Projeto Arquitetural do OnColo

Histórico de Alterações

Data	Versão	Descrição	Autor(es)
14/03/2023	*1.0*	*Criação do documento*	*Arthur Matheus dos Santos Lima* *Any Caroliny Souza Silva* *Leticia Cena dos Santos* *Marcel Reinan Sa dos Santos* *Paulo Ricardo de Jesus Lima* *Rafael Vinicius Sousa*

Sumário

1. INTRODUÇÃO 4
1.1 Objetivo do projeto 4
1.2 Descrição e delimitação do problema 4
1.3 Glossário, convenções, termos e abreviações 4
2. STAKEHOLDERS 4
3. CONCERNS 5
3.1 Usabilidade 5
3.2 Confiabilidade 5
3.3 Segurança 5
3.4 Desempenho 5
3.5 Restrições estabelecidas 5
4. DECISÕES ARQUITETURAIS 5
4.1 Aplicações Web e Mobile 5
4.2 Web service 6
4.3 Arquitetura em camadas 6
4.4 Linguagens e ferramentas 7
4.5 Padronização 7
5. REPRESENTAÇÃO ARQUITETURAL 7
5.1 Visão de Cenários 8
5.1.1 Diagrama de casos de uso 8
5.1.2 Descrição dos casos de uso 9
5.2 Visão Lógica 13
5.2.1 Modelo de classes de domínio (modelo conceitual) 13
5.2.2 Diagrama de pacotes 13
5.3 Visão de Implementação (Desenvolvimento) 14
5.3.1 Diagrama de pacotes 14
5.3.2 Diagrama de classes 15
5.3.3 Diagramas de componentes 16
5.3.4 Diagrama ou Modelo Entidade Relacionamento 17
5.4 Visão de Processos 17
5.4.1 Diagramas de sequência ou comunicação 17
5.4.2 Diagramas de atividades. 18
5.5 Visão de Distribuição 19
5.5.1 Diagrama de Distribuição 19
6. REFERÊNCIAS BIBLIOGRÁFICAS 19

1. Introdução

1.1 Objetivo do projeto

O objetivo do OnColo é fornecer uma ferramenta para fisioterapeutas acompanharem as pacientes no pós-tratamento de câncer de colo de útero, registrando informações no banco de dados, por meio de formulários digitais. Como também na visualização de gráficos com base nos dados informados a respeito da evolução do tratamento da paciente.

1.2 Descrição e Delimitação do Problema

De acordo com o Instituto Nacional do Câncer (INCA) são esperados 704 mil casos novos de câncer no Brasil para cada ano do triênio de 2023 a 2025. Especificamente, o câncer do colo do útero ocupa a sexta posição entre os tipos mais frequentes de câncer e as mulheres, é o terceiro câncer mais incidente em geral. Sendo assim, surge uma demanda de coletar dados acerca desses casos de câncer de colo de útero, criando um histórico das pacientes que passam por tratamento oncológico, para que possam ser criadas estratégias de enfrentamento de acordo com o cenário no Brasil.

1.3 Glossário, convenções, termos e abreviações

- INCA - Instituto Nacional do Câncer
- APK - Android PAckage Kit
- HTTP - HyperText Transfer Protocol
- JSON - Javascript Object Notation
- Flutter - Kit de desenvolvimento de interface de usuário
- API - Application Programming Interface
- MVC - Model-View-Controller
- REST - Representational State Transfer
- Django - É um framework para desenvolvimento rápido para web, escrito em Python, que utiliza o padrão model-template-view(MVT)
- SQL - Linguagem de consulta estruturada

2. Stakeholders

- Tiago Plácido (PO)
- INCA
- Fisioterapeutas
- Pacientes de câncer de colo de útero

3. Concerns

3.1 Usabilidade

Considerando a usabilidade, tem-se que:

- O sistema deve ser desenvolvido para ser acessado via navegadores web;
- O sistema deve ser responsivo e a interface da aplicação deve ser intuitiva e fácil de usar;
- O sistema deve possuir uma versão mobile(APK);
- A aplicação deve se adaptar ao tamanho da tela do dispositivo no qual está sendo executada.

3.2 Confiabilidade

O sistema deverá ter uma alta disponibilidade, com possíveis “paradas” apenas para manutenção ou por problemas de caráter externo.

