INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO NORTE DE MINAS GERAIS
***Campus* Montes Claros**

<GraficHub>
Documento de Arquitetura de Software

Versão <1.0>

Histórico da Revisão

Data	Versão	Descrição	Autor
26/08/2024	1.0	Primeira Versão do Documento	Flavio Rodrigo Sil… Ivanderlei Mendes … Maicon Pedro Mac… Pedro Gabriel Dias …
20/09/2024	1.1	Versão final	Flavio Rodrigo Sil… Ivanderlei Mendes … Maicon Pedro Mac… Pedro Gabriel Dias …

Índice Analítico

1. Introdução....4
1.1 Finalidade....4
1.2 Escopo....4
1.3 Definições, Acrônimos e Abreviações....4
1.4 Referências....5
1.5 Visão Geral....5
2. Representação Arquitetural....5
3. Metas e Restrições da Arquitetura....5
4. Visão de Casos de Uso....6
4.1 UC06 - Gerir Portfólio....6
4.2 UC08 - Visualizar Feed....7
4.3 UC09 - Gerenciar Projeto....9
5. Visão Lógica....11
5.1 Visão Geral....11
5.1.1 app....11
5.1.2 config....11
5.1.3 database....11
5.1.4 public....11
5.1.5 resources....11
5.1.6 routes....11
5.1.7 storage....11
5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura....11
5.2.1 Controllers....11
5.2.2 Requests....14
5.2.3 Models....16
5.3 Diagrama de Classes....17
5.4 Diagrama de Sequência....18
5.4.1 UC06 - Gerir Portfólio....18
5.4.2 UC08 - Visualizar Feed....19
5.4.3 UC09 - Gerenciar Projeto....20
6. Visão de Processos....21
6.1 Diagrama de Atividades....21
6.1.1 UC06 - Gerir Portfólio....21
6.1.2 UC08 - Visualizar Feed....22

6.1.3 UC09 - Gerenciar Projeto....23
7. Visão de Implantação....24
8. Visão da Implementação....24
8.1 Visão Geral....24
8.2 Camadas....25
8.2.1 Camada de Apresentação....25
8.2.2 Camada de Negócios....25
8.2.3 Camada de persistência....25
8.2.4 Camada de banco de dados....25
9. Visão de Dados....26
10. Tamanho e Desempenho....26
11. Qualidade....27

Documento de Arquitetura de Software

1. Introdução

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.
O documento irá adotar uma estrutura baseada na visão "4+1" ilustrada na Figura 1.

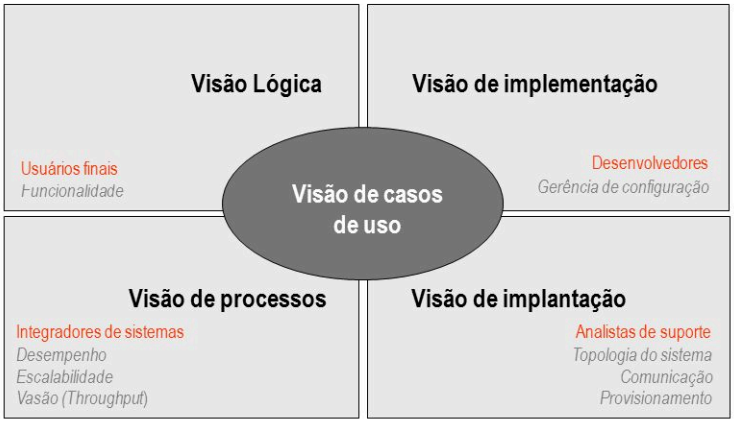


Figura 1 - Arquitetura 4+1

1.2 Escopo

O presente documento contempla a definição do modelo de arquitetura de software do *GraficHub*, que permite a organização e estruturação eficiente do sistema, além de proporcionar uma visão clara e abrangente, facilitando o entendimento e a comunicação entre os membros da equipe de desenvolvimento. Este documento serve como guia estratégico para o desenvolvimento e manutenção do sistema, influenciando diretamente nas tomadas de decisões técnicas, padrões utilizados no desenvolvimento, escalabilidade, manutenção, qualidade e comunicação.

1.3 Definições, Acrônimos e Abreviações

DAS - Documentos de Arquitetura de Software;

MVC - Modelo Visão Controlador (Model View Controller);

UI - Interface com Usuário;

UX - Experiência com Usuário;

SGBD - Sistema de Gerenciamento de Banco de Dados.

ORM - Sistema de mapeamento objeto-relacional do Laravel

SQL - Structured Query Language

1.4 Referências

KRUTCHEN, Philippe. *The "4+1" view model of software architecture.* Novembro de 1995. Disponível em: http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf. Acesso em: 30 ago. 2024.

SCIENCEDIRECT. *Layered Architecture.* Disponível em: https://www.sciencedirect.com/topics/computer-science/layered-architecture. Acesso em: 30 ago. 2024.

1.5 Visão Geral

Este documento contém uma visão detalhada e abrangente da estrutura e funcionamento do sistema, além das características arquiteturais selecionadas pela equipe de desenvolvimento para a solução em software do projeto *GraficHub.* Nele estão abordados os seguintes tópicos, respectivamente: Representação Arquitetural, Metas e Restrições de Arquitetura, Visão de Casos de Uso, Visão Lógica, Visão de Processos, Visão de Implantação, Visão da Implementação, Visão de Dados, Tamanho e Desempenho e Qualidade.

2. Representação Arquitetural

Para a implementação do projeto, foi adotado o framework Laravel, que segue a arquitetura MVC (*Model-View-Controller*). Essa abordagem organiza o código em três componentes principais: Modelos, Visões e Controladores. O *Model* é responsável pela interação com o banco de dados, a *View* cuida da interface e da exibição das informações (front-end), e o *Controller* atua como uma camada intermediária, facilitando o fluxo de dados entre o Model e a View.

Com isso, os próximos tópicos apresentam as visões baseadas no modelo "4+1".

Visão de Casos de Uso: Fornece informações essenciais para a implementação de cada funcionalidade do sistema, bem como detalhes sobre a interação entre os usuários (atores) e o sistema em cada funcionalidade.

Visão Lógica: Apresenta as funcionalidades do sistema, mostrando os diagramas de classes e descrevendo alguns atributos e funcionalidades das principais classes.

Visão de Processos: Explica os processos do sistema e como eles se comunicam por meio dos diagramas de sequência.

Visão de Implantação: Esclarece quais partes do software serão executadas em quais partes do hardware.

Visão da Implementação: O diagrama de implementação descreve a implementação física de informações geradas pelo programa de software em componentes de hardware.

Visão de Dados: É essencial para garantir a qualidade e a confiabilidade dos dados em um sistema. Ao compreender a estrutura e organização dos dados, é possível criar sistemas mais robustos e eficientes.

3. Metas e Restrições da Arquitetura

Requisitos	Solução
Linguagem	PHP e JavaScript
Framework	Laravel, Bootstrap e Tailwind CSS

Plataforma	Web
Portabilidade	A portabilidade do *GraficHub* é assegurada pela arquitetura baseada em tecnologias web padrão e pelo uso do framework Laravel, podendo ser acessado pelos navegadores modernos.
Eficiência	A eficiência do website será conceder os dados e suas informações de forma clara e rápida ao usuário, garantindo o gerenciamento e acompanhamento dos processos de forma imediata.
Segurança	A segurança é garantida através da implementação de autenticação e autorização via token em cada requisição. Dados sensíveis como senhas são criptografados em armazenamento. Além disso, segue práticas de segurança para proteger contra vulnerabilidades comuns como SQL injection e cross-site scripting (XSS), garantindo que o sistema seja resiliente contra ataques e mantenha a privacidade e integridade dos dados dos designers e clientes.

4. Visão de Casos de Uso

Nessa seção serão detalhados os casos de uso mais importantes e que englobam diversos aspectos do sistema. Além disso, é importante informar que todos os casos de uso do sistema estão listados e detalhados no Documento de Casos de Uso do Sistema. Dessa forma, os casos de uso que serão detalhados aqui são: UC06 - Gerir Portfólio, UC08 - Visualizar Feed e UC09 - Gerenciar Projeto.

