Backend de uma Plataforma para Gerenciamento de Trabalho de Conclusão de Curso

Trabalho de Conclusão de Curso

Engenharia da Computação

Selton Felipe Guedes da Silva

Orientador: Prof. Daniel Augusto Ribeiro Chaves

Coorientador: Prof. Joabe Bezerra de Jesus Júnior

Selton Felipe Guedes da Silva

Backend de uma Plataforma para Gerenciamento de Trabalho de Conclusão de Curso

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Engenharia de Computação

Escola Politécnica de Pernambuco

Universidade de Pernambuco

Orientador: Prof. Daniel Augusto Ribeiro Chaves

Coorientador: Prof. Joabe Bezerra de Jesus Júnior

Recife - PE, Brasil

novembro de 2022

Silva, Selton Felipe Guedes da

Backend de uma Plataforma para Gerenciamento de Trabalho de Conclusão de Curso / Selton Felipe Guedes da Silva. – Recife - PE, 2022.

xx, 32 f.: il. xx; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) Universidade de Pernambuco, Escola Politécnica de Pernambuco, Recife, 2022.

Orientador: Prof. Dr. Daniel Augusto Ribeiro Chaves.
Co-orientador: Prof. Ass. Joabe Bezerra de Jesus Júnior.

Inclui referências.

1. Desenvolvimento de Software. 2. Backend. 3. Sistema. 4. Aplicação. 5.Trabalho de Conclusão de Curso. I. Backend de uma Plataforma para Gerenciamento de Trabalho de Conclusão de Curso. II. Chaves, Daniel Augusto Ribeiro. III. Universidade de Pernambuco.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 19/10/2022, às 15h15min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **SELTON FELIPE GUEDES DA SILVA**, orientado(a) pelo(a) professor(a) **DANIEL AUGUSTO RIBEIRO CHAVES**, sob título Gestão de Documentação na Disciplina do Trabalho de Conclusão de Curso: Backend, a banca composta pelos professores:

LARISSA TENÓRIO FALCÃO ARRUDA (PRESIDENTE)
DANIEL AUGUSTO RIBEIRO CHAVES (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 9,0 (nove)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

Larissa T. Falcão Arruda

AVALIADOR 1: Prof (a) **LARISSA TENÓRIO FALCÃO ARRUDA**

AVALIADOR 2: Prof (a) **DANIEL AUGUSTO RIBEIRO CHAVES**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Este trabalho é dedicado à minha família.

Agradecimentos

Esta fase da minha vida é muito especial e não posso deixar de agradecer a Deus por toda força, ânimo e coragem que me ofereceu para ter alcançado minha meta. A esta instituição tão imponente eu agradeço pelo ambiente propício à evolução e crescimento, bem como a todas as pessoas que a tornam assim tão especial. Deixo também um agradecimento especial aos meus professores e orientadores, pois sem eles esta monografia não teria sido possível.

À minha família e amigos, que nunca desistiram de mim e sempre me ofereceram amor, eu deixo uma palavra e uma promessa de gratidão eterna. Foram eles que me incentivaram e inspiraram através de gestos e palavras a superar todas as dificuldades. Em especial minha mãe (Silvana), meus avós (Neves, Sebastião, Bila e Eunice), meus tios (Silvio e Sivaldo) e tias (Wberliana e Wberlanice), meus irmãos (Syel e Cecília) e minhas primas (Sara, Samara, Samilly e Sophia) vocês são grandes exemplos para mim, agradeço por tudo o que vocês fizeram por mim ao longo deste tempo e da minha vida, por sempre terem me apoiado, incentivado e motivado a seguir os meus sonhos. A vocês devo tudo.

Sou eternamente grato a Carol Veras e Luciana Passos por estarem comigo, me apoiarem e motivarem nos momentos mais difíceis do curso. Sua ajuda e apoio foram muito importantes para mim, e nunca vou esquecer tudo que vocês fizeram por mim. Muito obrigado! Com todo o carinho e de coração eu agradeço.

Ao professor Adalberto Júnior que me guiou nos meus primeiros passos no mundo da programação agradeço e reconheço a orientação repleta de conhecimento, sabedoria e paciência. Sou muito grato pois sem ele não teria chegado até aqui.

Agradeço a Rafael Lindoso por me motivar, aconselhar e fornecer acesso a cursos que ajudaram no meu desenvolvimento como profissional e no desenvolvimento do presente trabalho. Muito obrigado, sem sua ajuda eu não teria chegado até aqui.

Aos meus amigos Murilo Stodolni, Richard Jeremias, Marcel Souza, Matheus Alves, Michael Cavalcanti, Nilton Vieira e todos aqueles que participaram da minha jornada e não foram citados aqui, deixo minha gratidão, pois foram eles que fizeram com que eu seguisse sempre de cabeça erguida. Agradeço a eles do fundo do meu coração por todos os momentos que compartilhamos.

Não esqueço – é claro – todas as pessoas que não referi mas que fizeram parte do meu percurso. A todas eu deixo um agradecimento honesto e muito sentido. A todas as pessoas que de uma alguma forma me ajudaram e acreditaram em mim eu deixo um agradecimento eterno, porque sem elas não teria sido possível.

"Truth is ever to be found
in simplicity, and not
in the multiplicity and
confusion of things."
Isaac Newton

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Autorização de publicação de PFC

Eu, **Selton Felipe Guedes da Silva** autor(a) do projeto de final de curso intitulado: **Gestão de Documentação na Disciplina do Trabalho de Conclusão de Curso: Backend;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

Selton Felipe Guedes da Silva

Orientador(a): **Daniel Augusto Ribeiro Chaves**

Coorientador(a): **Joabe Bezerra de Jesus Júnior**

Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 19/10/2022

Resumo

O presente trabalho dedica-se à criação da API de uma plataforma para gerenciamento de trabalhos de conclusão de curso, permitindo a visualização e gerenciamento dos projetos dos concluintes do curso de Engenharia da Computação da Universidade de Pernambuco. A aplicação visa suprir as necessidades do professor da disciplina, corpo docente e discente da UPE. O framework usado no desenvolvimento da API foi o Express, que tem como objetivo fornecer uma camada de recursos fundamentais para aplicativos da web, como: métodos utilitários HTTP e middleware (software que provê serviços e recursos comuns a aplicações), permitindo a criação rápida e fácil de uma API. A linguagem de programação TypeScript foi escolhida por possuir recursos de projetos em larga escala, como: orientação a objetos, tipagem estática e detecção de erros e má prática em tempo de compilação. Uma grande vantagem dessa linguagem é o fato de compilar o código escrito em TypeScript para a linguagem de programação JavaScript, ou seja, é multiplataforma - pode ser utilizada em qualquer navegador e sistema operacional. Com essas ferramentas, é possível disponibilizar rotas que fornecerão os meios para que outros sistemas possam acessar os serviços oferecidos pelo backend desenvolvido. As rotas para gerencimento de usuários, permissões e solicitações de orientador foram desenvolvidas seguindo o padrão arquitetural MVC.