3.3 Segurança

No que diz respeito a segurança:

- O sistema não deverá fornecer informações de caráter privativo a outros usuários;
- O sistema deve ter medidas de segurança para garantir que somente fisioterapeutas autorizados possam acessar as informações registradas;
- A senha dos usuário deverá ser criptografada em hash antes de ser armazenada;
- Convém ressaltar que a aplicação deverá tratar os dados pessoais conforme a Lei Geral de Proteção de Dados Pessoais(LGPD), nº 13.709/2018.

3.4 Desempenho

A aplicação não consumirá uma quantidade de recursos além da necessária.

3.5 Restrições Estabelecidas

O sistema necessitará que o aparelho do usuário possua acesso à internet.

4. Decisões Arquiteturais

4.1 Aplicação Web e Mobile

A opção por essa arquitetura foi devido a necessidade de criar uma aplicação mobile e web, consequentemente, acessível a partir de qualquer dispositivo móvel ou computador que tenha acesso à internet. Como justificativa, temos que a aplicação deve dar suporte ao gerenciamento de fisioterapeutas por meio das instituições e ao gerenciamento das consultas, pacientes e formulários por

meio do próprio fisioterapeuta, tendo em vista que as funcionalidades devem estar disponíveis de qualquer lugar e a qualquer momento.

4.2 Web Service

O projeto inclui um web service para permitir a comunicação entre diferentes sistemas e plataformas. Isso possibilita que tanto o sistema web quanto o mobile se comuniquem com o servidor através de requisições HTTP (HyperText Transfer Protocol), utilizando o formato Javascript Object Notation (JSON). Também permite, caso necessário, que o sistema seja facilmente integrado com outro sistemas externos.

4.3 Arquitetura em Camadas

A arquitetura do nosso sistema segue a abordagem da arquitetura em três camadas: apresentação, aplicação e dados.

Na camada de apresentação, desenvolvemos o frontend da aplicação utilizando a tecnologia Flutter. Ela é responsável por apresentar a interface e receber as interações dos fisioterapeutas e instituições. A comunicação com a camada de aplicação é feita através do protocolo HTTP.

A camada de aplicação contém a API desenvolvida em Django, utilizando a arquitetura MVT (Model-View-Template). A camada é responsável por processar as informações da camada anterior.

1. A camada Model representa a camada de dados da aplicação. No Django, os modelos são definidos como classes que descrevem os campos e os comportamentos dos objetos que serão armazenados no banco de dados, que no caso da nossa aplicação são as classes Fisioterapeuta, Instituição e Paciente, eles podem ser acessados e manipulados por meio de consultas geradas pelo Django em SQL.
2. A camada View, por sua vez, é responsável por processar as solicitações recebidas do cliente e fornecer uma resposta. A View acessa os dados por meio dos modelos e aplica as regras de negócio da aplicação para fornecer a resposta.
3. A camada Template é usada para definir a apresentação da página final que será exibida ao usuário, mas não está sendo utilizada na nossa aplicação, pois a apresentação está sendo feita pelo Flutter.

Para a comunicação entre a camada de aplicação e a camada de persistência, utilizamos o protocolo SQL, que permite a execução de operações no banco de dados de forma eficiente.

Por fim, a camada de dados, nela temos o banco de dados SQL utilizado na nossa aplicação, responsável por armazenar os dados de forma segura e eficiente.

4.4 Linguagens e Ferramentas

No backend:

- Linguagem Python;
- Django REST framework;
- Banco de dados MySQL versão 8.0.32.0.

No frontend:

- Framework Flutter;
- Linguagem Dart;
- Ferramenta de documentação - Dartdoc.

4.5 Padronização

Para garantir a qualidade e a uniformidade do desenvolvimento de software no nosso projeto, utilizamos padronizações tanto no backend quanto no frontend, além de na comunicação entre as camadas do sistema.

No backend, utilizamos o framework Django e sua arquitetura MVT, seguindo as convenções de nomenclatura e as boas práticas recomendadas pela comunidade Django (PEP8). Utilizamos "querysets" para acesso ao banco de dados e separamos claramente as responsabilidades entre as camadas Model e View.