4.1 UC06 - Gerir Portfólio

Caso de Uso	**UC06 - Gerir Portfólio**
Escopo	Sistema *GraficHub*
Ator Principal	Usuário
Interessados e Interesses	Usuário: Gerir projetos em seu portfólio de forma segura, rápida e efetiva.
Pré-Condições	O sistema deve estar online e acessível. O usuário deve possuir um dispositivo com um navegador compatível. O usuário deve estar autenticado no sistema para gerir seu portfólio.
Pós-Condições	O usuário visualiza os projetos postados em seu portfólio.
Cenário Principal de Sucesso	1. O sistema exibe ao usuário um botão de criar um projeto. Além disso, o sistema exibe os projetos do usuário contendo título, número de curtidas e favoritos. **(FA-01) (FA-02) (FA-03)** 2. O caso de uso finaliza com sucesso.

Fluxo Alternativo (extensões)	**FA-01 - O usuário clica no botão de criar um projeto** 1. Incluir caso de uso **UC11 - Criar projeto.** 2. O fluxo retorna ao passo 1 do FP. **FA-02 - O usuário clica em um projeto.** 1. Incluir **UC09 - Gerenciar Projeto.** 2. O fluxo retorna ao passo 2 do FP. **FA-03 - O usuário possui mais de doze projetos e clica em alguma página da barra de navegação.** 1. O sistema exibe os projetos relacionados àquela página. 2. O fluxo retorna ao passo 1 do FP.
Requisitos especiais	**RNF01 - Design responsivo** **RNF03 - Segurança do sistema** **RN02 - Número de projetos no perfil**
Diversos (problemas em aberto)	

4.2 UC08 - Visualizar Feed

Caso de Uso	**UC08 - Visualizar *Feed***
Escopo	Sistema *GraficHub*
Ator Principal	Usuário
Interessados e Interesses	Usuário: Visualizar o *Feed* de Projetos
Pré-Condições	O sistema deve estar online e acessível. O usuário deve possuir um dispositivo com um navegador compatível.
Pós-Condições	O usuário visualiza projetos de outros usuários em seu *feed*.
Cenário Principal de Sucesso	1. O sistema exibe dois botões “Descobrir” e “Seguindo” e logo em seguida os projetos da aba descobrir são apresentados. **(FA-01) (FA-02) (FA-03) (FA-04) (FE-01) (FE-02)** 2. O caso de uso finaliza com sucesso.
Fluxo Alternativo (extensões)	**FA-01 - O usuário clica em um projeto do feed.** 1. Incluir **UC10 - Visualizar Projeto.**

	2. O fluxo retorna ao passo 2 do FP. **FA-02 - O usuário clica no botão Seguindo.** 1. O feed exibe os projetos da aba "Seguindo". 2. O fluxo retorna ao passo 2 do FP. **FA-03 - O usuário clica no botão Descobrir.** 1. O feed exibe os projetos da aba "Descobrir". 2. O fluxo retorna ao passo 2 do FP. **FA-04 - O sistema possui mais de dezoito projetos para exibir e o usuário clica em alguma página da barra de navegação.** 1. O sistema exibe os projetos relacionados àquela página 2. O fluxo retorna ao passo 1 do FP. **FE-01 - O usuário não segue nenhum outro usuário e clica no botão "Seguindo".** 1. O sistema não exibe nenhum projeto. 2. O fluxo retorna ao passo 2 do FP. **FE-02 - O usuário não está autenticado e clica no botão "Seguindo".** 1. Incluir **UC03 - Autenticar no Sistema.** 2. O fluxo retorna ao passo 2 do FP.
Requisitos especiais	**RNF01 - Design responsivo** **RNF03 - Segurança do sistema** **RN03 - Número de projetos no feed**
Diversos (problemas em aberto)	

4.3 UC09 - Gerenciar Projeto

Caso de Uso	**UC09 - Gerenciar Projeto**
Escopo	Sistema *GraficHub*

Ator Principal	Usuário
Interessados e Interesses	Usuário: Gerenciar um projeto específico de forma rápida e responsiva.
Pré-Condições	O sistema deve estar online e acessível. O usuário deve possuir um dispositivo com um navegador compatível. O usuário deve estar autenticado no sistema para gerenciar um projeto.
Pós-Condições	O usuário gerencia o projeto específico da forma que desejar.
Cenário Principal de Sucesso	1. Incluir **UC01 - *Navbar.*** 2. O sistema exibe um projeto contendo: título, imagem da capa, botões de avançar e voltar imagens, ferramentas utilizadas, descrição, tags, botão de comentários, arquivo do projeto se este existir e logo abaixo dois botões “Editar” e “Excluir”. **(FA-01) (FA-02) (FA-03) (FA-04) (FA-05) (FA-06).** 3. Incluir **UC14 - Curtir Projeto**. 4. Incluir **UC22 - Favoritar Projeto**. 5. O caso de uso finaliza com sucesso.
Fluxo Alternativo (extensões)	**FA-01 - O usuário clica no botão “Editar”.** 1. Incluir **UC12 - Editar Projeto.** 2. O fluxo retorna ao passo 1 do FP. **FA-02 - O usuário clica no botão “Excluir”.** 1. Incluir **UC13 - Apagar Projeto.** 2. O fluxo retorna ao passo 5 do FP. **FA-03 - O usuário clica no botão do arquivo do projeto.** 1. O sistema realiza o download do arquivo na máquina do usuário. 2. O fluxo retorna ao passo 1 do FP. **FA-04 - O usuário clica no botão “Comentários”.** 1. Incluir **UC24 - Publicar Comentário**

	2. O fluxo de execução retorna ao passo 1 do FP. **FA-05 - O usuário clica no botão de avançar imagens.** 1. O sistema exibe a próxima imagem do projeto. 2. O fluxo retorna ao passo 1 do FP. **FA-06 - O usuário clica no botão de voltar imagens.** 1. O sistema exibe a imagem anterior a imagem atual do projeto. 2. O fluxo retorna ao passo 1 do FP.
Requisitos especiais	**RNF01 - Design responsivo** **RNF03 - Segurança do sistema**
Diversos (problemas em aberto)	

5. Visão Lógica

5.1 Visão Geral

Esta seção descreve a estrutura dos principais diretórios do Laravel.

5.1.1 app

Diretório que contém o código principal da aplicação. Ele contém as classes principais que compõem a lógica do aplicativo, como *models*, *controllers*, *middlewares* e outros componentes personalizados.

5.1.2 config

É o diretório onde ficam armazenados os arquivos de configuração do aplicativo. Esses arquivos contêm configurações que são usadas em todo o aplicativo, como conexões de banco de dados, opções de cache e serviços externos.

5.1.3 database

É onde ficam as migrações de banco de dados, *model, factories* e *seeds*.

5.1.4 public

O diretório *public* contém o arquivo *index.php* que é o ponto de entrada para todas as solicitações que entram no aplicativo e configuram o carregamento automático. Este diretório também abriga as imagens, JavaScript e CSS.

5.1.5 resources

É o diretório que contém suas visualizações (*views*), bem como os ativos brutos e não compilados, como CSS ou JavaScript.

5.1.6 routes

O diretório de rotas contém todas as definições de rota da aplicação.

5.1.7 storage

O diretório de armazenamento contém os logs, modelos Blade compilados, sessões baseadas em arquivo, caches de arquivo e outros arquivos gerados pelo framework.

E o diretório *storage/app/public* é usado para armazenar arquivos gerados pelo usuário, como sua foto de perfil, que devem ser acessíveis publicamente.

