos (arquivos de build, imagens, etc.) para o Azure DevOps, tornando-os disponíveis para implantação.
    - Disponibiliza os artefatos de build para ambientes de teste ou produção, permitindo a implantação do aplicativo.

Considerando o conjunto de tarefas utilizadas para fazer a construção da aplicação, o pipeline de build é um ponto chave para a verificação da qualidade, confiabilidade e eficiência no desenvolvimento da aplicação, automatizando o processo de compilação do código, análise e implementação.

Quando a execução da esteira é concluída, é gerado um artefato com o conteúdo da aplicação (Figura 11):

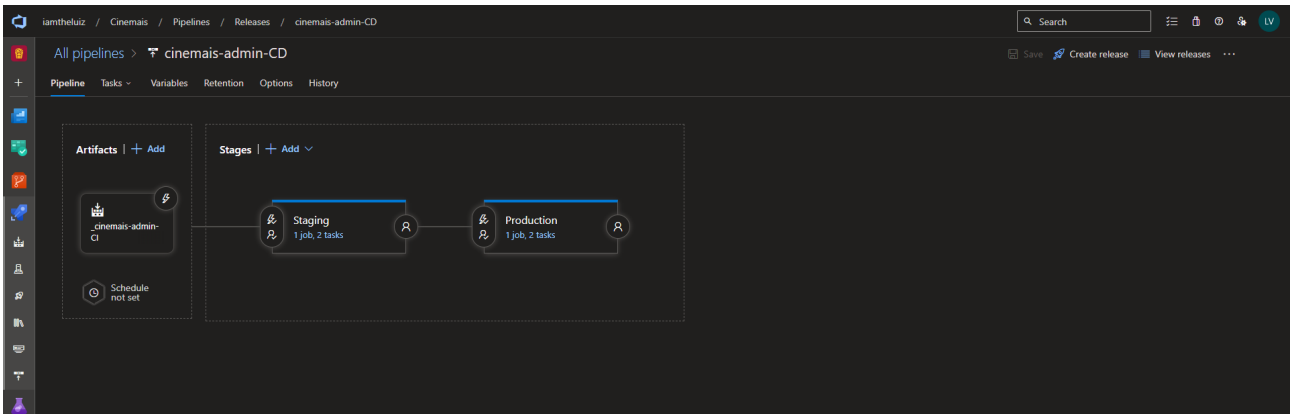
Figura 11 – Artefato gerado pelo pipeline “cinemais-admin-CI”



Fonte: Elaborado pelo autor

Esse artefato é utilizado pelo pipeline de release para o Frontend (Figura 12):

Figura 12 – Pipeline de release “cinemais-admin-CD”



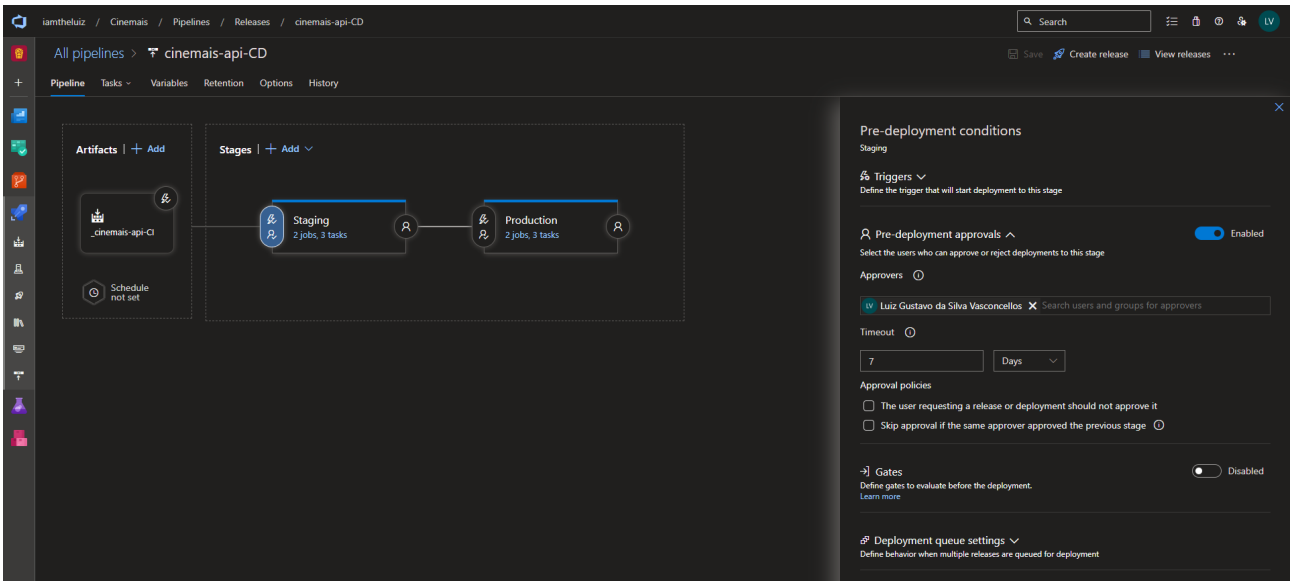
Fonte: Elaborado pelo autor

O pipeline se divide nos dois ambientes em que o Frontend pode ser publicado (da mesma forma que é feito com o Backend):

- **Staging**: Ambiente de testes e validação do projeto, sendo uma réplica do ambiente final
- **Production**: Ambiente final para o projeto, que é apresentado ao usuário final

Com o objetivo de garantir que as publicações sejam feitas no momento correto, existe a configuração de aprovação de uma nova atualização do ambiente (Figura 13):

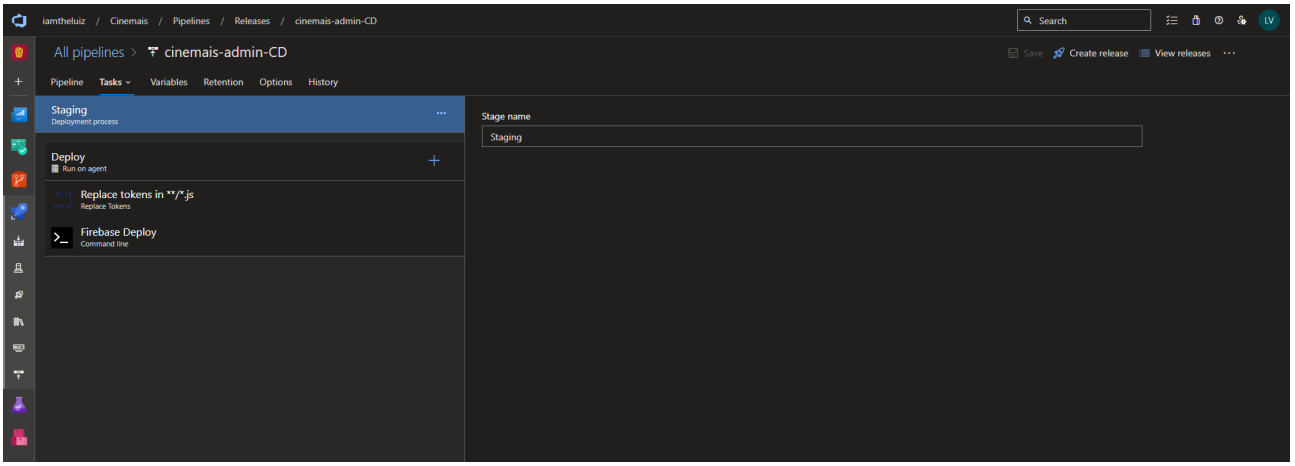
Figura 13 – Aprovação para publicação no pipeline “cinemais-admin-CD”



Fonte: Elaborado pelo autor

Dessa forma, após ser aprovada a publicação, são executadas as tarefas do pipeline (Figura 14):

Figura 14 – Tarefas presentes no pipeline “cinemais-admin-CD”



Fonte: Elaborado pelo autor

Dessa forma, após ser aprovada a publicação, são executadas as tarefas do pipeline:

- **Replace tokens in /*.js**: Esta tarefa substitui tokens específicos (variáveis) nos arquivos JavaScript (*.js) por valores correspondentes. Geralmente, isso é usado para substituir variáveis de ambiente ou configurações específicas de ambiente durante o processo de build.