No frontend, utilizamos nomes de classe padrão CamelCase sugerido pelo framework utilizado (Flutter). Os nomes de arquivos foram definidos com uso de letras minúsculas, separação por _, e final do nome terminando com web ou mobile. No desenvolvimento no GitHub, 3 branches foram criadas cada uma no respectivo nome dos 3 membros do frontend. Todas as branches iniciaram a partir da mesma estrutura de pastas e cada membro ia fazendo commits e push gradual das suas funcionalidades. Logo após essa etapa concluída foi realizado reunião geral para merge das branchs na main e correção de conflitos.

5. Representação Arquitetural

A arquitetura é representada por várias visualizações arquitetônicas diferentes, que em sua essência, são extrações que ilustram os elementos “arquiteturalmente significativos” dos modelos. Para isso é necessário representações que consigam abordar diversas visões e um dos principais modelos utilizados é modelo “4+1”, pois consegue atender a abordagens grandes e desafiadoras. Um modelo de arquitetura é algo que está em alto nível, que busca atender aos requisitos funcionais e não funcionais. As 5 diferentes visões que compõem o modelo “4+1” são:

- Visão de Cenários;
- Visão Lógica;
- Visão de Implementação(Desenvolvimento);
- Visão de Processos;

- Visão de Distribuição(Física).

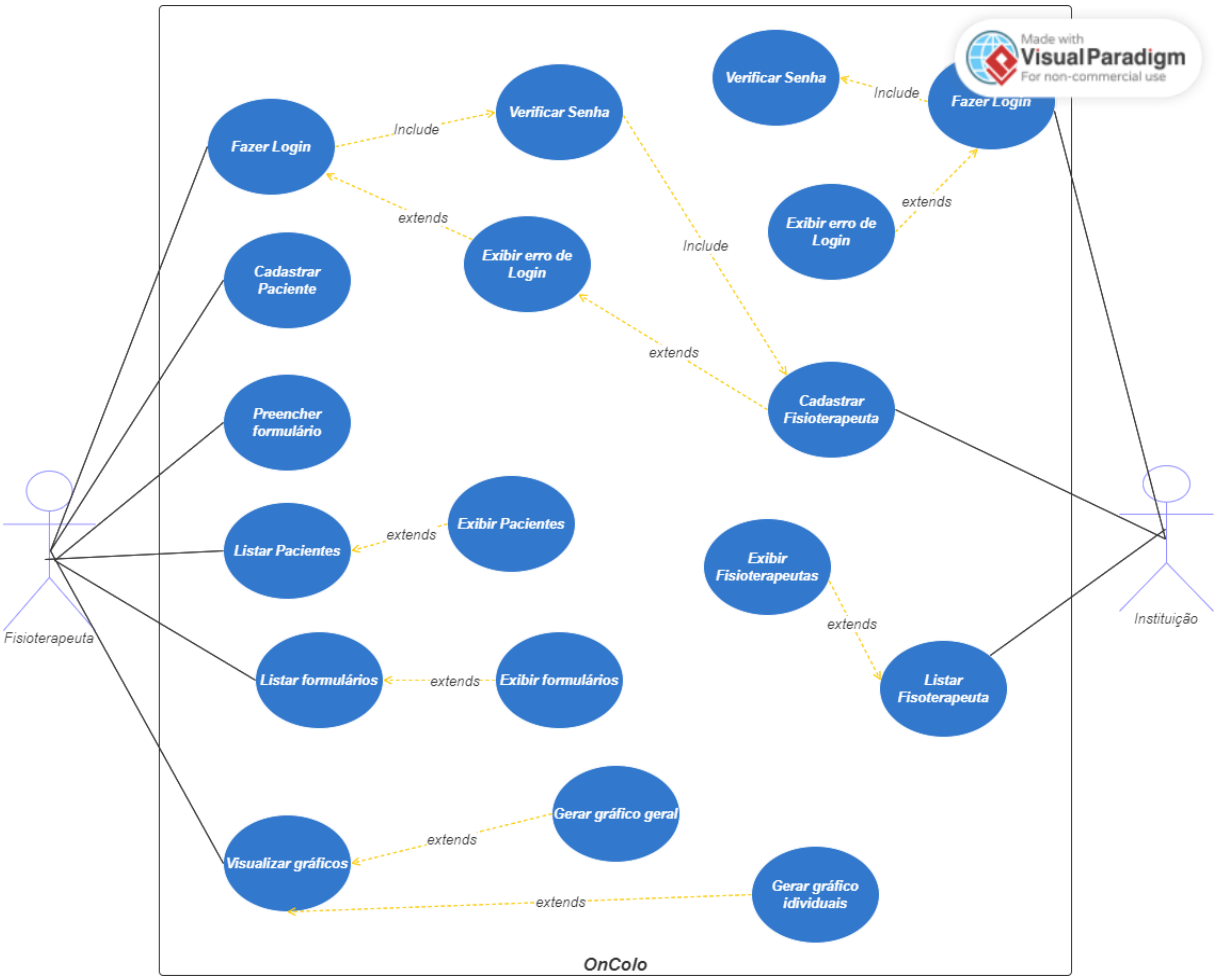
Cada uma das visões é descrita por um modelo que utiliza a sua própria notação particular. Cabe ao arquiteto decidir qual dos vários estilos existentes para usar em cada visão.