Palavras-chave: TCC. Backend. Desenvolvimento. Sistema.

Abstract

The present work is dedicated to create an API to a management platform to manager final paper, granting visualization and management of projects of the Computer Engineering course graduates of the University of Pernambuco. The application has the purpose of supplying the needs of the professor of the discipline, faculty and student at UPE. The framework used for the development of the API was Expresso, which aims to provide a layer of fundamentals resources to web applications how: HTTP utilities methods and middleware (software the provide services and common resources to applications), allowing the fast and easy API creation. The TypeScript programming language was chosen because of its large-scale design capabilities, such as: object orientation, static typing and errors and bad practices detection at compile time. A great advantage of this language is the fact that it compiles the code written in Typescript to the JavaScript programming language, in other words it is cross-platform - it can be used in any browser and operating system. With these tools, it is possible to provide routes that will provide the means for other systems to access the services offered by the developed back-end. The routes for managing users, permissions and orientation requests were developed following the MVC architectural pattern.

Keywords: TCC. Back-end. Development. System.

Lista de Figuras

Lista de abreviaturas e siglas

API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
TCC	*Trabalho de Conclusão de Curso*
TS	*Typescript*
MVC	*Model-View-Controller*

Sumário

1 Introdução

Este trabalho de conclusão de curso propõe-se ao desenvolvimento do backend de uma plataforma para gerencimanto de Trabalhos de Conclusão de Curso (TCC), permitindo a visualização e gerenciamento dos projetos dos concluintes do curso de Engenharia de Computação da Escola Politécnica de Pernambuco (POLI) da Universidade de Pernambuco (UPE).

A dificuldade de gerir os Trabalhos de Conclusão de Curso está no fato de não haver uma plataforma única que permite a gestão de maneira fácil e acessível por parte do professor responsável pela disciplina de Trabalho de Conclusão de Curso.

1.1 Caracterização do Problema

O desenvolvimento de um TCC é um processo obrigatório nos cursos de graduação (bacharelados e licenciaturas) [1][2] que funciona como um meio para os alunos aplicarem os conhecimentos adquiridos durante o curso de graduação. Na UPE, a gestão dos trabalhos de conclusão de curso é feita pelo professor encarregado de ministrar a disciplina de Trabalho de Conclusão de Curso. Entretanto, não há uma plataforma única para gestão dos trabalhos, fazendo-se necessário a utilização de *software* de terceiros para auxiliar nas atividades e processos relacionados à disciplina.

No curso de Engenharia de Computação da UPE, segundo o manual de normas do PFC (Projeto Final de Curso) [3], os processos relacionados ao TCC podem ser resumidos em 5 cinco etapas:

1. Escolha do tema e professor orientador;
2. Apresentação de um projeto de TCC
3. Apresentação de um texto monográfico;
4. Apresentação pública do trabalho desenvolvido e avaliação da banca;
5. Avaliação/aprovação final do professor da disciplina.

O primeiro passo do desenvolvimento de um TCC, é a escolha de um tema. Na UPE, é necessário o envio de um termo de concordância assinado pelo orientador, pelo co-orientador (se houver), pelo professor da disciplina e pelo aluno como documento comprobatório da solicitação e da concordância do orientador e do tema do trabalho.

A segunda etapa do processo é a entrega do pré-projeto elaborado pelo aluno e assinado pelo orientador. Esse pré-projeto será avaliado pelo professor da disciplina e uma nota será concedida ao final dessa fase. O pré-projeto segue um padrão proposto pelo professor, normalmente composto pelos tópicos: caracterização do problema, objetivos e metas, resultados e impactos esperados, conclusões, referências bibliográficas e cronograma de atividades.

Após a entrega do pré-projeto, dá-se início a terceira etapa, na qual o discente começa a desenvolver seu trabalho final. É preciso preparar um projeto que contenha informações como assunto, objetivos e cronograma das atividades [4] para que os professores e orientadores consigam enxergar melhor o assunto, os objetivos e problemas levantados. Essa etapa é composta pela escrita do documento, aprovação do orientador e definição da banca examinadora. O documento entregue ao final dessa fase deve seguir uma das seguintes formas de apresentação: texto monográfico, artigos técnicos ou científicos, protótipos de invenção, *software* e pedidos de patente.

A etapa seguinte inicia-se com a apresentação pública do aluno sobre o seu trabalho. Nessa apresentação o aluno dispõe de 30 minutos para defender o seu trabalho e a banca avaliadora (composta por professores especialistas) tem até 30 minutos para dúvidas e questionamentos relacionados ao trabalho. A banca examinadora avalia a apresentação, o documento e aprova ou, se necessário, solicita revisão. A banca avaliadora se reúne e atribui uma nota ao trabalho do aluno.

Em seguida, o professor da disciplina dá início a última etapa onde deve avaliar o documento já revisado pela banca e discente. Caso alguma revisão seja necessária, o documento será revisado pelo aluno e logo voltará para as mãos do professor da disciplina até que o mesmo seja aprovado.

Nesse contexto, este trabalho tem como objetivo desenvolver uma Interface de Programação de Aplicação (API) para o sistema de gestão de TCCs do curso de Engenharia da Computação da UPE. Uma API é um conjunto de padrões que permitem, de uma forma simples e prática, a comunicação entre dois componentes de *software* usando um conjunto de definições e protocolos. Assim sendo, uma API será criada para possibilitar a comunicação entre os componentes desenvolvidos e os usuários, interfaces e ainda outras aplicações.

1.2 Objetivo Geral

Este trabalho tem como objetivo principal desenvolver o *backend* para um sistema escalável e seguro para gestão dos Trabalhos de Conclusão de Curso do curso de Engenharia de Computação da Universidade de Pernambuco (UPE). O *backend* é responsável por garantir o funcionamento da aplicação no que se refere à estrutura, segurança e banco de

dados.

Os principais objetivos são: permitir o cadastro de usuários (professores, alunos e administradores), solicitação de orientação por parte dos alunos e aprovação/rejeição das solicitações por parte dos professores.