    - Garante que as configurações, como URLs de API ou chaves de acesso, sejam corretas para o ambiente de produção, facilitando a configuração dinâmica da aplicação em diferentes ambientes.
- **Firebase Deploy**: Esta tarefa implanta a aplicação no Firebase Hosting, que é um serviço de hospedagem para aplicativos da web oferecido pelo Firebase, uma plataforma do Google.
    - Facilita a implantação rápida e fácil da aplicação React em um ambiente de produção. O Firebase Hosting oferece escalabilidade, segurança e confiabilidade, sendo uma escolha popular para hospedar aplicativos da web estáticos e dinâmicos.

Essas tarefas juntas garantem que a aplicação React seja construída corretamente, que as configurações sejam ajustadas para o ambiente de produção e que a aplicação seja implantada de forma eficaz e confiável para os usuários finais.

4. Resultados e Discussão

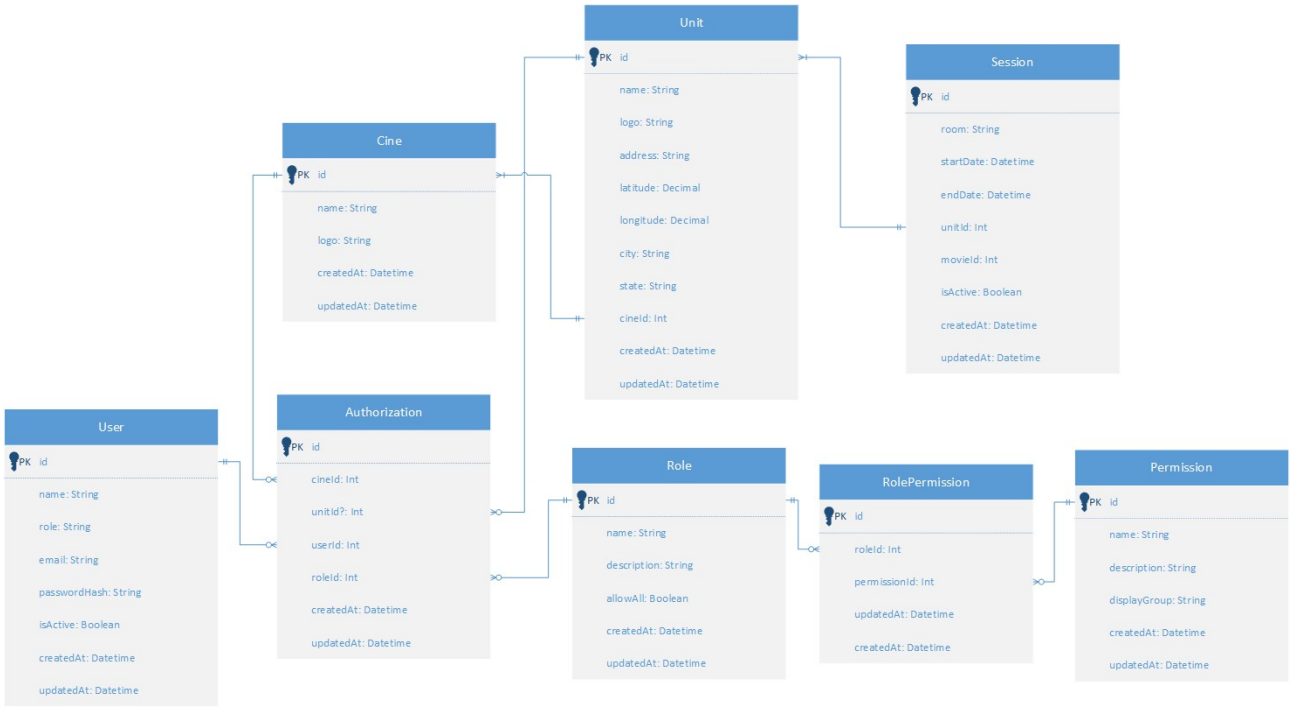
A presente sessão tem como objetivo demonstrar o material desenvolvido como resultado da metodologia aplicada e sua relevância.

4.1 Modelagem de Dados

O primeiro passo realizado para a construção do projeto foi a definição da modelagem de dados utilizada pelo sistema, garantindo uma previsibilidade dos recursos necessários para a cobertura dos requisitos necessários para uma primeira versão do sistema.

Dentre os artefatos gerados no processo de modelagem de dados, foi desenvolvido o Modelo Entidade Relacionamento (MER) para o banco de dados do projeto, relacionando as entidades, seus atributos e relacionamentos (Figura 15).

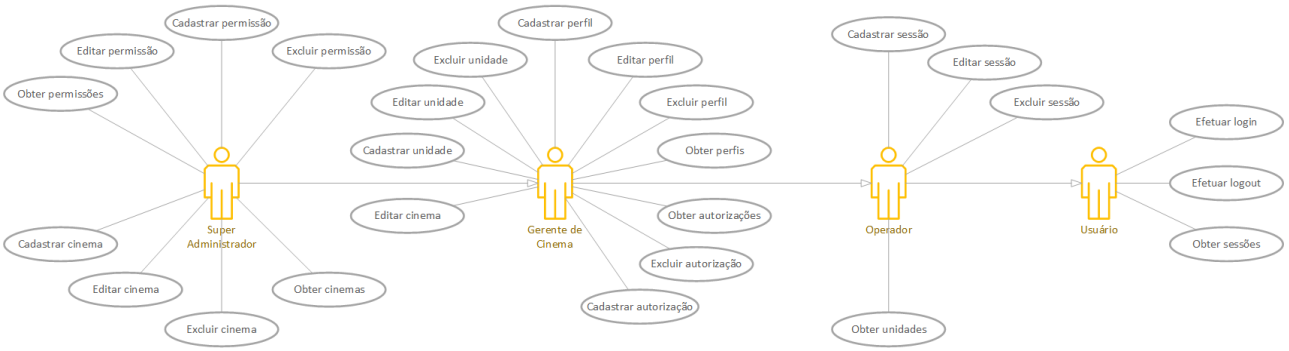
Figura 15 – Modelo entidade relacionamento do projeto



Fonte: Elaborado pelo autor

Em conjunto com MER, também foi gerado um diagrama de caso de uso com a demonstração dos principais atores do sistema e seus papéis (Figura 16).