5.1 Visão de Cenários

Os cenários, é o que dá liga aos elementos das outras 4 visões, uma vez que foram pensados para trabalharem de forma conjunta, com sequências de interações entre os objetos e processos, sendo uma abstração de requisitos mais importantes. É um ponto de vista redundante dos outros mas que serve à dois propósitos: como guia para descobrir os elementos arquitetônicos durante o projeto arquitetural, e como papel de validação e ilustração quando o projeto de arquitetura é completado. Serve também como ponto de partida para os testes de um protótipo de arquitetura.

5.1.1 Diagrama de casos de uso

O diagrama de caso de uso descreve a funcionalidade proposta para um novo sistema que será projetado, é uma excelente ferramenta para o levantamento dos requisitos funcionais do sistema. Um caso de uso representa uma unidade discreta da interação entre um ator e o sistema. Para o caso do nosso sistema OnColo, possui dois atores, fisioterapeuta e instituição.



5.1.2 Descrição dos casos de uso

CSU01 – Cadastrar paciente

Ator	Fisioterapeuta
Gatilho	O fisioterapeuta deseja cadastrar novo paciente
Pré-condição	Fisioterapeuta autenticado
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. O fisioterapeuta informa os dados da paciente	
2. O sistema valida os dados fornecidos	
3. O sistema armazena os dados no banco	

CSU02 - Preencher formulário

Ator	Fisioterapeuta
Gatilho	O fisioterapeuta deseja preencher um novo formulário para paciente
Pré-condição	Fisioterapeuta autenticado
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. O sistema apresenta os formulários	
2. O fisioterapeuta escolhe o formulário	
3. O sistema apresenta as pacientes	
4. O fisioterapeuta escolhe a paciente	
5. O fisioterapeuta preenche o formulário	

CSU03 - Visualizar listagem pacientes

Ator	Fisioterapeuta
Gatilho	O fisioterapeuta deseja visualizar todas as pacientes cadastradas
Pré-condição	Fisioterapeuta autenticado
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. O fisioterapeuta solicita a listagem de pacientes	
2. O sistema exibe a lista com todas as pacientes	
3. O fisioterapeuta escolhe a paciente que deseja selecionar	
4. O sistema exibe informações sobre a paciente selecionada	

CSU04 - Visualizar listagem formulários

Ator	Fisioterapeuta
Gatilho	O fisioterapeuta deseja visualizar todos os formulários cadastradas
Pré-condição	Fisioterapeuta autenticado
Cenário Principal de Sucesso (Sequência típica de eventos)	

1. O fisioterapeuta solicita a listagem de formulários
2. O sistema exibe a lista com todos os formulários
3. O fisioterapeuta escolhe o formulário que deseja visualizar
4. O sistema exibe informações sobre o formulário selecionada

CSU05 - Visualizar gráficos

Ator	Fisioterapeuta
Gatilho	Fisioterapeuta deseja ter uma visão da desenvolvimento das pacientes
Pré-condição	Fisioterapeuta autenticado
Nível	Extensão
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. O fisioterapeuta solicita a visualização	
2. O fisioterapeuta indica o tipo de visualização (individual ou coletiva)	
3. Escolhe o tipo de formulário que vai gerar o gráfico	
4. O sistema exibe os gráficos solicitados	
Fluxos Secundários	
1. Visualizar cenário individual da paciente	
2. Visualizar cenário coletivo de todas as pacientes cadastradas	