1.3 Objetivos Específicos

Utilizando a linguagem de programação *TypeScript*, o *framework Express* e padrões de projeto tornam o código mais flexível, fácil de ser lido, entendido e reusável. Permitindo o desenvolvimento de uma API (Interface de Programação de Aplicativos) com código reusável, flexível e escalável para atingir as metas específicas que foram definidas para definir o escopo do projeto e estruturar seu desenvolvimento. São elas:

- Criar o Banco de Dados MySQL;
- Estruturar uma arquitetura backend com *TypeScript* e *Express*;
- Usar Docker para empacotar toda a aplicação, dependências e facilitar a distribuição.
- Autenticação de usuários através do Firebase e seus serviços de autenticação para permitir que os dados dos usuários sejam salvos na nuvem com segurança.
- Desenvolvimento da API para permitir acesso aos dados do sistema;
- Implementação de boas práticas de arquitetura e segurança (por exemplo: validação de permissões de usuário, validação dos campos de formulário e identificadores imprevisíveis).

1.4 Funcionalidades do *Software*

1.4.1 Autenticação e Autorização (Controle de Acesso)

O controle de acessos do sistema é feito a partir de uma lista de usuários que possuem permissão para acessar o sistema. A lista é gerenciada pelos usuários Administradores, onde os mesmos têm permissão para visualizar, adicionar, remover e atualizar as informações de cada permissão de usuário.

Para efetuar login no sistema, é necessário que o usuário tenha uma conta do Google e uma permissão atrelada ao seu email na lista de permissões de usuário, seja ela de Estudante, Professor ou Administrador. No primeiro acesso, será solicitado ao usuário: CPF, Nome, Sobrenome e Telefone para completar o seu cadastro.

Sempre que uma nova requisição for feita, o token do usuário deverá ser enviado no cabeçalho da requisição para que seja possível fazer a autenticação do usuário.

1.4.2 Solicitação de Orientação

Todo usuário com permissão de Estudante pode criar, remover, atualizar e reenviar solicitações de orientação para professores. No momento da solicitação, o usuário deve informar: o título do seu trabalho, a sub-área, descrição (opcional), professor orientador, professor coorientador (opcional) e anexar o termo de concordância.

As solicitações de orientação são enviadas para os professores orientadores e os mesmos podem aprovar ou rejeitar. Quando uma solicitação de orientação é rejeitada, o aluno deve enviar uma nova solicitação.

1.4.3 Listagem de Solicitações de Orientação

Cada aluno tem acesso a cada uma de suas solicitações de orientação, assim como cada professor tem acesso as solicitações de orientação onde o mesmo está presente como orientador ou coorientador. O administrador do sistema tem permissão para inativar e visualizar toda e qualquer solicitação de orientação criada.

2 Requisitos de *Software*

Antigamente dizia-se que requisitos eram sinônimos de funções, mas atualmente foi assumido que requisitos de *software* são muito mais do que apenas funções. Requisitos são descrições dos recursos e funcionalidades que o sistema deve possuir para satisfazer contratos, padrões ou especificações de acordo com o(s) desejo(s) do(s) usuário(s). De forma mais geral um requisito é uma condição necessária para satisfazer um objetivo [5].

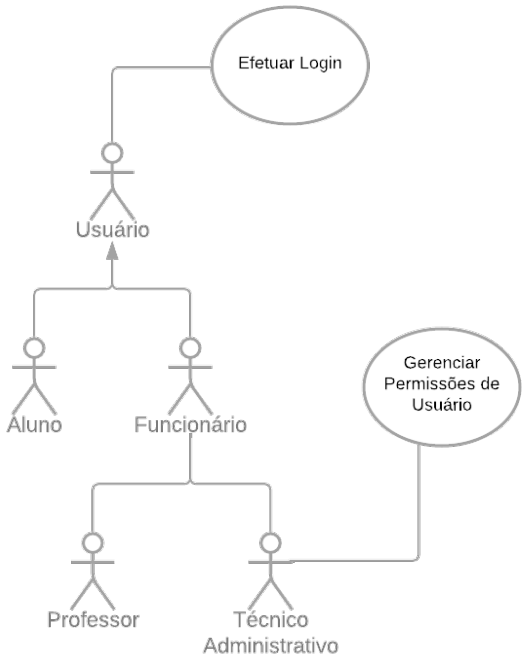
O conjunto dos requisitos como um todo representa um acordo negociado entre todas as partes interessadas no sistema, mas não significa que os desenvolvedores bem entendidos não possam contribuir com sugestões e propostas que levem em conta o desejo do cliente [5].

Ao satisfazer tais expectativas, se garante que o sistema seja funcional e confiável, bem como apresente um desempenho que está de acordo com o que foi especificado. Nesta seção está descrito todos os requisitos de *software* e casos de uso coletados para o bom funcionamento do sistema.

2.1 Casos de Uso

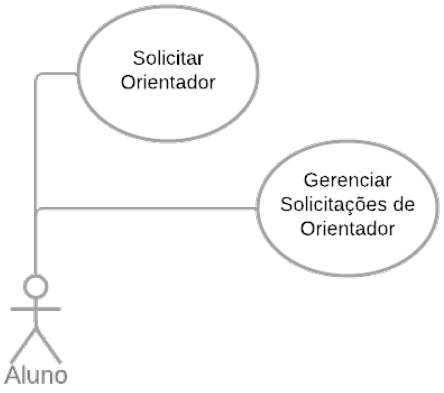
Os usuários são divididos em dois grupos: Alunos e Funcionários. Aluno não pode ser um administrador, enquanto que Funcionários podem ser Professores ou Técnico Administrativos, conforme apresenta a figura 1. Usuários administradores podem gerenciar as permissões e todos os usuários, ou seja, podem adicionar, remover, visualizar e atualizar as permissões de qualquer usuário.