Figura 16 – Diagrama de caso de uso do projeto



Fonte: Elaborado pelo autor

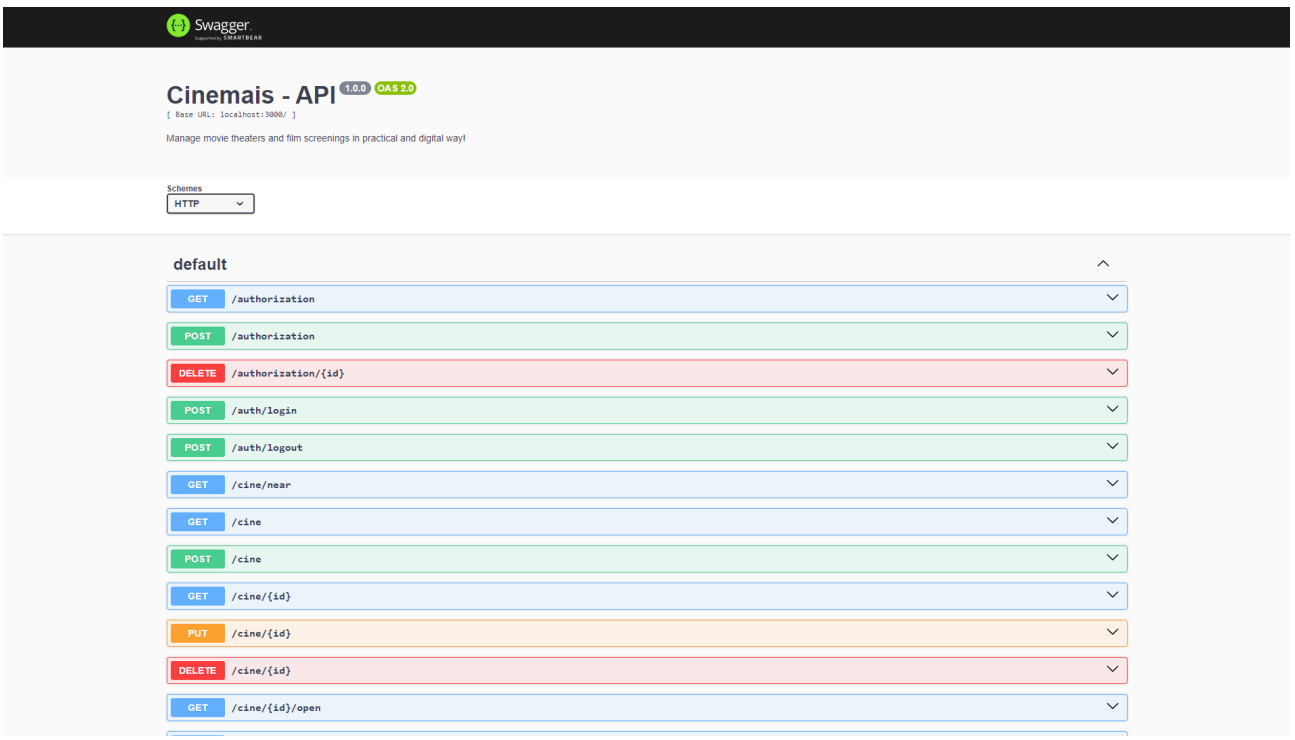
É importante reforçar que o diagrama apresenta atores que representam um cenário genérico para levantar as ações necessárias para o sistema, mas a proposta é possibilitar a criação de diferentes perfis que podem ter permissões personalizadas conforme a necessidade dos administradores dos cinemas, respeitando o limite das permissões de gerenciamento do sistema.

4.2 Definição da API

A API do Cinemais foi desenvolvida para fornecer funcionalidades essenciais, possibilitando que sejam feitas expansões nos serviços que utilizam seus recursos e regras. As rotas foram cuidadosamente criadas para garantir uma integração suave com o frontend da aplicação, proporcionando uma interface eficaz para a interação dos usuários.

A figura abaixo apresenta a documentação da API gerada pelo Swagger, destacando as diferentes rotas disponíveis, os parâmetros aceitos e as respostas correspondentes. O Swagger oferece uma visão clara e interativa das funcionalidades da API, facilitando tanto o desenvolvimento quanto o teste das operações (Figura 17).

Figura 17 – Rotas do backend do projeto



Fonte: Elaborado pelo autor

As rotas implementadas foram:

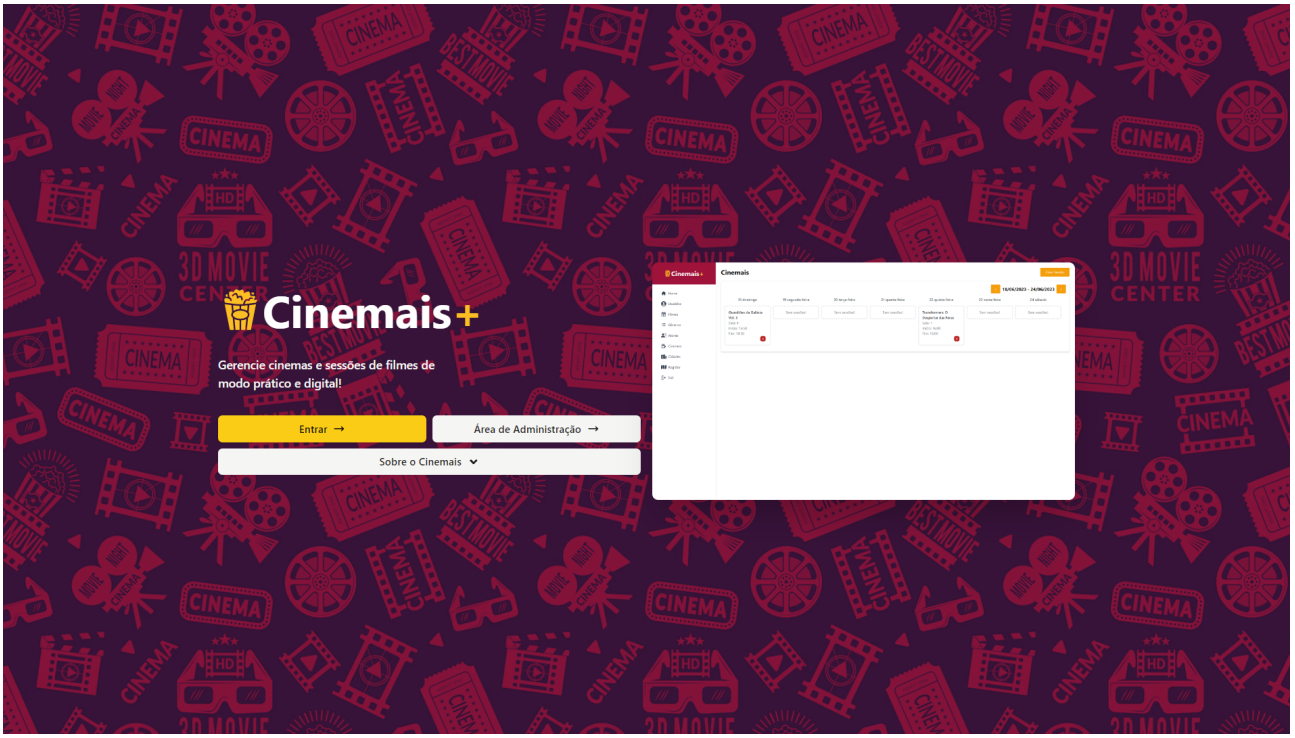
- **Authorization:** /authorization
- **Authorization (por ID)**: /authorization/{id}
- **Login de Autenticação**: /auth/login
- **Logout de Autenticação**: /auth/logout
- **Cinemas Próximos**: /cine/near
- **Cinemas**: /cine
- **Cinemas (por ID)**: /cine/{id}
- **Abrir Cinema (por ID)**: /cine/{id}/open
- **Painel Inicial**: /home/dashboard
- **Painel Administrativo**: /home/admin/dashboard
- **Logs**: /log
- **Logs (por ID)**: /log/{id}
- **Filmes**: /movie
- **Filmes (por ID)**: /movie/{id}
- **Permissões:** /permission
- **Permissões (por ID)**: /permission/{id}
- **Funções**: /role
- **Funções (por ID)**: /role/{id}
- **Sessões**: /session
- **Sessões (por ID)**: /session/{id}
- **Unidades Próximas**: /unit/near
- **Unidades**: /unit
- **Unidades (por ID)**: /unit/{id}
- **Usuários**: /user
- **Usuários (por ID)**: /user/{id}

Com a implementação do backend é possível fazer o desenvolvimento de diversos clientes que fazem o consumo dos recursos oferecidos por ela, apenas respeitando as regras e dados que devem ser recebidos para seu uso.

4.3 Interface

Com a modelagem de dados e também a API com as principais rotas definidas, foram desenvolvidas as telas para uso do sistema pelos administradores do sistema e dos cinemas comportados pela plataforma (Figura 18).