5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura

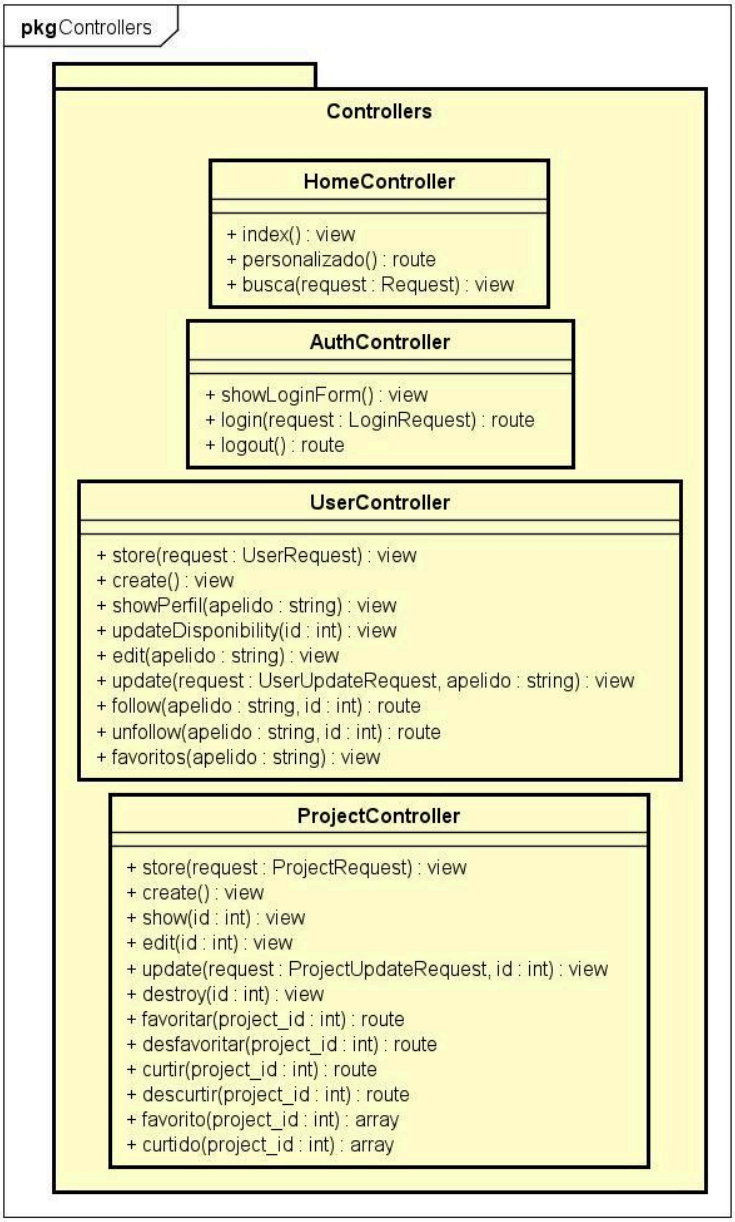
5.2.1 Controllers

Diretório: *App\Http\Controllers*

Os Controllers são responsáveis por manipular as solicitações HTTP e orquestrar a lógica da aplicação. Eles recebem as entradas do usuário, interagem com os Models para acessar e modificar dados e retornam a resposta apropriada para o usuário. Em resumo, os Controllers servem como intermediários entre as Views e os Models, gerenciando a lógica de aplicação e decidindo quais dados devem ser passados para as Views.

Classes significativas	**Descrição**
AuthController	Responsável pela autenticação no sistema.
HomeController	Responsável pelo *feed*.

ProjectController	Responsável pelo gerenciamento dos projetos.
UserController	Responsável pelo gerenciamento dos usuários.

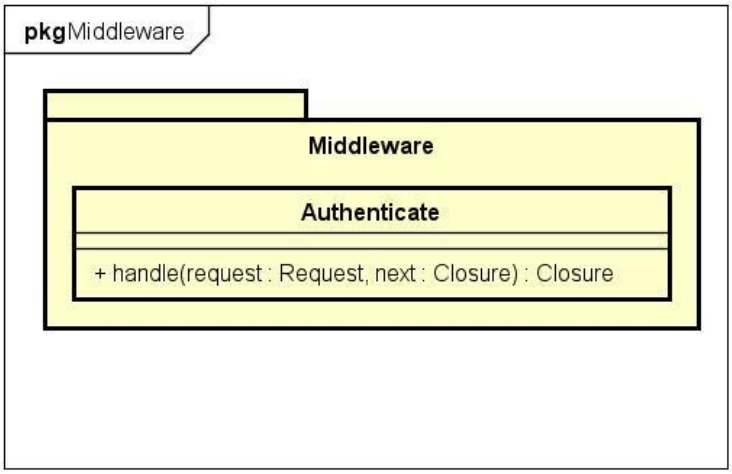


5.2.2 Middlewares

Diretório: *App\Http\Middleware*

O middleware é uma camada intermediária entre a requisição do usuário e a aplicação, responsável por executar uma lógica antes ou depois de uma solicitação HTTP ser processada pelo sistema.

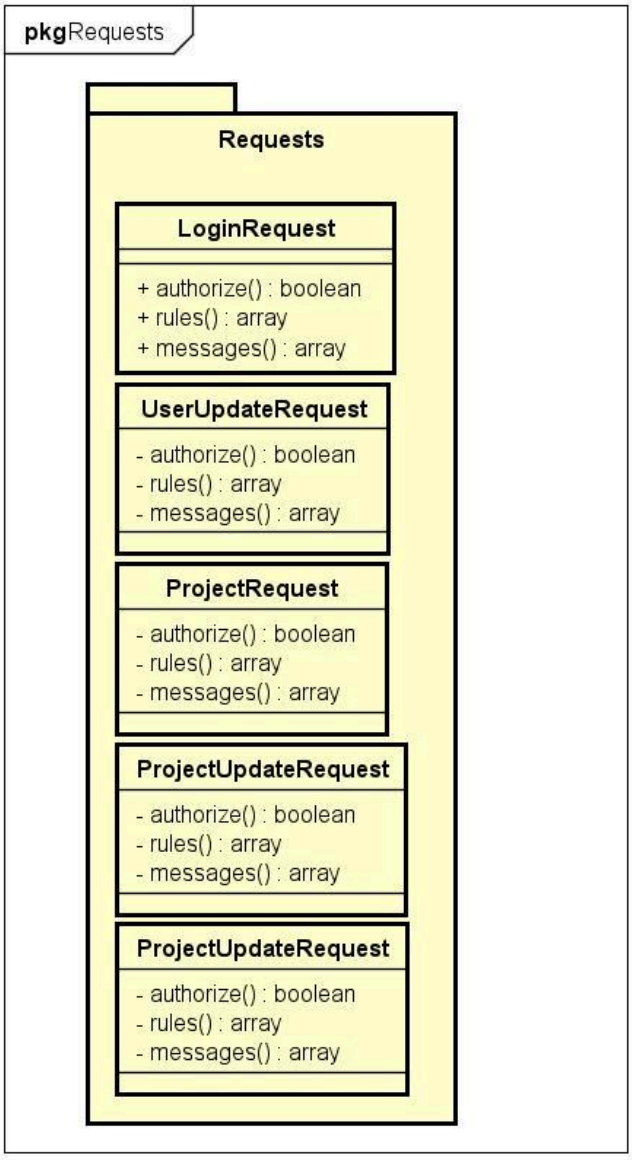
Classes significativas	**Descrição**
Authenticate	Responsável por verificar se o usuário está autenticado. Caso esteja, a requisição segue; caso contrário, o usuário é redirecionado para a página de login, e a requisição é retornada.



5.2.3 Requests

Diretório: *App\Http\Requests*

Embora o Laravel siga a arquitetura MVC, onde o model gerencia os dados, a lógica e as regras da aplicação, o framework permite desacoplar parte das regras de validação do model, delegando essa responsabilidade para as Requests. As Requests são responsáveis por validar os atributos e podem ser criadas para diferentes cenários de validação.