Figura 1 – Usuários do sistema



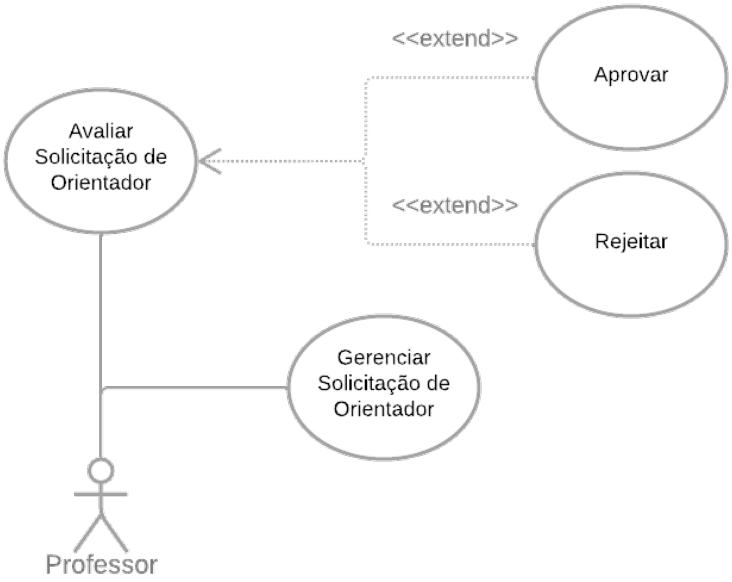
Cada Aluno tem acesso as suas solicitações de orientador e poderá gerenciar suas solicitações (figura 2), ou seja, os alunos podem visualizar, adicionar, deletar e atualizar suas solicitações, mas nunca solicitações de outros alunos.

Figura 2 – Casos de uso de Alunos



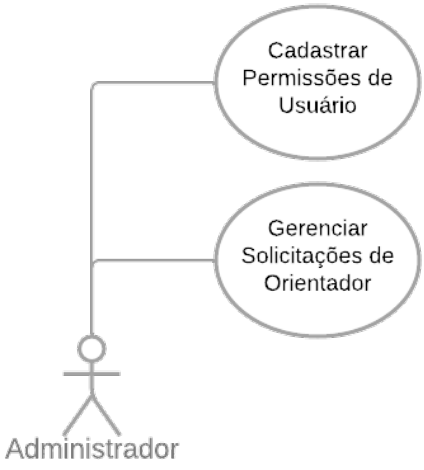
Cada professor é responsável por gerenciar e avaliar as suas solicitações. Cada solicitação pode ser Aprovada ou Rejeitada pelo professor, assim como mostra a figura 3.

Figura 3 – Casos de uso de Professores



Todo administrador pode gerenciar permissões de usuário e solicitações de orientação de todos os professores e alunos, conforme a figura 4.

Figura 4 – Casos de uso de Administradores



2.2 Especificação de Requisitos

2.2.1 Autenticação

1. RF01 - **Como um** administrador, **eu quero** cadastrar usuários manualmente, **para** permitir que os usuários acessem o sistema.

    - Critérios de Aceitação:

        a) **Dado um** usuário administrador,
        quando gerenciando um usuário,
        então devo estar habilitado para definir:
        - Email
        - Nome
        - Descrição
        - Permissão (Aluno ou Professor)
        - isAdmin

2. RF02 - **Como um** usuário, **eu quero** fazer login, **para** acessar o sistema

    - Critérios de Aceitação:

        a) **Dado um** usuário,
        quando clicar em fazer login com Google,
        então devo ser redirecionado para página inicial.

2.2.2 Solicitação de Orientador

1. RF01 - **Como um** aluno, **eu quero** solicitar orientação a um professor, **para** que possa dar continuidade ao meu tcc.

    - Descrição: A primeira etapa do Trabalho de Conclusão de Curso é definir o tema e o orientador. Ao solicitar orientação a um professor, o aluno deve escolher o seu tema.
    - Critérios de Aceitação:

        a) **Dado um** aluno,
        quando solicitando orientação,
        então devo estar habilitado para definir:
        - Título provisório do TCC
        - Descrição
        - Sub-área
        - Professor orientador

   - Professor co-orientador (opcional)

b) **Dado um** aluno,
quando solicitações de orientação forem criadas,
então devo estar habilitado para excluí-las.

c) **Dado um** aluno,
quando solicitações de orientação forem criadas,
então devo estar habilitado para atualizar todos os itens de (a).

2. RF02 - **Como um** professor, **eu quero** aceitar/rejeitar solicitações de orientação, **para** que os alunos possam dar continuidade ao seus trabalhos

   - Descrição: A primeira etapa do Trabalho de Conclusão de Curso é definir o tema e o orientador. Ao solicitar orientação a um professor, o aluno deve escolher o seu tema. O professor poderá aceitar ou recusar a orientação.
   - Critérios de Aceitação:

     a) **Dado um** professor,
     quando solicitado orientação,
     então devo estar habilitado para definir:
        - Aceitar ou Rejeitar a orientação

     b) As solicitações de orientador podem ter os seguintes estados:
        - Pendente: aguardando resposta do orientador
        - Rejeitada: professor rejeitou a solicitação
        - Aprovada:professor aceitou ser orientador

     c) Administradores poderão ativar/inativar as solicitações a qualquer momento.

3. RF03 - **Como um** professor, **eu quero** visualizar as solicitações de orientação enviadas para mim, **para** que eu possa controlar minhas requisições de orientação.

   - Critérios de Aceitação:

     a) **Dado um** professor,
     quando visualizar suas solicitações,
     então devo estar habilitado para ver:
        - Título
        - Sub-área
        - Autor da requisição
        - Data da requisição
        - Status

3 Referencial Teórico

A arquitetura de *software* representa os padrões de design e estruturas a serem seguidos durante o desenvolvimento de um sistema, ou seja, refere-se aos fundamentos estruturais na composição de blocos de software, sendo a disciplina responsável pela organização e comunicação dos elementos durante a construção de componentes modulares de *software* [6].

Nesta seção está descrito os detalhes referentes as tecnologias utilizadas no desenvolvimento e sobre o padrão arquitetural que serviu como base em todo o processo.

3.1 Ambiente de Trabalho

Em particular, o foco é desenvolver um *backend* escalável e seguro na linguagem de programação *TypeScript* (TS) [7] e o *framework Express*.

A linguagem de programação *TypeScript* é uma linguagem de código aberto baseada na linguagem *JavaScript* desenvolvida pela *Microsoft* que compila o código para *Javascript* simples e pode ser executado em qualquer navegador e sistema operacional [8]. O *Express* é um *framework* mínimo e flexível para o Node.js que fornece um conjunto de recursos para aplicações web e móvel [9]. O Node.js é um *software* multiplataforma de código aberto que permite a execução de códigos *JavaScript* fora de um navegador web projetado para criar aplicativos de rede escaláveis [10], assim sendo, é possível desenvolver uma aplicação escalável e multiplataforma com estas ferramentas.

Em relação ao armazenamento dos dados, está sendo utilizado o banco de dados MySQL que constrói um sistema administrativo de dados prático e ágil, garantindo que todas as informações estejam bem organizadas e seguras [11].