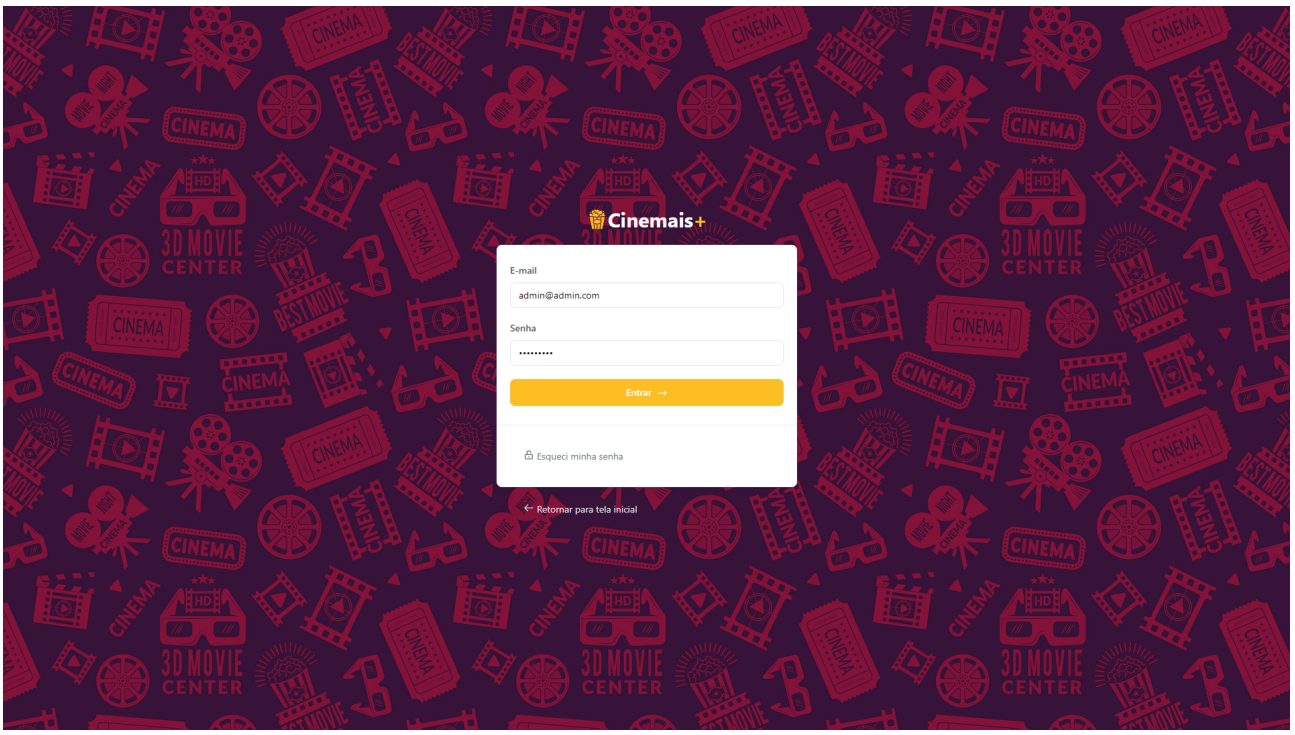
Figura 18 – Tela inicial do projeto “Cinemais”



Fonte: Elaborado pelo autor

Ao acessar a seção “Área de Administração” são solicitadas as credenciais de acesso do usuário (Figura 19).

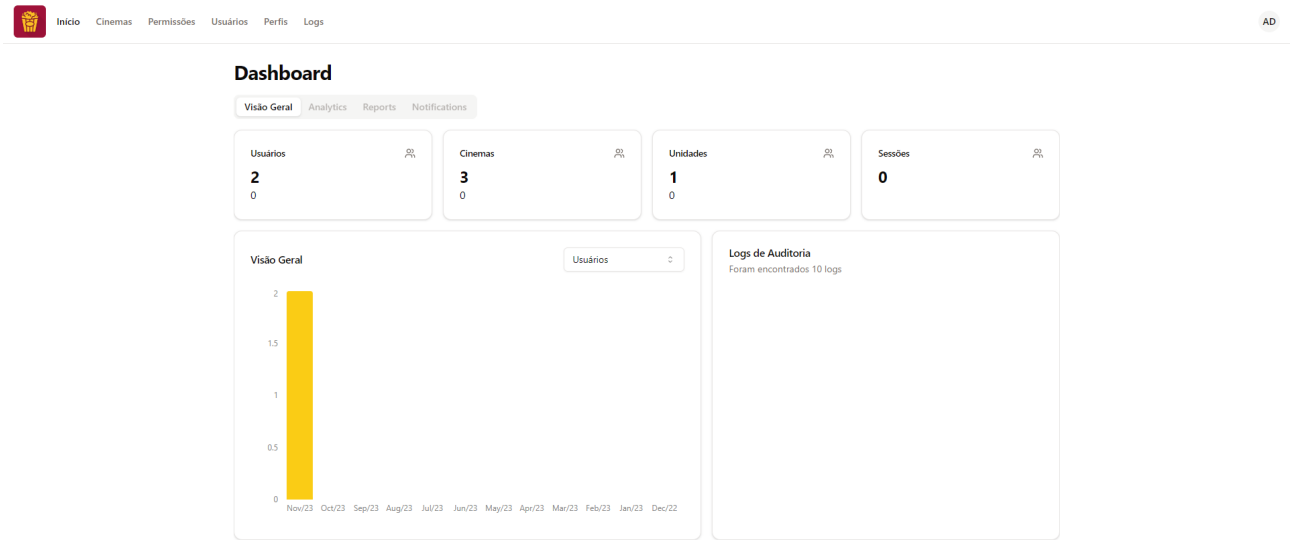
Figura 19 – Tela de login na área administrativa



Fonte: Elaborado pelo autor

Caso o usuário seja válido e tenha autorização para visualizar a área administrativa, sua tela será redirecionada para o dashboard do sistema (Figura 20).

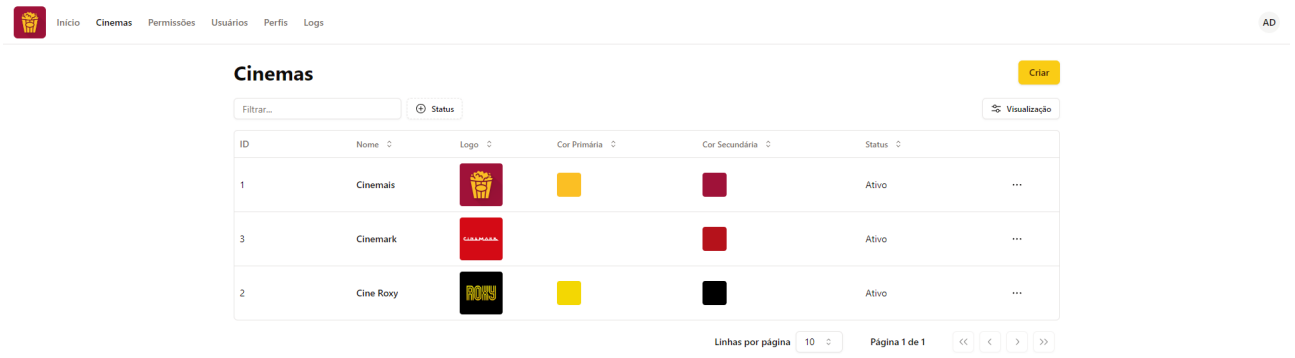
Figura 20 – Tela inicial da área administrativa



Fonte: Elaborado pelo autor

Essa sessão é exclusiva para os administradores do Cinemais, dando a possibilidade de gerenciar os cinemas existentes na plataforma e seus dados (Figura 21):