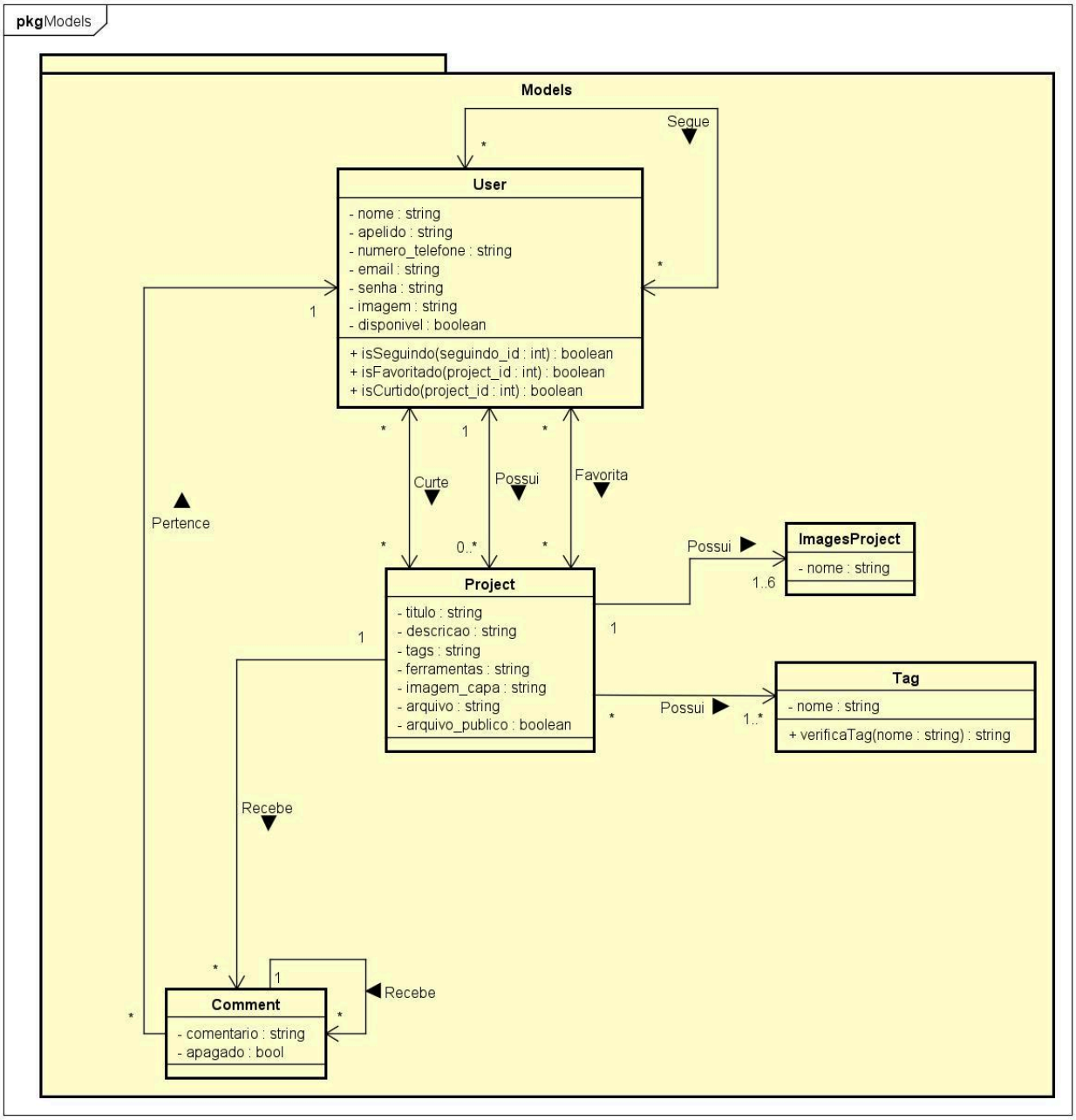
5.2.4 Models

Diretório: *App\Models*

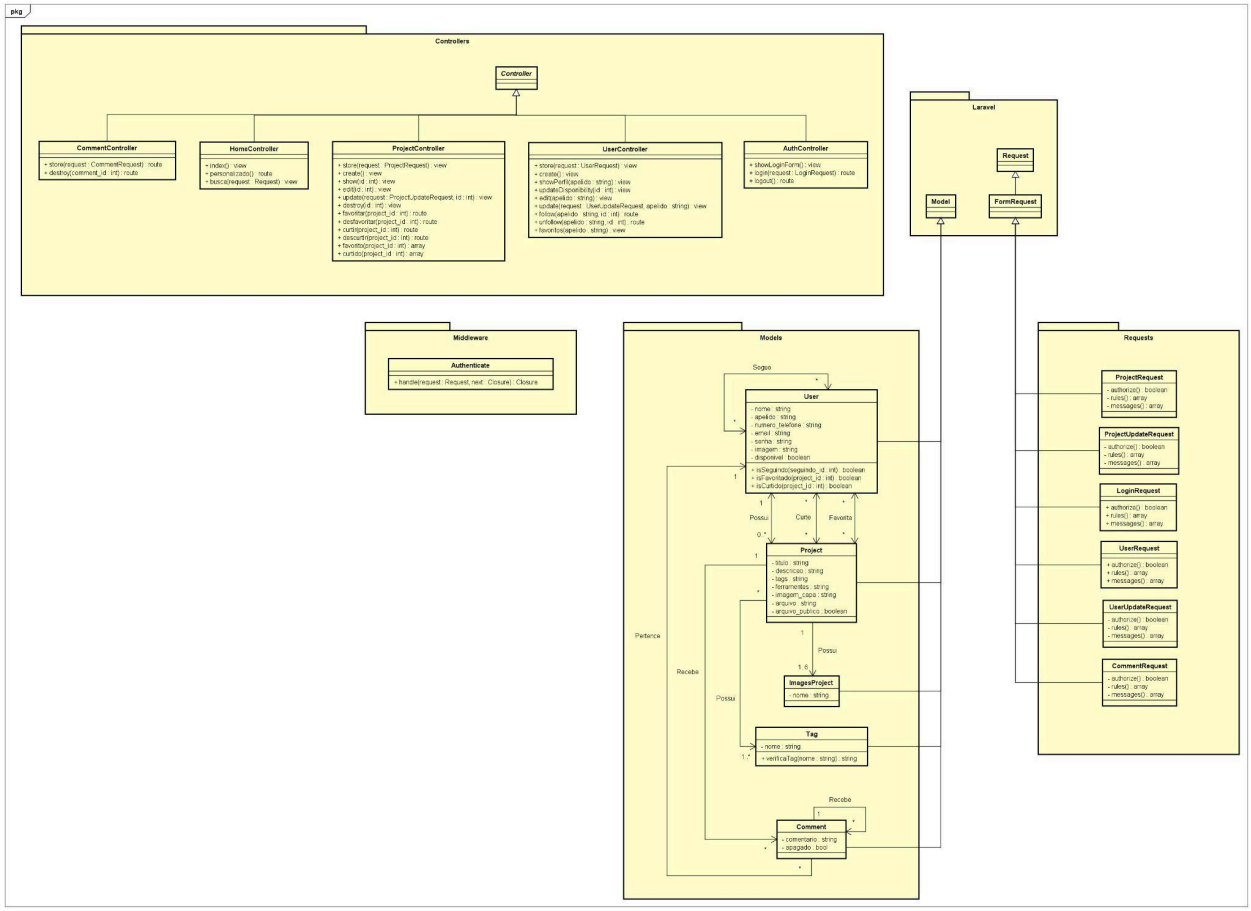
Os Models representam a estrutura dos dados e as interações com o banco de dados. Eles definem a estrutura das tabelas, as relações entre diferentes entidades e as operações que podem ser realizadas sobre esses dados. Os Models fornecem uma abstração para interagir com o banco de dados usando Eloquent, o ORM (Object-Relational Mapping) do Laravel, facilitando a criação, leitura, atualização e exclusão de registros no banco de dados.

Classes significativas	Descrição	Atributos importantes
User	Responsável por manter os dados	apelido, email e password

	do usuário no sistema e suas credenciais.	
Project	Responsável por manter os dados dos projetos de um usuário no sistema e suas credenciais.	titulo, imagem_capa