Para gerenciar a autenticação de usuários tratada na Seção 1.4.1, está sendo utilizado um banco de dados NoSQL hospedado na nuvem chamado Firebase. Com ele, é possível armazenar e sincronizar dados entre os usuários em tempo real [12]. O mesmo está sendo utilizado para armazenar os emails dos usuários e gerenciar os seus tokens de acesso ao sistema.

Para permitir o empacotamento de toda a aplicação e suas dependências, está sendo utilizado docker que permite a criação de contêineres com todas as configurações de ambientes necessárias para a execução da API. O Docker é uma plataforma Open Source que possibilita o empacotamento de uma aplicação dentro de um contêiner [13].

3.2 Arquitetura MVC

O criador do padrão MVC (*Model-View-Controller*), descreveu um padrão como sendo um problema que se repete inúmeras vezes em um mesmo contexto e que contém uma solução para resolvê-lo de tal modo que essa solução seja utilizada em diversas situações. O termo padrões de projeto ou *Design Patterns*, descreve soluções para problemas recorrentes no desenvolvimento de sistemas de *software* orientados a objetos [14].

Algumas vantagens de utilizar pelo menos um padrão de projeto em desenvolvimento de *software* são:

- **Baixo acoplamento:** o acoplamento significa o quanto uma classe depende da outra para funcionar. Quanto maior for essa dependência entre duas classes, mais fortemente essas classes estão acopladas. O ideal é diminuir o acoplamento para reduzir o impacto em possíveis manutenções e facilitar o gerenciamento e mudanças no negócio [15].
- **Coesão:** está ligado ao princípio da responsabilidade única que diz que uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira satisfatória [15]. Quando uma classe é elaborada com um único e bem focado propósito, é dito que ela tem alta coesão. Quando uma classe possui propósitos que não pertencem apenas a ela, é dito que ela tem uma baixa coesão.

Existem diversos padrões e cada um possui uma função bem específica. Ao utilizá-los de forma simultaneamente em um projeto de *software*, são trazidos diversos benefícios como:

- Aumento de produtividade.
- Uniformidade na estrutura do *software*.
- Facilidade de mantenção.
- Redução de complexidade do código
- Redução do tempo de desenvolvimento do projeto
- Reutilização de módulos do sistema em outros sistemas

A arquitetura MVC é um padrão responsável por contribuir na otimização da velocidade entre as requisições feitas pelo comando dos usuários. Essa arquitetura é dividida em três camadas essenciais: Model, View e Controller [16]. Cada uma delas executa o que lhe foi definido e nada além disso.

- Model: é responsável por controlar e gerenciar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.
- View: é a camada responsável por apresentar as informações de forma visual ao usuário através de uma combinação de gráficos e textos.
- Controller: é responsável por interpretar as entradas de dados e mapeiar essas ações do usuário em comandos que são enviados para o Model e/ou View para que as devidas ações sejam realizadas.

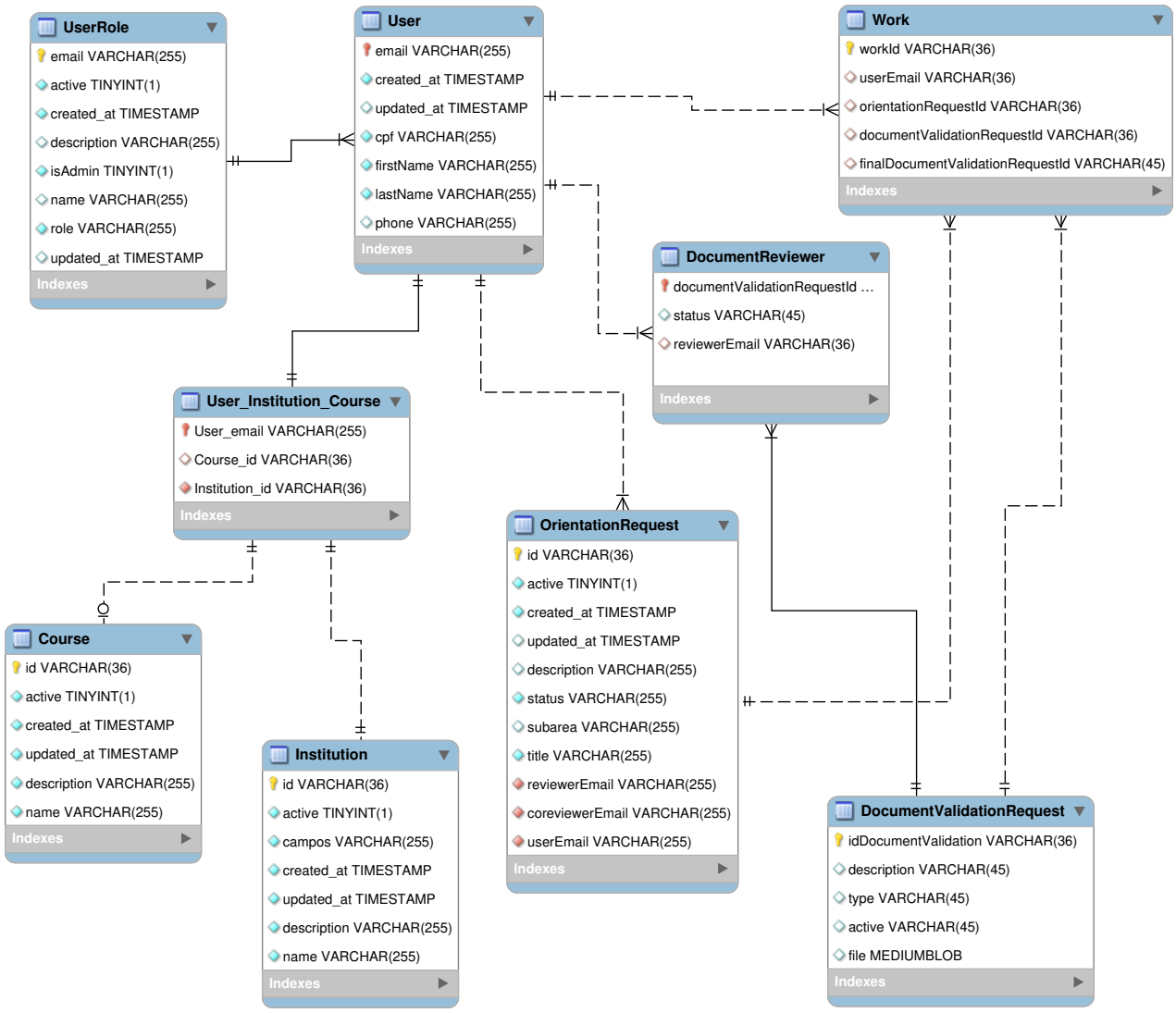
Para reduzir a complexidade no código e facilitar a manutenção, a arquitetura está sendo desenvolvida baseada no padrão MVC por ser um dos padrões mais utilizados atualmente e por possibilitar a divisão do projeto em camadas bem definidas [14].