CSU06 - Fazer Login

Ator	Fisioterapeuta
Gatilho	O fisioterapeuta deseja realizar o login no sistema
Pré-condição	Verificar se fisioterapeuta já foi cadastrado
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. O fisioterapeuta informa login e senha	
2. O sistema permite a entrada do fisioterapeuta	

CSU07 – Cadastrar fisioterapeuta

Ator	Instituição
Gatilho	A instituição deseja cadastrar novo fisioterapeuta
Pré-condição	Instituição autenticada
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. A instituição informa os dados do fisioterapeuta	
2. O sistema valida os dados fornecidos	
3. O sistema armazena os dados no banco	

CSU08 - Visualizar listagem fisioterapeutas

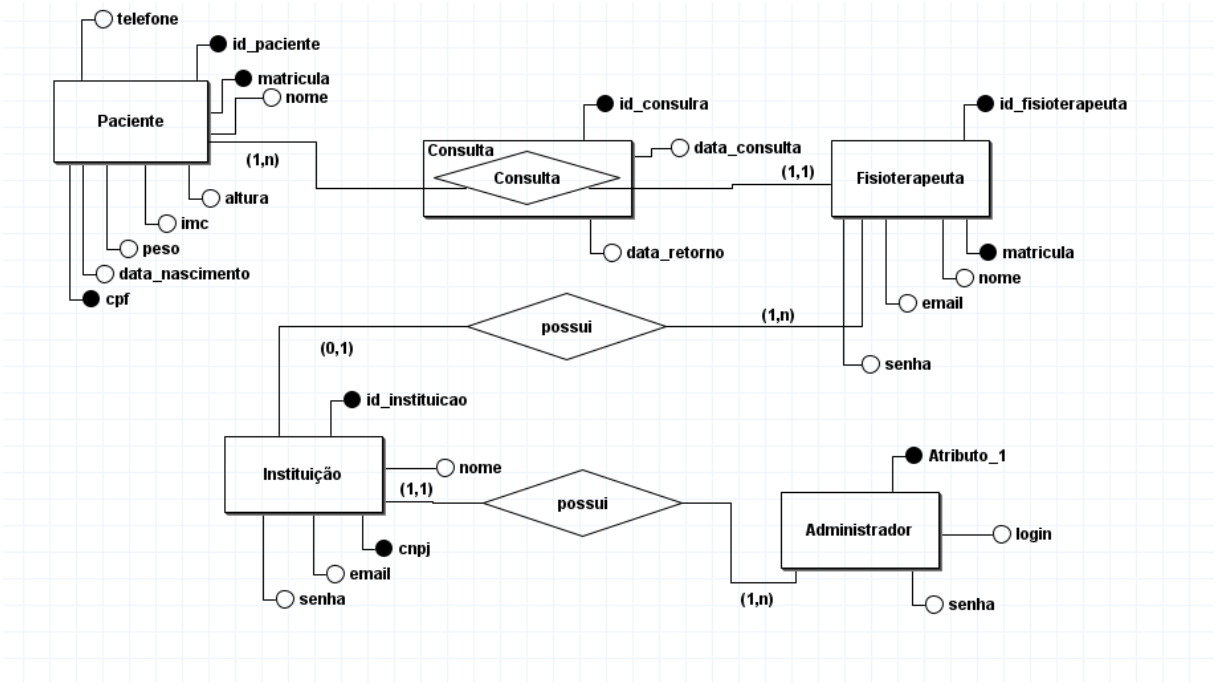
Ator	Instituição
Gatilho	A instituição deseja visualizar todos os fisioterapeutas cadastrados
Pré-condição	Instituição autenticada
Cenário Principal de Sucesso (Sequência típica de eventos)	
1. A instituição solicita a listagem de fisioterapeutas	
2. O sistema exibe a lista com todas os fisioterapeutas	
3. A instituição escolhe o fisioterapeuta que deseja selecionar	
4. O sistema exibe informações sobre o fisioterapeuta selecionado	

5.2 Visão Lógica

É um modelo de arquitetura de software que tem como objetivo a decomposição do software orientado a objetos. Ela oferece suporte primário para requisitos funcionais, ou seja, o que o sistema precisa fornecer em termos de serviços para os usuários.