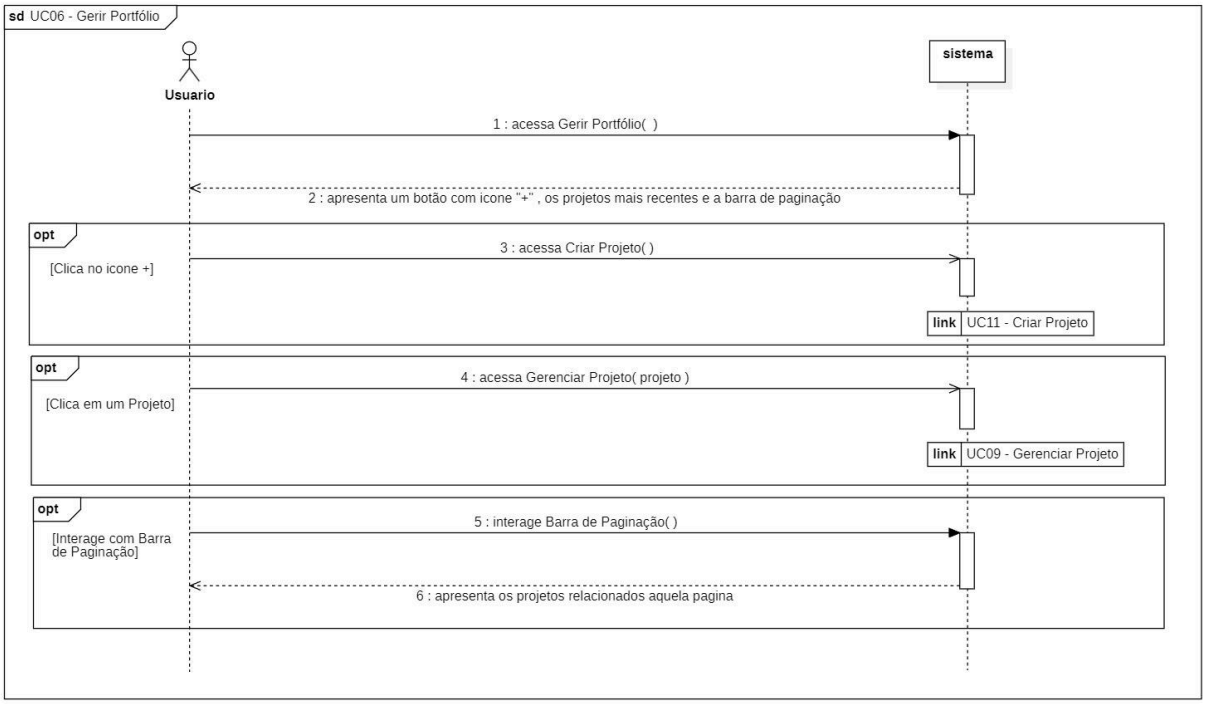


5.3 Diagrama de Classes

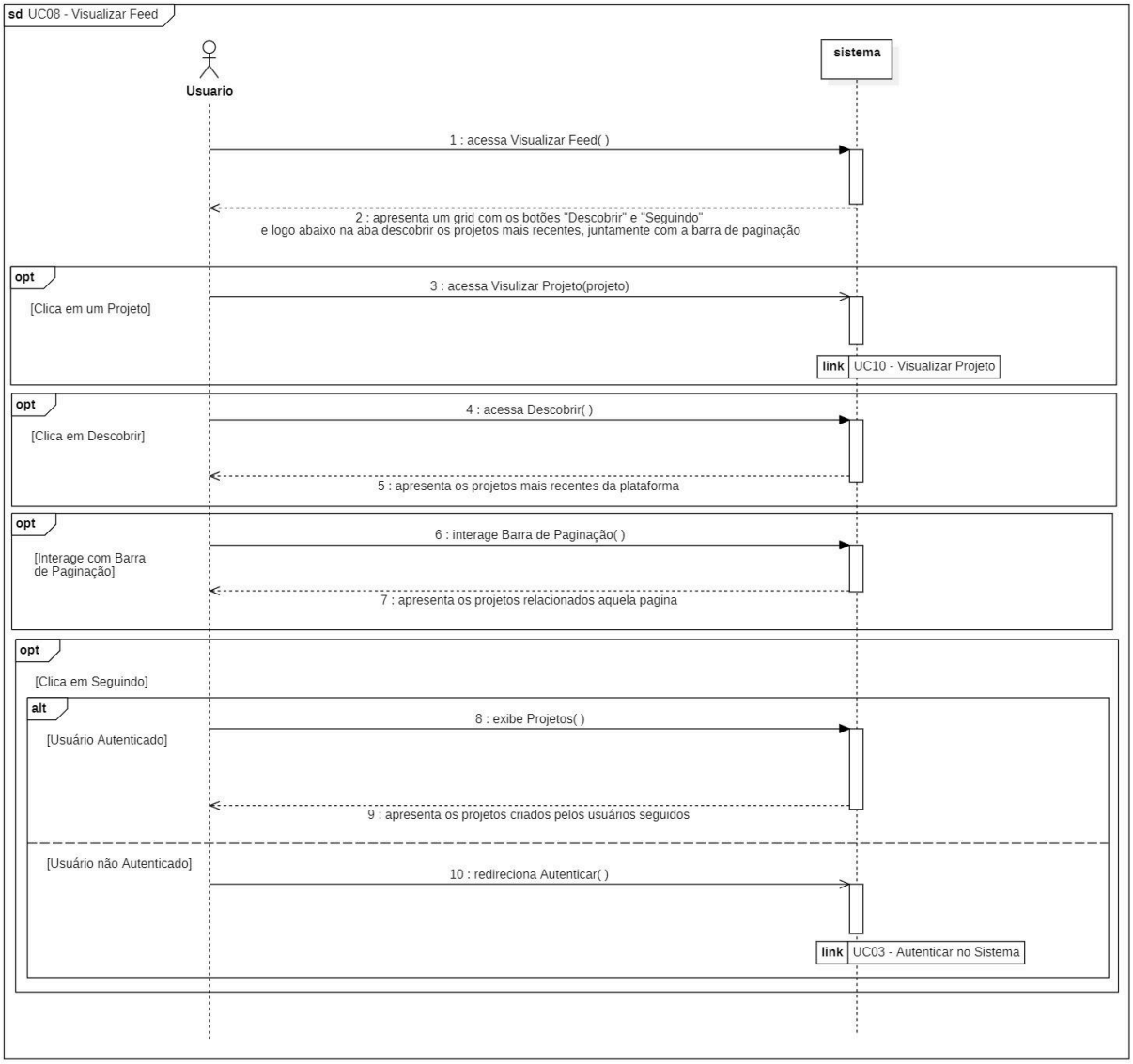

pkg
Controllers
Controller
CommentController
+ store(request : CommentRequest) : route
+ destroy(comment_id : int) : route
HomeController
+ index() : view
+ personalizado() : route
+ busca(request : Request) : view
ProjectController
+ store(request : ProjectRequest) : view
+ create() : view
+ show(id : int) : view
+ edit(id : int) : view
+ update(request : Project_UpdateRequest, id : int) : view
+ destroy(id : int) : view
+ favoritar(project_id : int) : route
+ desfavoritar(project_id : int) : route
+ curtir(project_id : int) : route
+ descurtir(project_id : int) : route
+ favoritos(project_id : int) : array
+ curtidos(project_id : int) : array
UserController
+ store(request : UserRequest) : view
+ create() : view
+ showPerfil(apelido : string) : view
+ updateDisponibility(id : int) : view
+ edit(apelido : string) : view
+ update(request : User_UpdateRequest apelido : string) : view
+ follow(apelido : string, id : int) : route
+ unfollow(apelido : string, id : int) : route
+ favoritos(apelido : string) : view
AuthController
+ showLoginForm() : view
+ login(request : LoginRequest) : route
+ logout() : route
Laravel
Request
Model
FormRequest
Middleware
Authenticate
+ handle(request : Request, next : Closure) : Closure
Models
Segue
User
- nome : string
- apelido : string
- numero_telefone : string
- email : string
- senha : string
- imagem : string
- disponivel : boolean
+ isSeguindo(seguindo_id : int) : boolean
+ isFavoritado(project_id : int) : boolean
+ isCurtido(project_id : int) : boolean
Possui
Curte
Favorita
Project
- titulo : string
- descricao : string
- tags : string
- ferramentas : string
- imagem_capa : string
- arquivo : string
- arquivo_publico : boolean
Pertence
Possui
Recebe
ImagesProject
- nome : string
Possui
Tag
- nome : string
+ verificaTag(nome : string) : string
Recebe
Comment
- comentario : string
- apagado : bool
Requests
ProjectRequest
- authorize() : boolean
- rules() : array
- messages() : array
ProjectUpdateRequest
- authorize() : boolean
- rules() : array
- messages() : array
LoginRequest
+ authorize() : boolean
+ rules() : array
+ messages() : array
UserRequest
+ authorize() : boolean
+ rules() : array
+ messages() : array
UserUpdateRequest
- authorize() : boolean
- rules() : array
- messages() : array
CommentRequest
- authorize() : boolean
- rules() : array
- messages() : array