Além disso, um dos pontos fortes de usar o padrão MVC é a garantia do isolamento das regras de negócios da interface com o usuário, possibilitando a existência de várias interfaces com o usuário que podem ser modificadas sem alterações nas regras de negócios sejam necessárias, oferecendo muito mais flexibilidade e oportunidades de reutilização de classes.

3.3 Diagrama Entidade Relacional

O diagrama apresentado na figura 5 criado com ajuda do *software* MySQL Workbench mostra como o banco de dados está sendo projetado. Esse é o plano inicial para o banco de dados, mas não se limita a isso. Em trabalhos futuros tabelas devem ser criadas para que seja possível enviar notificações e publicar trabalhos conforme solicitação do professor da disciplina.

Figura 5 – Diagrama Entidade Relacional



4 Descrição da API

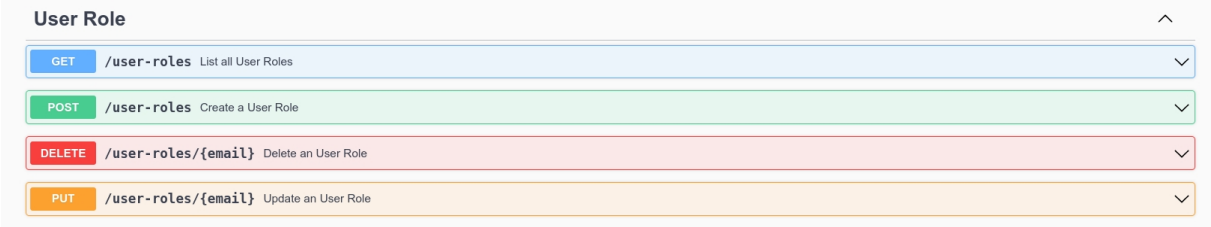
Durante o desenvolvimento do projeto, está sendo utilizado um conjunto de ferramentas que nos ajuda a construir o design da API chamado Swagger. Ele é responsável por auxiliar na modelagem e documentação de APIs.

Todas as rotas foram devidamente testadas através de testes manuais. Entretanto, não foram criados testes de integração, unitários e funcionais durante o desenvolvimento.

4.1 Rotas da Aplicação (Swagger API)

A figura 6 mostra as Permissões de usuário que são cadastradas por administradores. É possível definir os emails que poderão efetuar login na plataforma e associar esses emails a um "papel (do termo inglês role)"de usuário "Aluno"ou "Professor". Ainda é possível definir se o aluno terá permissões de administrador ou não.

Figura 6 – Rotas de Gerenciamento das Permissões de Usuários

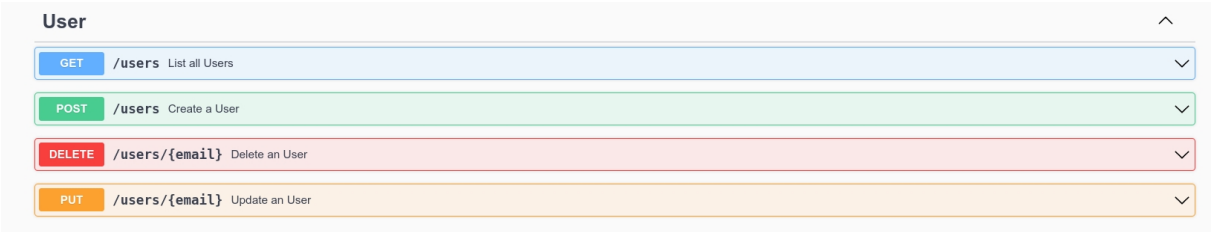


Os métodos de requisição definidos para as permissões de usuário são:

- *Get*: para listar todas as permissões cadastradas para os usuários;
- *Post*: para cadastrar uma nova permissão para um usuário;
- *Delete*: para deletar uma permissão de um usuário através do email;
- *Put*: para atualizar uma permissão de um usuário através email.

Os usuário (do termo inglês user) só podem ser criados se uma permissão de usuário já tiver sido criada para o mesmo. Quando um usuário acessa pela primeira vez a plataforma, o backend envia um documento como resposta para informar que o usuário tem permissão, mas ainda não foi criado. A figura 7 mostra as rotas criadas para o gerenciamento de usuários.

Figura 7 – Rotas de Gerenciamento de Usuários

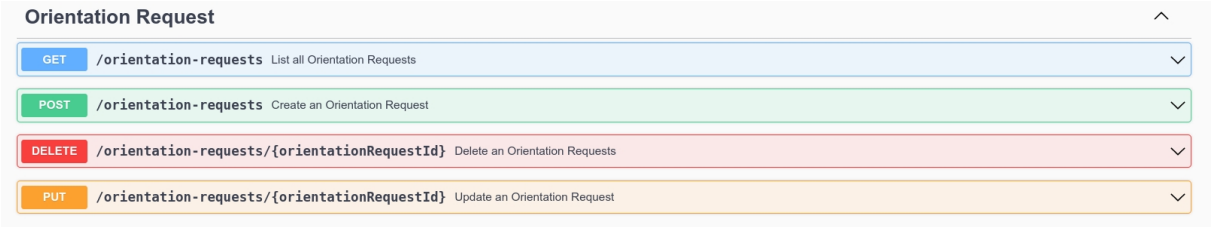


Os métodos de requisição definidos para os usuário são:

- *Get*: para listar todos os usuários cadastrados;
- *Post*: para cadastrar um novo usuário;
- *Delete*: para deletar um usuário através do seu email;
- *Put*: para atualizar um usuário através do seu email.

As "orientation requests", em português solicitações de orientação, podem ser criadas por alunos e aprovadas ou rejeitadas por professores. Os administradores podem desativar as solicitações, com isso elas só são exibidas na busca quando solicitadas. A figura 8 mostra as rotas de gerenciamento das solicitações de orientação.