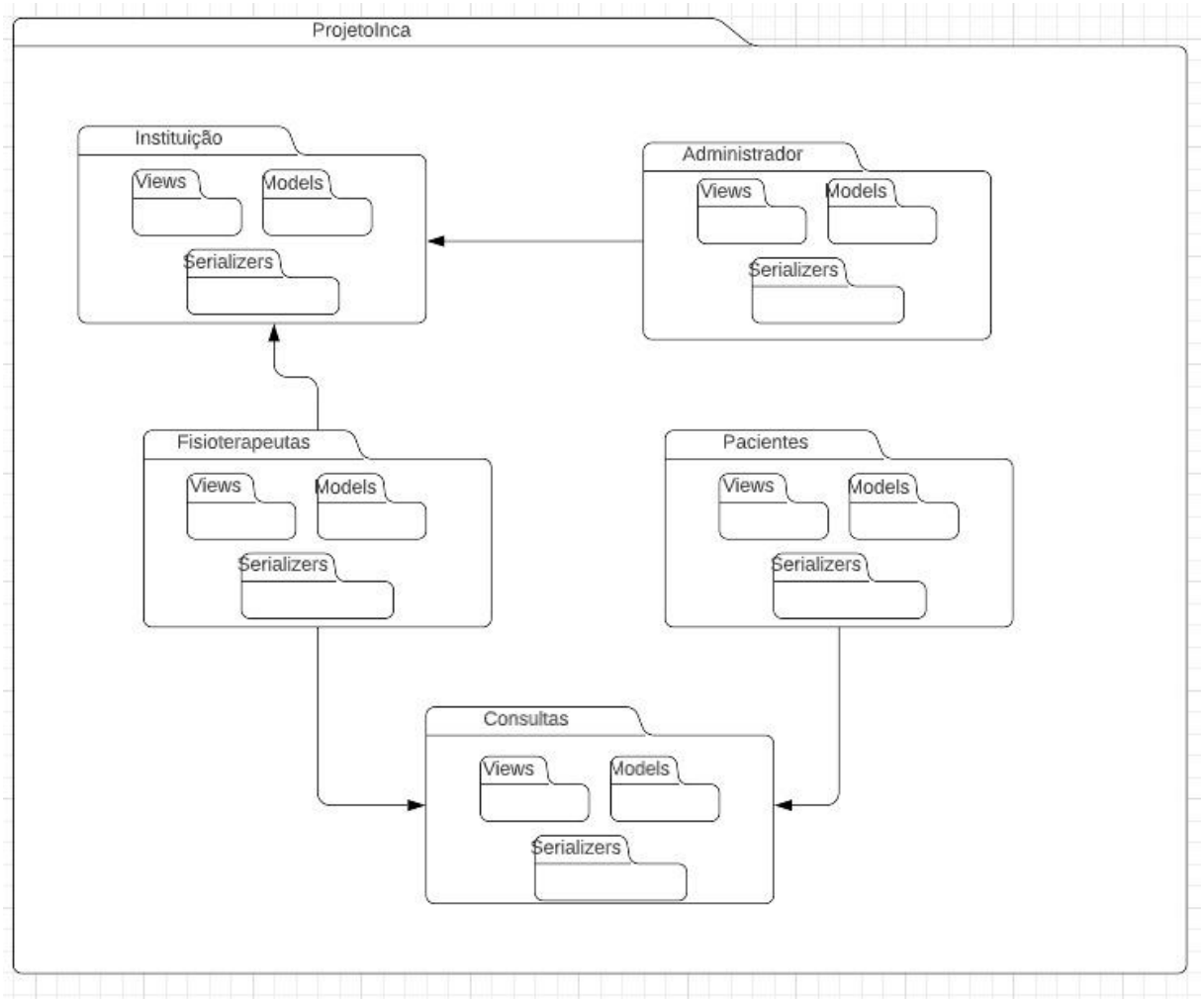
5.2.1 Modelo de classes de domínio (modelo conceitual)

É uma representação visual de classes conceituais ou objetos do mundo real em um domínio de interesse que indicarão as regras de negócio de um sistema. No caso do OnColo foi representado Paciente, Fisioterapeuta, Instituição, Administrador e Consulta.



5.2.2 Diagrama de pacotes

Diagramas de pacote são diagramas estruturais usados para mostrar, em uma forma de pacote, a organização e disposição de vários elementos de modelos. Um pacote é um agrupamento de elementos relacionados, como diagramas, documentos, classes ou até mesmo outros pacotes. A seguir foi representado de uma maneira simples o backend do nosso projeto, onde a pasta principal chamada ProjetoInca possui mais cinco pastas com seus respectivos Models, Views e Serializers.