5.4 Diagrama de Sequência

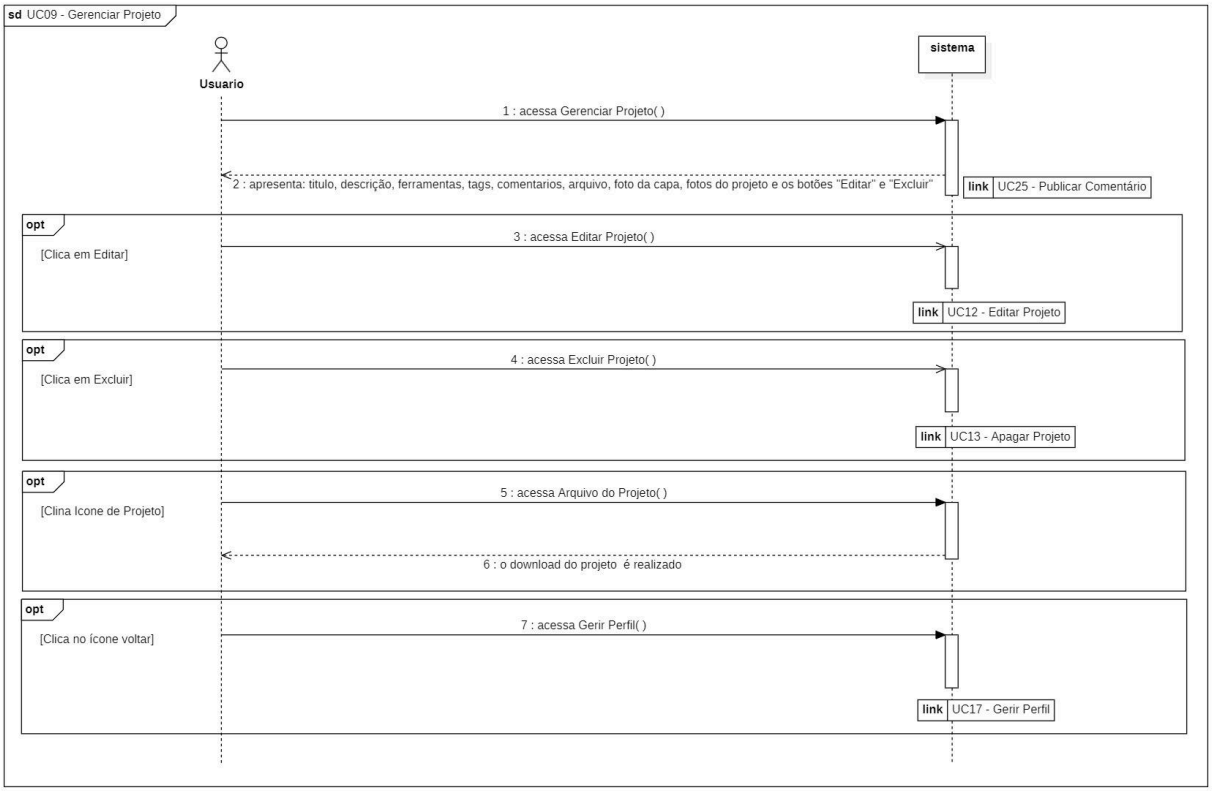
5.4.1 UC06 - Gerir Portfólio



5.4.2 UC08 - Visualizar Feed



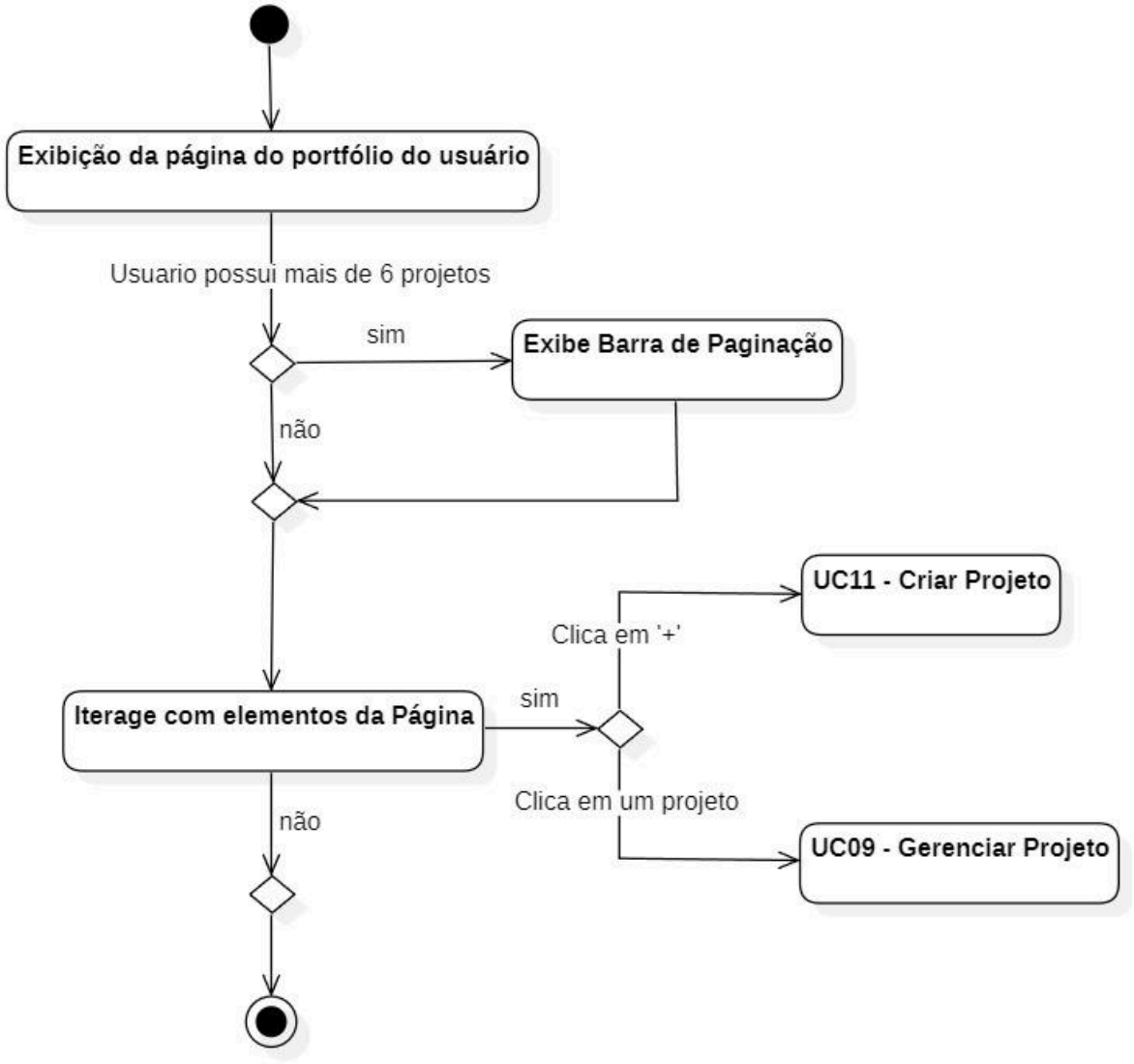
UC09 - Gerenciar Projeto



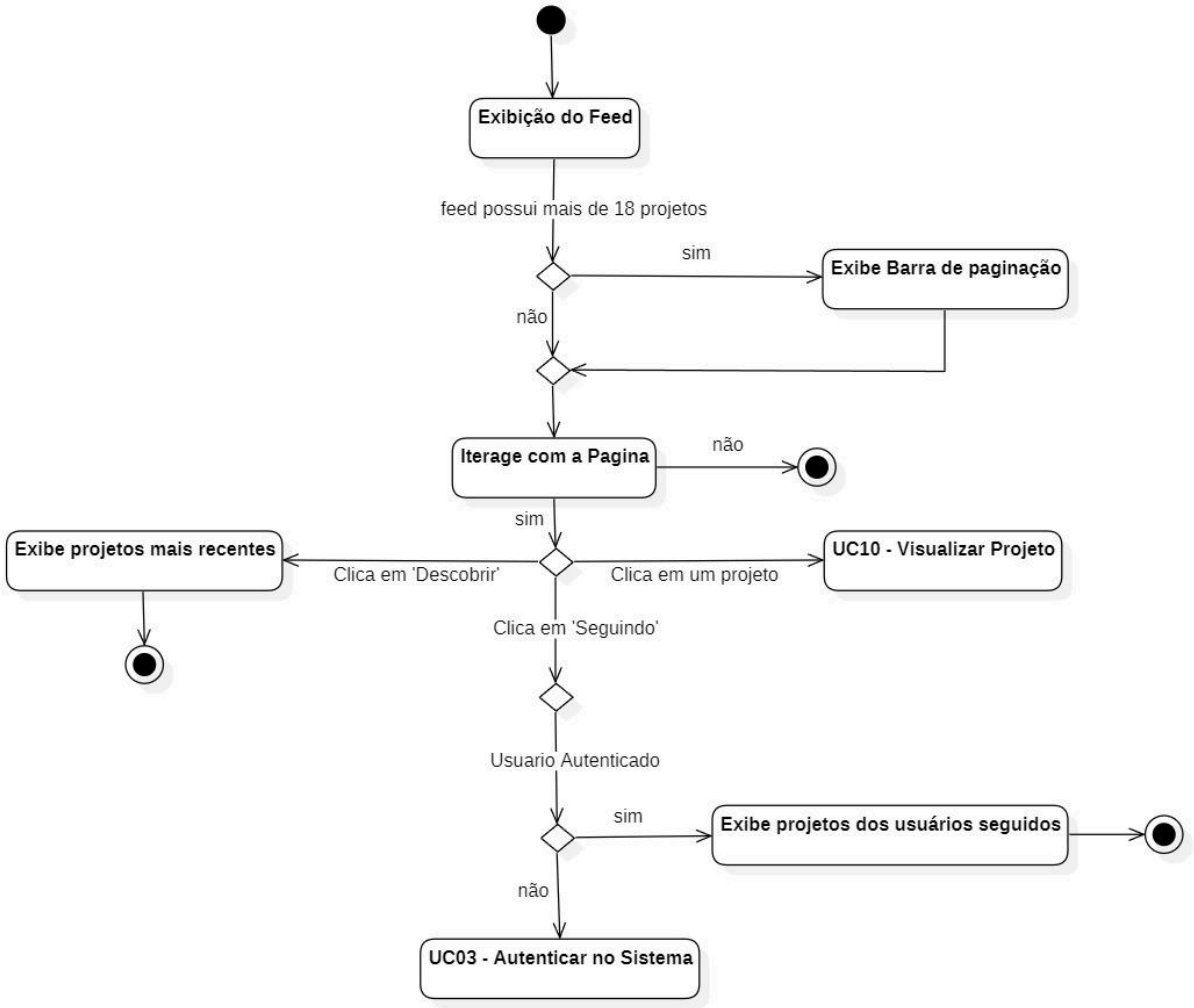
6. Visão de Processos

6.1 Diagrama de Atividades

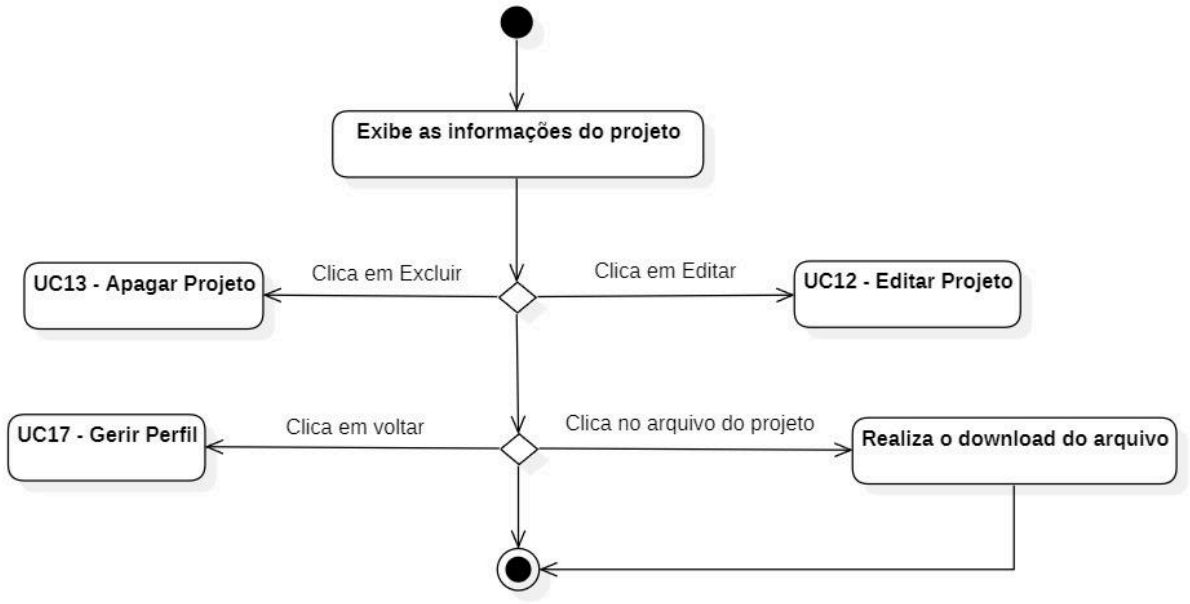
6.1.1 UC06 - Gerir Portfólio



6.1.2 UC08 - Visualizar Feed

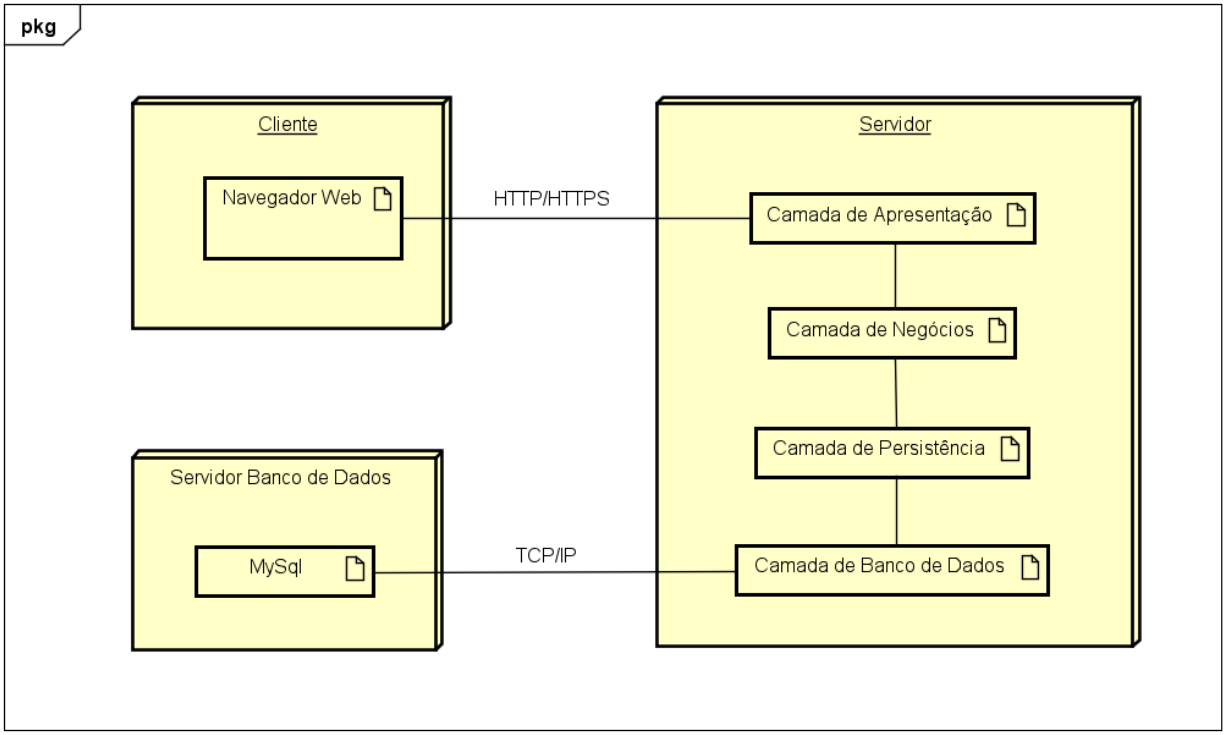


6.1.3 UC09 - Gerenciar Projeto



7. Visão de Implantação

O sistema é composto por um servidor que executa o MySql e um servidor que executa a aplicação. Esses dois servidores se comunicam para armazenar e recuperar dados e podem ser executados em uma mesma máquina física. A partir do momento que os dois servidores estão em execução o cliente pode acessar o sistema a partir de outra máquina física com acesso a internet através de requisições HTTP. Essas requisições são recebidas pela aplicação no servidor que as processa, acessa o servidor de dados caso necessário e envia de volta os resultados da requisição.



8. Visão da Implementação

O framework utilizado adota uma arquitetura baseada em MVC (Model-View-Controller), onde o código é organizado em três camadas principais:

Model: Responsável pela lógica de negócio e pela interação com o banco de dados.

View: Gerencia a apresentação dos dados, principalmente através de templates Blade.

Controller: Intermedia a comunicação entre o Model e a View, lidando com a lógica de controle.