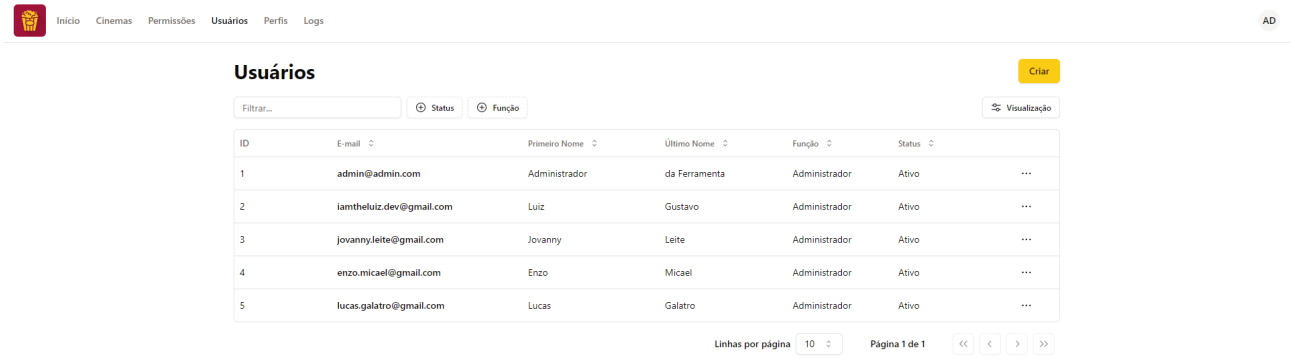
Figura 21 – Tela de gerenciamento de cinemas da área administrativa



Fonte: Elaborado pelo autor

Bem como a verificação dos usuários e suas autorizações cadastradas (Figura 22):

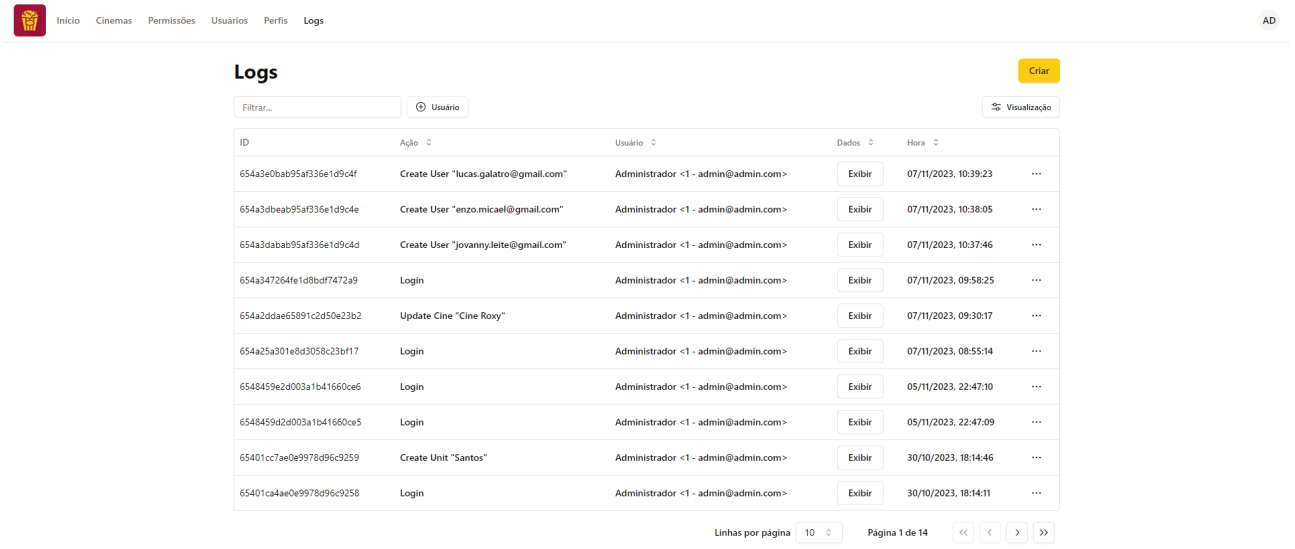
Figura 22 – Tela de gerenciamento de usuários da área administrativa



Fonte: Elaborado pelo autor

Complementarmente, a área de administração do sistema tem a possibilidade de verificar os logs de atividades que são executadas na plataforma, obtendo o registro das mudanças performadas (Figura 23):

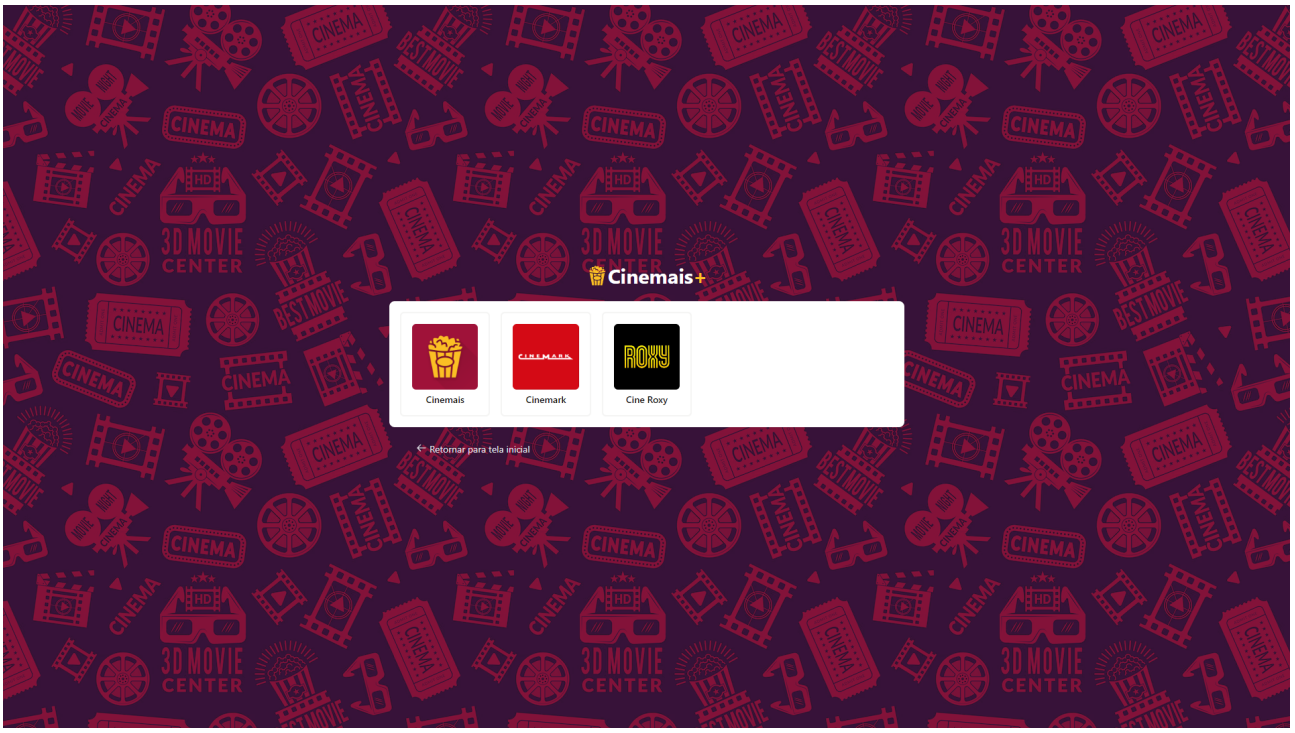
Figura 23 – Tela de gerenciamento de logs da área administrativa



Fonte: Elaborado pelo autor

Ao clicar no botão “Entrar”, presente na tela inicial do sistema, o usuário é redirecionado para a seleção de qual cinema ele tem autorização para acessar (Figura 24):