Figura 8 – Rotas de Gerenciamento das Solicitações de Orientação

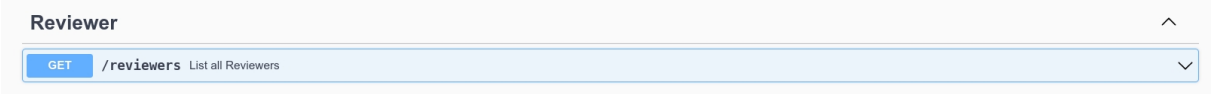


Os métodos de requisição definidos para as solicitações de orientação são:

- *Get*: para listar todas as solicitações de orientação;
- *Post*: para cadastrar uma nova solicitações de orientação;
- *Delete*: para deletar uma solicitações de orientação através do seu id;
- *Put*: para atualizar uma solicitações de orientação através do seu id.

Um serviço chamado "Reviewer"(Revisores, em português) foi criado para visualizar a lista de todos os orientadores, conforme mostra a figura 9.

Figura 9 – Rotas de Revisores

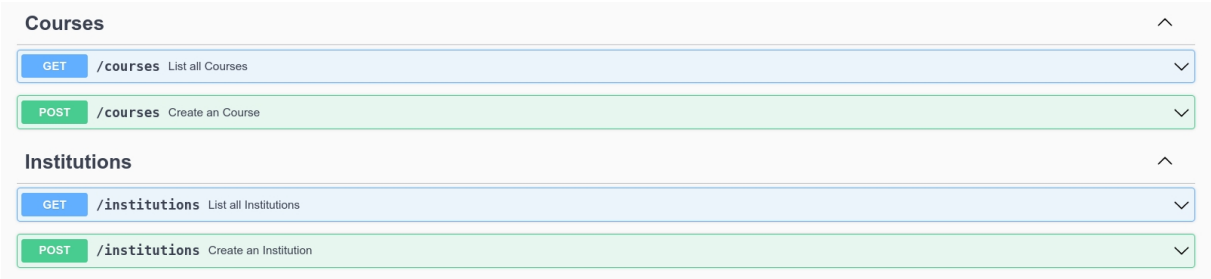


Apenas um métodos de requisição está definido para esse serviço:

- *Get*: para listar todos orientadores;

Para cadastrar e listar Cursos (Courses, em inglês) e Instituições (Institutions, em inglês) é possível utilizar as rotas mostradas na figura 10. As rotas para atualizar, deletar e conectar essas entidades não foram definidas, mas podem ser implementadas em trabalhos futuros.

Figura 10 – Rotas de Cursos e Instituições



Os métodos de requisição definidos para os cursos são:

- *Get*: para listar todas os cursos;
- *Post*: para cadastrar uma novo curso;

Os métodos de requisição definidos para as instituições são:

- *Get*: para listar todas as instituições;
- *Post*: para cadastrar uma nova instituições;

5 Resultados

Com a finalização deste projeto é fornecido uma API para o sistema de gestão dos Trabalhos de Conclusão de Curso do curso de Engenharia de Computação da UPE, englobando boas práticas de desenvolvimento, segurança e capaz de suprir as principais necessidades dos discentes e docentes incluídos no contexto da disciplina de TCC. Com a possibilidade de expansão do uso da API em outros trabalhos para que seja possível utilizar a API nas disciplinas de TCC de outros cursos ofertados na UPE.

Dessa maneira, espera-se que o sistema supra às principais necessidades do professor da disciplina quanto a gestão dos trabalhos e sirva como base para trabalhos futuros que venham a incrementar funcionalidades na aplicação ou pretendam desenvolver aplicações robustas e escaláveis. Sendo assim, o presente trabalho contribui na viabilização do processo de Trabalho de Conclusão de Curso do curso de Engenharia da Computação da Universidade de Pernambuco.

Os serviços de cursos e instituições estão incompletos, é necessário criar rotas para atualizar e deletar seus dados. Além disso, não há uma conexão estabelecida entre essas entidades. Em trabalhos futuros recomendo a criação de uma nova tabela para conectar os cursos, usuários e instituições.

6 Conclusões e Trabalhos Futuros

Neste trabalho está sendo descrito o desenvolvimento do *backend* da plataforma de gerenciamento de trabalhos de conclusão de curso, envolvendo a criação de casos de uso, requisitos de *software*, definição de arquitetura do *software* e detalhamento da API desenvolvida.

Este projeto auxilia o professor da disciplina na gestão dos processos, facilitando e unificando procedimentos das etapas referentes ao trabalho de conclusão de curso dos discentes do curso de Engenharia da Computação. Sendo assim, contribuindo com a UPE, discentes e docentes que venham a envolver-se na disciplina de conclusão de curso.

Além disso, a maneira como o *software* está sendo projetado permite que trabalhos futuros possam adicionar novas funcionalidades e expandir a API para que seja possível utilizá-la em outros cursos da UPE.

O presente trabalho está sendo desenvolvido para suprir as principais necessidades do professor da disciplina e servir como base para trabalhos futuros que venham a dar manutenção e/ou adicionar novas funcionalidades na plataforma. Devido a arquitetura e padrões utilizados, a maior parte do código pode ser reutilizado facilitando futuras alterações.

Referências

[1] ARRUDA, L. B. Sistema de gestão de trabalhos de final de curso. *Trabalhos de Final de Curso — Universidade Federal de Minas Gerais*, p. 26, 2010. Citado na página 12.

[2] ABMES. *Educação Superior Comentada | Políticas, diretrizes, legislação e normas do ensino superior*. [S.l.]. Disponível em: <https://abmes.org.br/colunas/detalhe/298/educacao-superior-comentada-politicas-diretrizes-legislacao-e-normas-do-ensino-superior#:~:text=O%20trabalho%20de%20gradua%C3%A7%C3%A3o%20ou,do%20Conselho%20Pleno%20(CP).&text=Ver%20Engenharia%20Agron%C3%B4mica.&text=Resolu%C3%A7%C3%A3o%20CES%2FCNE%20n%C2%BA%202,6%C2%BA%20e%209%C2%BA.>. Acesso em: 11 de agosto de 2022. Citado na página 12.

[3] UPE. *Projeto De Final De Curso*. [S.l.]. Disponível em: <https://drive.google.com/file/d/1ZCeRz8GY3aVJ9RYrwavvcH5wxDOp7wDS/view>. Acesso em: 13 de agosto de 2022. Citado na página 12.