8.1 Visão Geral

Principais camadas:

Camada de Apresentação

A camada de apresentação é responsável por exibir a interface do sistema aos usuários finais. Ela recebe as requisições dos usuários e as encaminha para a camada de negócios.. O Laravel utiliza o template Blade para criar as views

Camada de Negócios

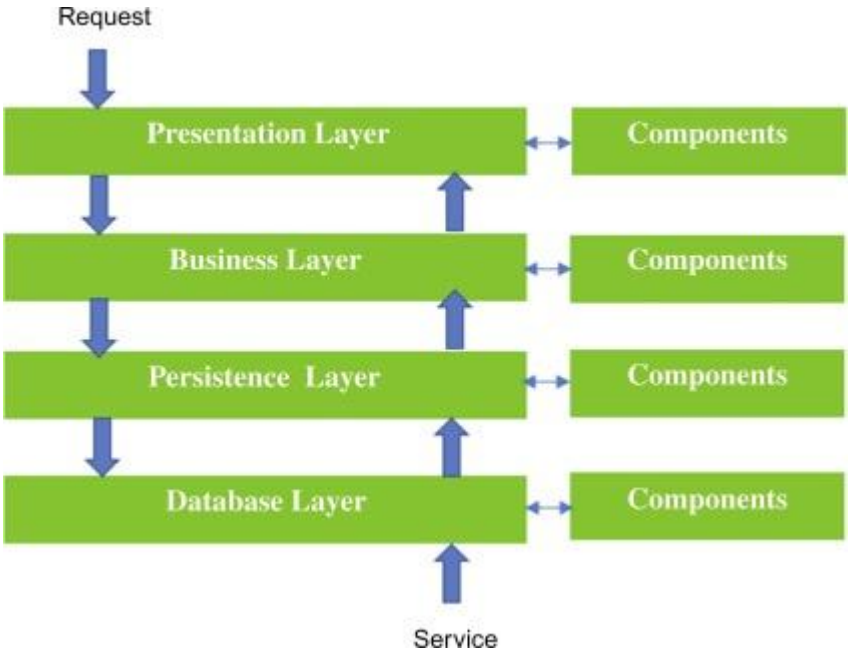
A camada de negócios contém a lógica de negócios do sistema. Ela processa as requisições recebidas da camada de apresentação, realiza validações, executa operações de negócio e interage com a camada de persistência. Isso é feito por meio dos Controllers, Requests e Middlewares.

Camada de persistência

A camada de persistência é responsável pela apresentação de dados. Ela inclui o objeto de acesso a dados, mapeamento relacional de objetos e outros modos de apresentação de dados persistentes no nível da aplicação. Para isso o framework usa o Eloquent.

Camada de banco de dados

A camada de banco de dados é responsável por armazenar dados da aplicação e recuperá-los. No framework são usadas Migrations para definir a estrutura do banco de dados, como criação e alteração de tabelas, colunas, índices, e chaves estrangeiras



8.2 Camadas

8.2.1 Camada de Apresentação

Templates Blade: auth, project, user e home

8.2.2 Camada de Negócios

Controllers: AuthController, UserController, ProjectController, HomeController e CommentController.

Request: LoginRequest, UserRequest, UserUpdateRequest, ProjectRequest, ProjectUpdateRequest.

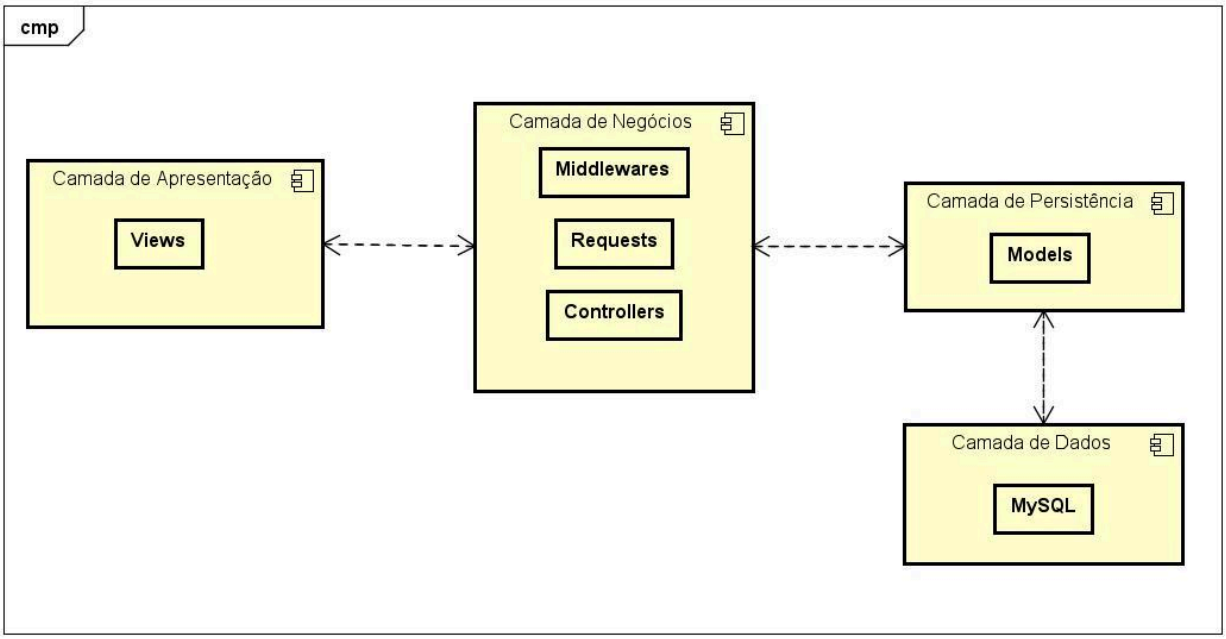
Middlewares: Authenticate.

8.2.3 Camada de persistência

Model (Eloquent): User, Project, ImagesProject, Comment, Tag.

8.2.4 Camada de banco de dados

Migrations: user, project, images_project e followers.



9. Visão de Dados

O *GraficHub* utiliza um Modelo de Dados Relacional, gerenciado pelo MySQL, um SGBD que utiliza a linguagem SQL como interface. A decisão de sua utilização ocorreu devido sua facilidade de uso, estabilidade, compatibilidade, controle de acesso e segurança de dados. Além disso, o MySQL possui integração nativa com o framework Laravel, simplificando a configuração de conexão ao banco de dados, utilização das migrations, além de facilitar a persistência dos dados pelo ORM Eloquent do Laravel. Sua principal função será armazenar e organizar informações essenciais, como perfis dos usuários, portfólios dos designers e as interações nos projetos e entre designers e clientes.

O volume de dados esperado é significativo devido ao armazenamento de imagens de alta resolução, arquivos de projetos e informações detalhadas dos projetos e dos usuários. Inicialmente, o armazenamento de arquivos e imagens é realizado localmente no servidor de hospedagem, isso facilita o acesso direto aos arquivos. Além disso, foi criado uma estrutura de diretórios projetada para ser eficiente e escalável, garantindo que os arquivos sejam armazenados de forma organizada e acessível.

10. Tamanho e Desempenho

O sistema é uma aplicação Web cujo os principais objetivos do *GraficHub* são o cadastro, gerenciamento e exibição de portfólios a fim de criar um ambiente de troca de informações entre designers e a possibilidade de atrair possíveis clientes. É esperado que a plataforma seja capaz de armazenar grandes volumes de dados, incluindo portfólios detalhados com imagens em alta resolução, arquivos do projeto e dados relacionados ao designer. Logo a arquitetura deve prever um banco de dados otimizado para suportar o volume de dados, além de utilizar estratégias de particionamento no upload e hospedagem dos dados. É previsto que o sistema seja capaz de suportar um grande número de usuários simultâneos, incluindo os designers que gerenciam seus portfólios e os clientes que navegam por esses portfólios.

A plataforma deve garantir tempos de resposta rápidos para ações comuns, como login, cadastro, busca e especialmente em operações críticas como a visualização de portfólios e carregamento de imagens. Como dito anteriormente a aplicação é baseada em uma plataforma Web, que será acessível e compatível com os principais navegadores do mercado. A arquitetura foi escolhida de forma que a aplicação tenha mecanismos de armazenamento, busca e visualização robustos e eficientes para que seja possível atender, de forma apropriada, até mesmo dispositivos eletrônicos de entrada com hardware e software menos eficientes. É importante ressaltar que a velocidade de conexão do usuário com a web e a disponibilidade do servidor têm grande efeito na experiência com a aplicação.

11. Qualidade

A arquitetura do *GraficHub* foi projetado para atender algumas metas, a fim de atender altos padrões de qualidade, como a facilidade no acréscimo de novas funcionalidades, manutenção e reaproveitamento do código. A utilização de padrões de projeto facilitam a implementação de novos comportamentos e funcionalidades no sistema, minimizando a necessidade de longas refatorações. O site inclui um sistema robusto e eficiente de autenticação e autorização, garantindo que haja a criptografia de dados sensíveis, como senhas, e que apenas usuários autenticados possam ter acesso a determinados recursos do sistema, como as informações de contato dos designers. Por fim, a aplicação oferece uma interface de usuário intuitiva e responsiva, permitindo uma melhor experiência ao usuário, além de que, os padrões de designer UI/UX são respeitados para garantir maior acessibilidade e facilidade de uso.