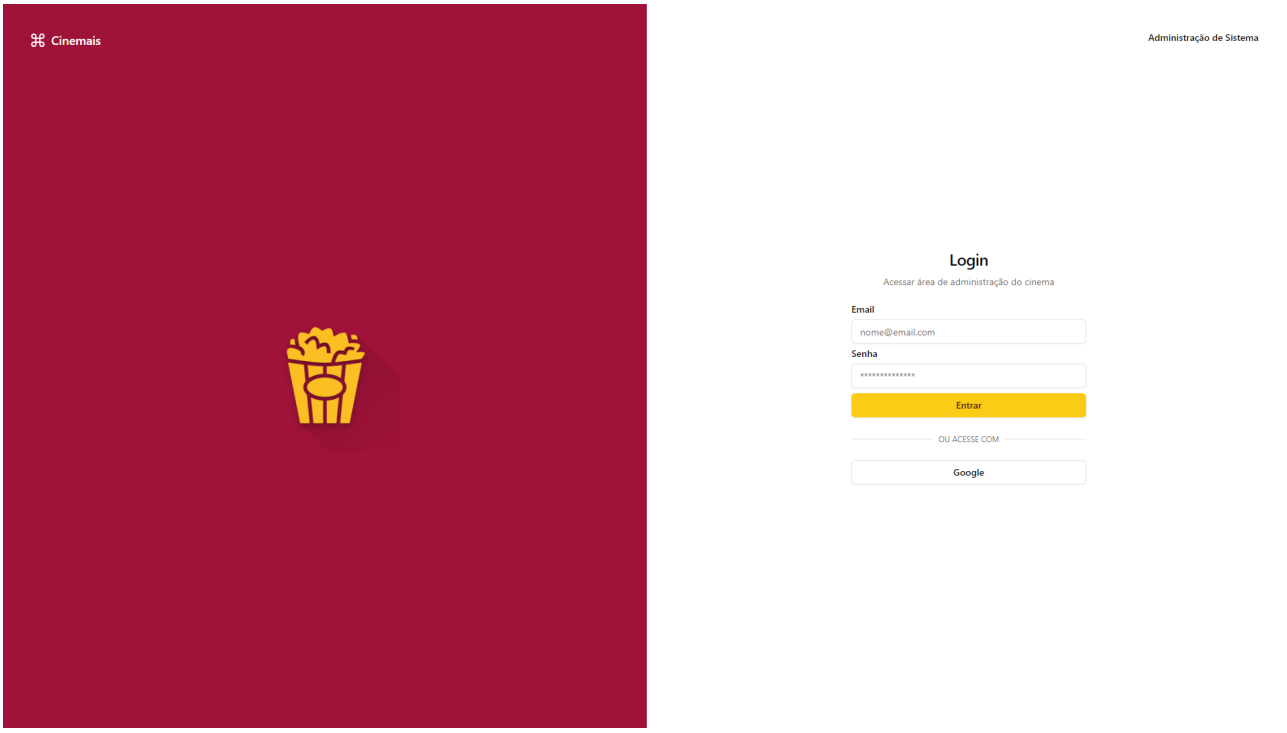
Figura 24 – Lista de cinemas para acesso



Fonte: Elaborado pelo autor

Selecionando o cinema desejado, será exibido um formulário para preenchimento das credenciais de acesso do usuário (Figura 25):

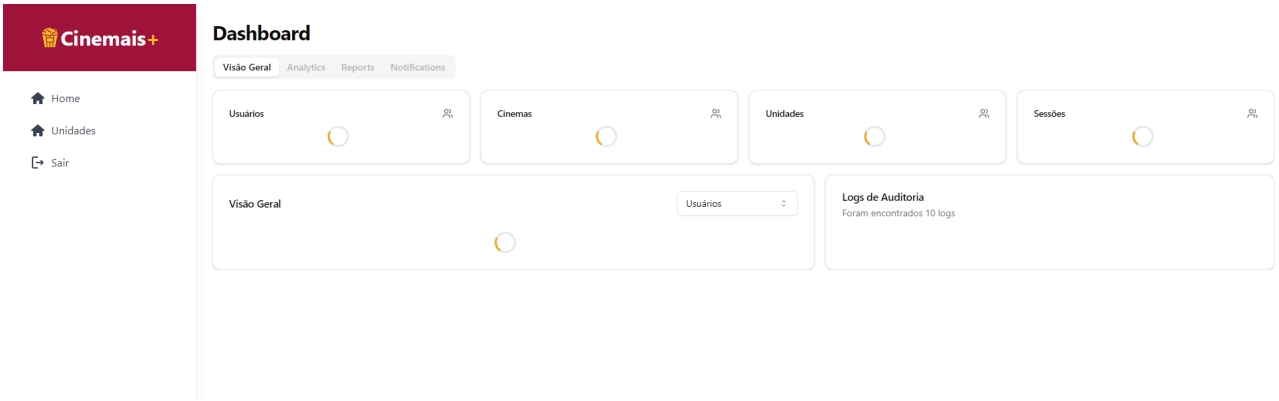
Figura 25 – Tela de login no cinema escolhido



Fonte: Elaborado pelo autor

Caso sejam fornecidas as credenciais corretas para um usuário cadastrado, é feito o redirecionamento para a tela inicial da gestão do cinema, que possibilita a administração exclusiva das unidades vinculadas a esse ambiente (Figura 26):

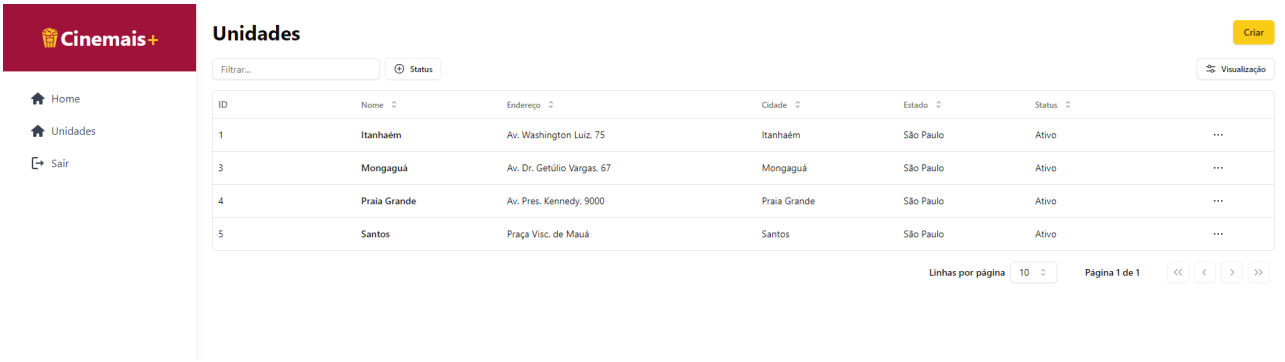
Figura 26 – Tela inicial da gestão do cinema



Fonte: Elaborado pelo autor

Ao acessar o menu “Unidades”, é possível verificar cada uma das instalações gerenciadas pelo cinema (Figura 27):