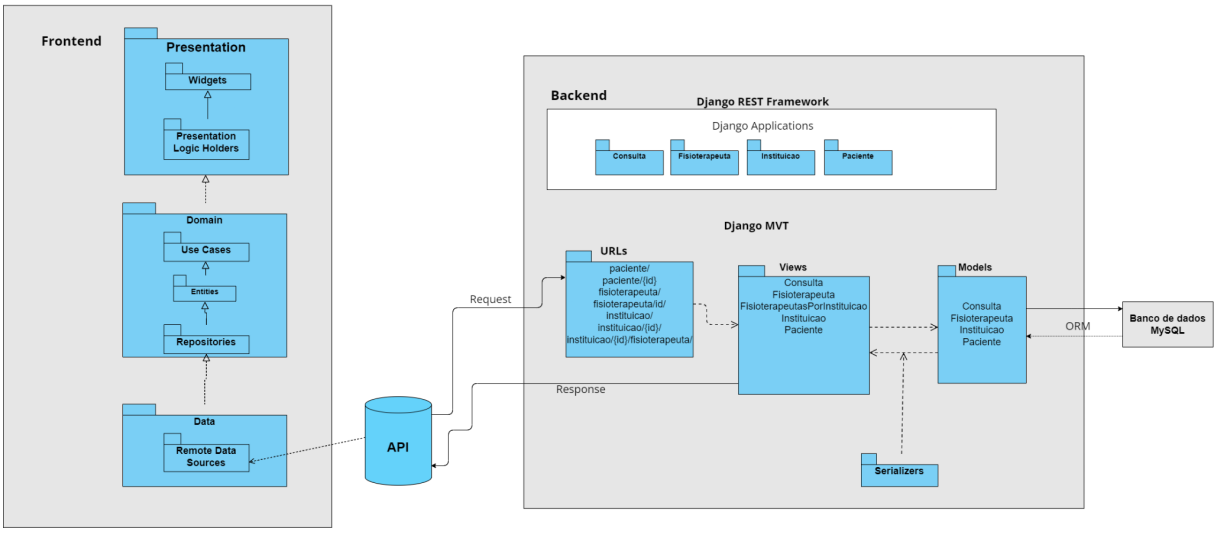
5.3 Visão de Implementação (Desenvolvimento)

A visão da implementação consiste em uma das visões de arquitetura para um sistema. Seu objetivo é capturar quais foram as determinações no processo de implementação. Essa classe de visualização se aplica na atribuição de trabalho aos integrantes de uma equipe de desenvolvimento, além de auxiliar na avaliação do montante de código a ser desenvolvido e considerar a possibilidade de reutilização em grande escala, juntamente com as estratégias de release.

5.3.1 Diagrama de pacotes

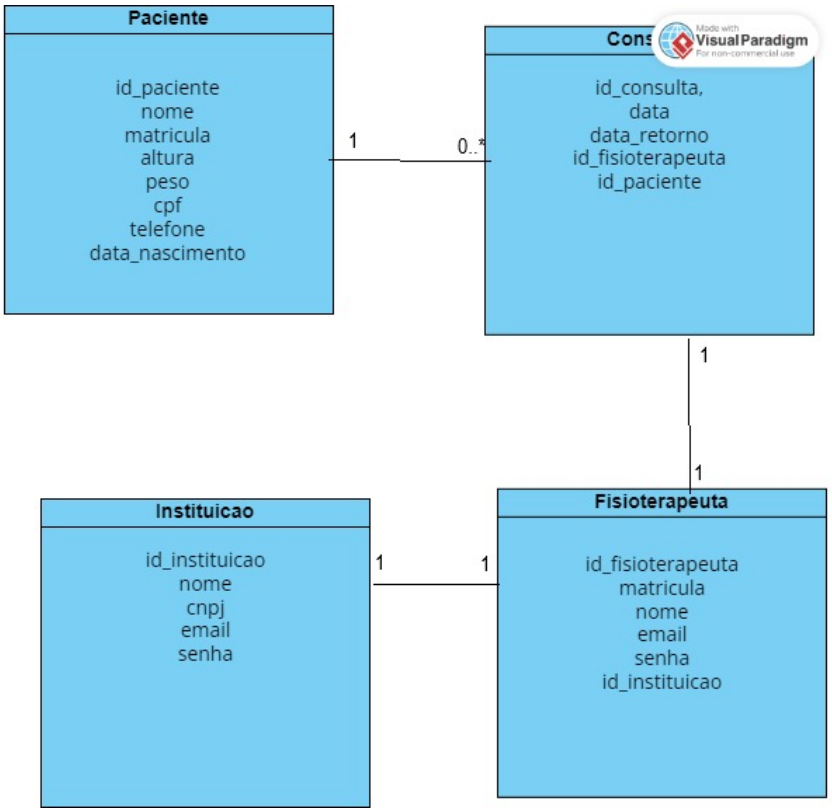
Os diagramas de pacotes consistem em estruturas idealizadas para representar um formato de pacotes com a disposição de diversos itens de um determinado modelo. Sendo um pacote um conjunto de elementos relacionados, podendo representar diagramas, documentos, classes ou mesmo outros pacotes. São comumente utilizados a fim de proporcionar uma organização visual de uma