[4] UNOCHAPECÓ, B. D. *Trabalho de Conclusão de Curso*. [S.l.], 2020. Disponível em: <https://www.unochapeco.edu.br/blog/entenda-a-importancia-do-tcc-para-o-aprendizado-e-a-vida-profissional#:~:text=O%20Trabalho%20de%20Conclusão%20de%20Curso%20é%20um%20momento%20importante,estudo%20mais%20dedicado%20e%20refinado>. Acesso em: 30 de julho de 2022. Citado na página 13.

[5] DEVMEDIA. *Introdução a Requisitos de Software*. [S.l.], 2013. Disponível em: <https://www.devmedia.com.br/introducao-a-requisitos-de-software/29580>. Acesso em: 11 de outubro de 2022. Citado na página 16.

[6] GONÇALVES, M. M. *Arquitetura de Software: Estilos e Padrões de Design*. [S.l.], 2021. Disponível em: <https://medium.com/@marcelomg21/arquitetura-de-software-estilos-e-padr%C3%B5es-de-design-50d62d684ef2>. Acesso em: 11 de outubro de 2022. Citado na página 21.

[7] TYPESCRIPT. *The starting point for learning TypeScript*. [S.l.]. Disponível em: <https://www.typescriptlang.org/docs/>. Acesso em: 01 de agosto de 2022. Citado na página 21.

[8] HERRON, D. *Quick Start to using Typescript and TypeORM on Node.js for CLI and web applications*. 4. ed. [S.l.: s.n.], 2022. 244 p. Citado na página 21.

[9] EXPRESS. *Express - framework de aplicativo da web Node.js*. [S.l.]. Disponível em: <http://expressjs.com/pt-br/>. Acesso em: 01 de agosto de 2022. Citado na página 21.

[10] NODEJS. *About Node.js*. [S.l.]. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 11 de agosto de 2022. Citado na página 21.

[11] TECNOLOGIA, H. *Por que usar MySQL? Saiba as razões*. [S.l.]. Disponível em: <https://www.hti.com.br/blog-mobile/543-por-que-usar-mysql-saiba-as-razoes#:~:text=Um%20MySQL %20%C3%A9%20essencial%20para,estejam%20bem %2Dorganizadas%20e%20seguras.>. Acesso em: 26 de setembro de 2022. Citado na página 21.

[12] FIREBASE. *Armazene e sincronize dados em tempo real*. [S.l.]. Disponível em: <https://firebase.google.com/products/realtime-database?hl=pt-br>. Acesso em: 26 de setembro de 2022. Citado na página 21.

[13] TECNOLOGIA, V. *O que é Docker?* [S.l.]. Disponível em: <https://vertigo.com.br/o-que-e-docker/>. Acesso em: 26 de setembro de 2022. Citado na página 21.

[14] DEVMEDIA. *Introdução ao Padrão MVC*. [S.l.], 2013. Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>. Acesso em: 04 de outubro de 2022. Citado 2 vezes nas páginas 22 e 23.

[15] DEVMEDIA. *Entendendo Coesão e Acoplamento*. [S.l.], 2020. Disponível em: <https://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538>. Acesso em: 04 de outubro de 2022. Citado na página 22.

[16] WAGON, L. *O que é padrão MVC? Entenda arquitetura de softwares!* [S.l.], 2020. Disponível em: <https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc>. Acesso em: 04 de outubro de 2022. Citado na página 22.

APÊNDICE A – Tutorial de Execução do Código

A.1 Acesso ao Código

O código pode ser acessado no repositório upe.tcc.api do professor co-orientador É necessário ele forneça acesso ao repositório (https://github.com/jbjj/upe.tcc.api) para ser possível acessá-lo.

A.2 Primeira Execução do Código

Na primeira execução do código, é necessário criar os containers do Docker e as tabelas no banco. Para isso, seguir os seguinte passos:

1. Ir a pasta raiz do projeto.
2. Executar o comando: *yarn start-docker*. Esse comando cria os containers e as tabelas no banco de dados. Além disso, após sua execução os container devem estar sendo executados em paralelo. Para verificar isso, executar o comando: *docker ps* no terminal.

A.3 Configuração do Firebase

É necessário que definir no código do projeto um serviço no Firebase. Para isso, siga os passos abaixo:

- Criar uma conta no Firebase.
- Criar um novo projeto.
- Vá para a seção de Autenticação e adicione login por "Email/Senha"(recomendado para o desenvolvimento) e "Google".
- Vá até as configurações do seu projeto no Firebase e clique na aba "Contas de Serviço"para gerar uma nova chave privada.
- Após clicar no botão para gerar a chave, um arquivo será baixado. Copie o código desse arquivo e sobrescreva o código do arquivo *src/config/serviceAccount.json* dentro do projeto upe.tcc.api.

- Para permitir que uma aplicação web acesse seu banco de dados, crie um app Web para seu projeto na seção Geral das configurações do Firebase.
- Ao criar o app, o próprio Firebase irá fornecer um tutorial sobre como configurar o ambiente da aplicação Web. Basta apenas seguir os passos do tutorial.

A.4 Configuração do Ambiente de Desenvolvimento

Para facilitar o desenvolvimento e os testes, uma configuração muito importante é criar um arquivo na raiz do projeto chamado *.env*. Nesse arquivo devem estar definidas as variáveis de ambiente seguindo o padrão: NOME_VARIAVEL=valor. As variáveis de ambiente que devem ser definidas são:

- ADMIN_EMAIL: definir um email válido.
- ADMIN_PASS: a senha deve exatamente a do usuário cadastrado com o email definido na propriedade anterior.
- ADMIN_UID: o uid deve exatamente a do usuário cadastrado com o email definido na propriedade anterior.
- ENV: essa propriedade define qual o ambiente do projeto. Os valores possíveis são: dev (desenvolvimento), test (testes) e prod (produção).
- SERVER_PORT: definição da porta para acessar a aplicação. Recomendado: 3333.
- CORS_ORIGIN: define quem pode acessar a API.
  Exemplo: *CORS_ORIGIN=http://localhost:3000*

Para definir e modificar variáveis de ambiente, modificar o arquivo: *src/types/environment.d.ts*.

A.5 Acesso a API

A aplicação é executada localmente na porta 3333, para acessar a API abra o seu *browser* e navegue até o seguinte link: <http://localhost:3333/api-docs>.

A.6 Criação de Migrations

Para criar uma nova migration, executar o comando: *yarn ts-node-dev -r tsconfig-paths/register ./node_modules/typeorm/cli.js migration:create ./src/shared/infra/typeorm/migrations/NomeDaClasse*.