Figura 27 – Tela de gestão de unidades do cinema



Fonte: Elaborado pelo autor

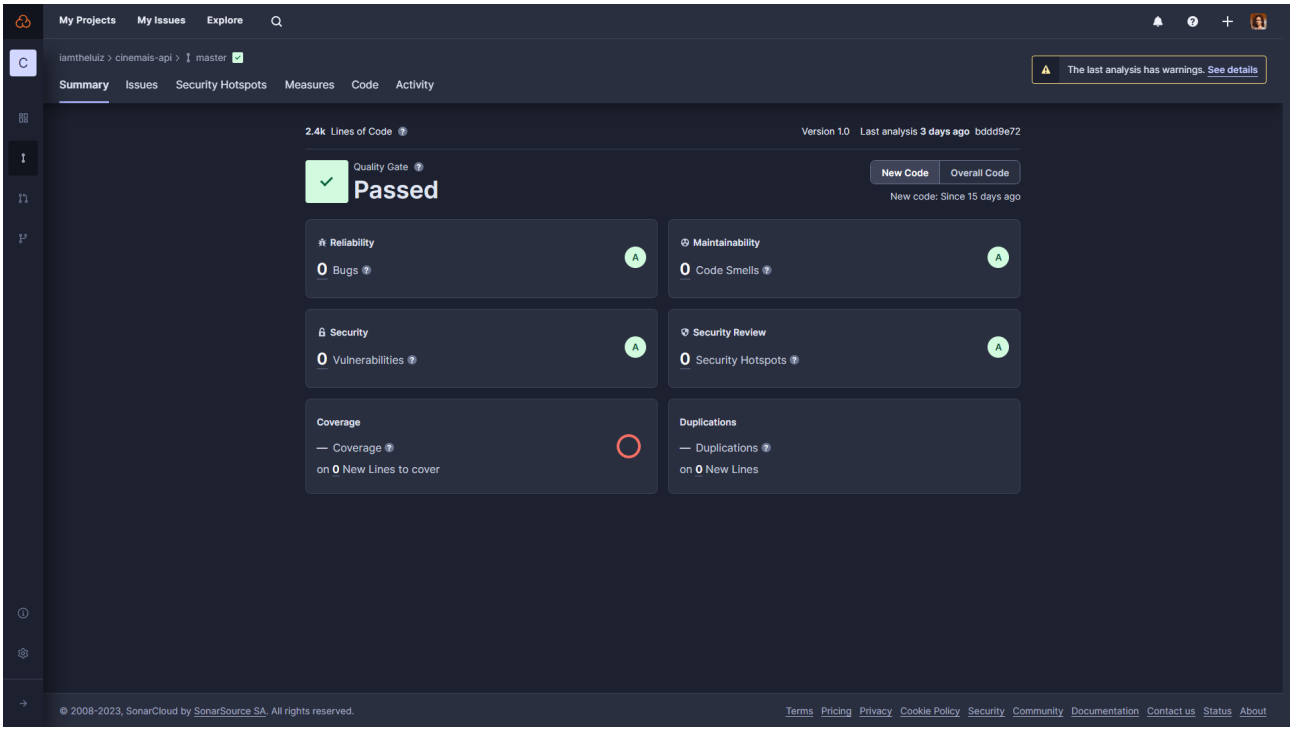
4.4 Análise de Qualidade do Código

Além da implementação obtida do projeto, uma das prioridades definidas como parte essencial do processo foi a garantia da segurança e, por consequência, a qualidade do código desenvolvido.

Para essa verificação, a ferramenta SonarCloud foi o indicador quantitativo em relação a falhas de código, más práticas de desenvolvimento, vulnerabilidades e pontos de atenção que precisam ser analisados.

Após a conclusão do desenvolvimento do projeto, foi possível manter a qualidade do software e das novas funcionalidades implementadas. Esse indicador pode ser verificado diretamente na interface da ferramenta, que mantém a contabilidade de cada um desses itens para o projeto, como pode ser visto nas análise relacionadas ao Backend do sistema (Figura 28):

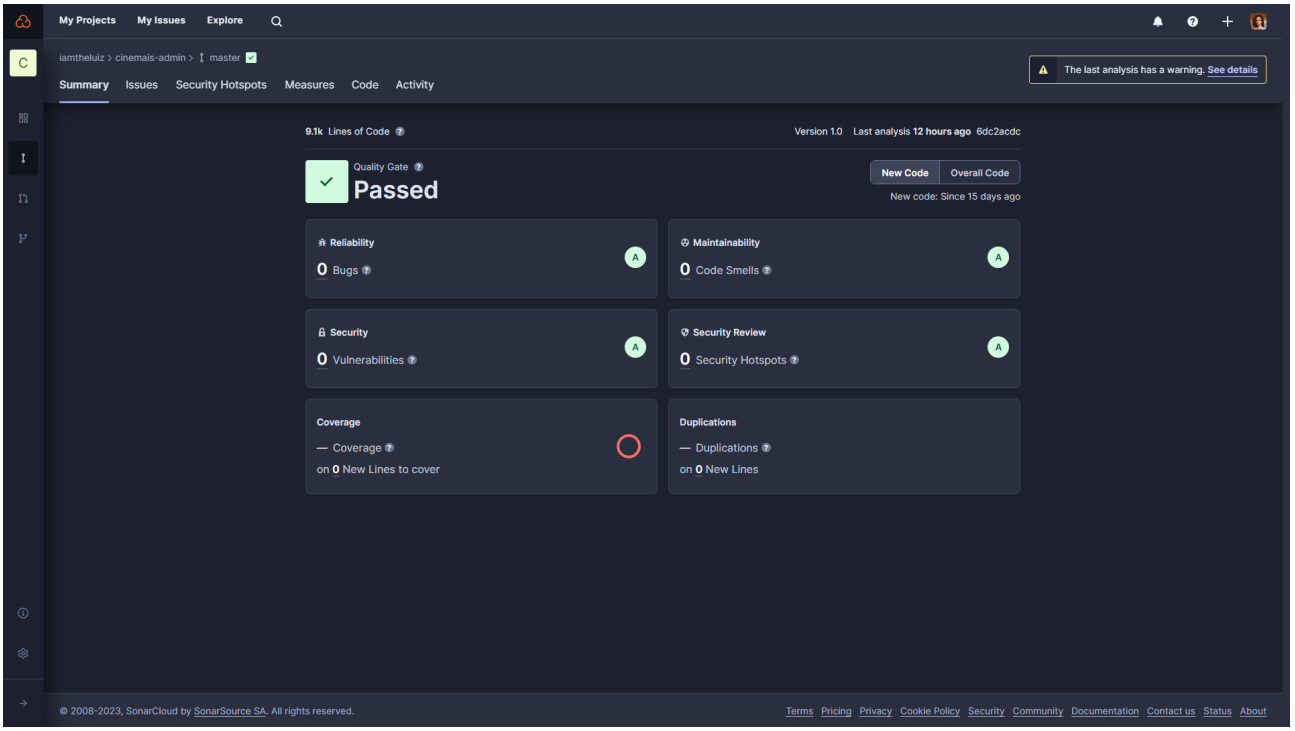
Figura 28 – Análise de código do Backend



Fonte: Elaborado pelo autor

Bem como nas análise realizadas para o Frontend do projeto (Figura 29):

Figura 29 – Análise de código do Frontend



Fonte: Elaborado pelo autor

Dessa forma, em conjunto com as tecnologias escolhidas para fazer o desenvolvimento do projeto e as metodologias para validação dos requisitos implementados, a análise de código por meio de uma ferramenta especializada como o SonarCloud supre um aspecto importante que não pode ser quantificado apenas com os testes de validação pelo usuário, que seria a possibilidade de medir o grau de qualidade do que foi tecnicamente escrito para a criação do sistema.

Esse mesmo processo poderia ser feito por um dos integrantes do projeto durante a codificação, com prejuízo da velocidade e eficiência do processo.

5. Considerações Finais

Durante o desenvolvimento do projeto foi dada a oportunidade para os membros do grupo de conhecer novos recursos relacionados ao processo de gestão, desenvolvimento e manutenção de sistemas, criando uma base de novos conhecimentos aplicáveis.

Dentre os diversos recursos aplicados, foi possível implementar os aspectos propostos como objetivo de modo prático e visualizando os benefícios decorrentes dessas implementações, destacando-se a aplicação dos princípios de DevOps para lidar com todas as fases de construção.

Devido ao escopo definido para o projeto, após a consideração do esforço necessário para o desenvolvimento dos requisitos, foi verificado que existem aspectos que podem ser estendidos e melhorados, como a adição de funcionalidades para a gestão de recursos e materiais internos de cada uma das unidades de cinemas, considerando itens como inventário de produtos e vendas feitas dentro de cada empreendimento, possibilitando uma administração baseadas em índices calculados de modo sistêmico e centralizado.

Em relação ao processo de análise de código, foi verificada a necessidade de avaliar o uso de ferramentas complementares ao SonarCloud para a análise da aplicação, já que ele foca em analisar o sistema apenas em seu código fonte, sem a avaliação dele publicado em funcionamento, abrindo espaço para a presença de problemas que não são diagnosticados pela plataforma.

6. Referências

ADRIANO, Thiago Da Silva. **Guia prático de TypeScript: Melhore suas aplicações JavaScript**. São Paulo: Casa do Código, 2021.

BASS, L.; Weber, I.; Zhu, L. DevOps: **A software architect's perspective**. [S.l.]: Addison-Wesley Professional, 2015.

BABICH, W. A. **Configuration Management Best Practices: Practical Methods that Work in the Real World**. Addison-Wesley Professional, 2010.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V., et al. (2001). **Manifesto for Agile Software Development**. Agile Alliance.

BROWN, E. **Web Development with Node and Express: Leveraging the JavaScript Stack. [S.l.]**: O'Reilly Media, 2014.

CHEN, L. **Continuous delivery: Huge benefits, but challenges** too. IEEE software, IEEE 2015.

COPPOLA, M. **Node.js Design Patterns**. Birmingham: Packt Publishing, 2016.

DANIELSSON, W. **React native application development**. Linköpings universitet, Swedia, 2016.

EBERT, C.; GALLARDO, G.; HERNANTES, J.; SERRANO, N. **Devops**. Ieee Software, IEEE, 2016.

FOWLER, M. **Patterns of Enterprise Application Architecture**. Boston: Addison-Wesley, 2002.

FOWLER, M.; FOEMMEL, M. **Continuous integration**. 2006. Disponível em: <https://www.martinfowler.com/articles/continuousIntegration.html>. Acesso em: 30 out. 2023.

FOWLER, M.; HUMBLE, J. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. Addison-Wesley, 2010.

GOMES, L. V. N. **Desenhando: um panorama dos sistemas gráficos**. Santa Maria: Ed.UFSM, 1998.

GOTTESDIENER, E.; GORMAN, M. **Discover to Deliver: Agile Product Planning and Analysis**. EBG Consulting, 2015.

KHOLMATOVA, A. **Design Systems: A practical guide to creating design languages for digital products**. [S.l.]: Smashing Media AG, 2017.

KINIBERG, H. **Scrum e XP Direto das Trincheiras**. Estocolmo: C4media Inc, 2007.

KUMAR, A.; SINGH, R. K. **Comparative analysis of angularjs and reactjs**. International Journal of Latest Trends in Engineering and Technology, v. 7, n. 4, p. 225–227, 2016.

LAURINDO, F. J. B.; SHIMIZU, T.; CARVALHO, M. M.; RABECHINI, R. **O Papel da Tecnologia da Informação (TI) na Estratégia das Organizações**. Gestão & Produção. v.8, n.2, p.160-179, ago. 2001.

MARCOLINO, A. S. **Frameworks Front End**. Editora Saraiva, 2021.

MARDAN, A. **Express.js Guide: The Comprehensive Book on Express.js. [S.l.]**: Azat Mardan, 2014.

MARTIN, R. C. **Agile Software Development: Principles, Patterns, and Practices**. Pearson, 2003.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. Wiley, 2011.

PRESSMAN, R.S. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. Porto Alegre: AMGH, 2011.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do Trabalho Científico: métodos e técnicas da pesquisa e do trabalho acadêmico**. 2ª ed. Novo Hamburgo, RS: Feevale, 2013.

RASK, J. K.; MADSEN, F. P.; BATTLE, N.; MACEDO, H. D.; LARSEN, P. G. **Visual studio code vdm support**. John Fitzgerald, Tomohiro Oda, and Hugo Daniel Macedo (Editors), 2021.

RAVICHANDRAN, A.; TAYLOR, K.; WATERHOUSE, P. **Devops in the ascendency. In: DevOps for Digital Leaders**. [S.l.]: 2016.

ROSSBERG, J. **Agile Project Management with Azure DevOps: concepts, templates, and metrics**. New York City: Apress, 2019.

SABBAGH, R. **Scrum: Gestão Ágil para Projetos de sucesso**. São Paulo: Casa do Código, 2014.

SCHWABER, K; SUTHERLAND, J. **Guia do SCRUM**. Cidade: Scrum.Org and ScrumInc 2014.

SUAREZ, M. et al. **Design Systems Handbook**. 2018. Disponível em: <https://www.designbetter.co/design-systems-handbook>. Acesso em: 30 out. 2023.

VIEGA, J.; MCGRAW, G. **Building Secure Software: How to Avoid Security Problems the Right Way**. Addison-Wesley Professional, 2002.

Internet

ANDRADE, A. P. de. **O que é Firebase?** 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-firebase>. Acesso em: 23 out. de 2023.

BATISCHINSKI, G. **Backend as a Service: Prós e Contras**. 2016. Disponível em: <https://www.infoq.com/br/news/2016/07/backend-pros-e-contras/>. Acesso em: 23 de out. de 2021.

BITBUCKET, Atlassian. **What is Git**. Disponível em: <https://www.atlassian.com/git/tutorials/what-is-git>. Acesso em: 22 nov. 2022.

DAHL, R. 2009. **Ryan Dahl: Node JS – JSConf.eu 2009**: Disponível em: <https://www.youtube.com/watch?v=EeYvFl7li9E&ab_channel=JSConf>. Acesso em: 10 out. 2023

EXPRESS.JS. 2023. **Express - framework de aplicativo da web Node.js**. Disponível em: <https://expressjs.com/pt-br/>. Acesso em: 19 out. 2023.

FIREBASE, D. **Firestore | Firebase**. 2021. Disponível em: <https://firebase.google.com/docs/firestore?hl=pt-br>. Acesso em: 23 out. 2023.

MICROSOFT. **O que é o DevOps?** 2023. Disponível em: <https://azure.microsoft.com/ptbr/overview/what-is-devops/>. Acesso em: 20 out. 2023.

MICROSOFT. **Visual Studio Code Documentation**. 2023. Disponível em: <https://code.visualstudio.com/docs>. Acesso em 20 out. 2023.

MICROSOFT DEVELOPER. 2019. **Inside TypeScript with Anders Hejlsberg - BDL2011**: Disponível em: <https://www.youtube.com/watch?v=tXK50czRbdA&ab_channel=MicrosoftDeveloper>. Acesso em: 10 out. 2023

NODE.JS. 2023. **About Node.js® | Node.js**. Disponível em: <https://nodejs.org/en/about>. Acesso em: 19 out. 2023.

NPM TRENDS. **Angular vs react vs vue**. 2023. Disponível em: <https://npmtrends.com/angular-vs-react-vs-vue>. Acesso em: 20 out. 2023.

POSTGRESQL. 2023. **PostgreSQL: About**. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 19 out. 2023.

PRISMA. 2023. **Prisma | Next-generation ORM for Node.js & TypeScript**. Disponível em: <https://www.prisma.io/>. Acesso em: 19 out. 2023.

REACT. 2023. **React – Uma biblioteca JavaScript para criar interfaces de usuário**. Disponível em: <https://pt-br.legacy.reactjs.org/>. Acesso em: 19 out. 2023.

SHADCN/UI. 2023. **shadcn/ui**. Disponível em: <https://ui.shadcn.com/>. Acesso em: 19 out. 2023.

TYPESCRIPT. 2023. **TypeScript: JavaScript With Syntax For Types.**. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 19 out. 2023.