arquitetura em camadas. No caso do OnColo os pacotes são agrupados em frontend e backend, além de outros elementos externos que dizem respeito a API e ao banco de dados relacional. A ideia é ter uma visualização de como é dividido o código da aplicação de acordo com as tecnologias escolhidas.



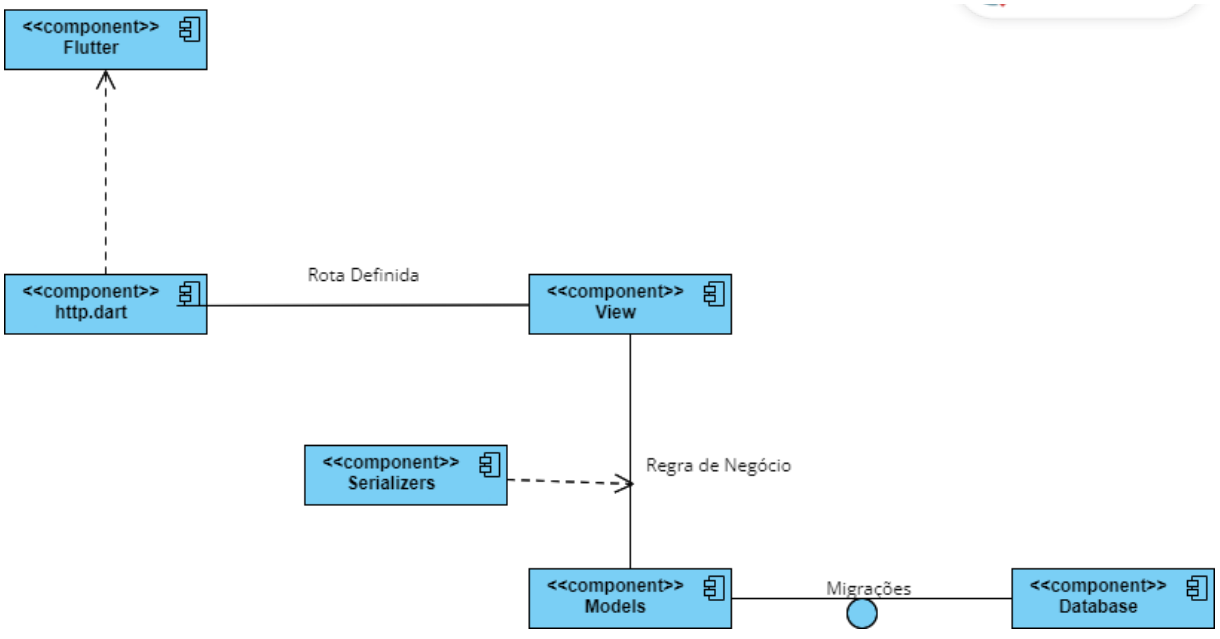
5.3.2 Diagrama de Classes

Diagramas de classes são essenciais para a fase de modelagem do sistema, definindo estaticamente a estrutura do mesmo. É possível utilizá-los para modelar os componentes do sistema, demonstrando seus relacionamentos, descrevendo seu papel e o acoplamento entre eles. Para o OnColo, o diagrama de classes engloba todas as entidades essenciais para o sistema até o desenvolvimento da release em questão e como se dá o seu relacionamento.



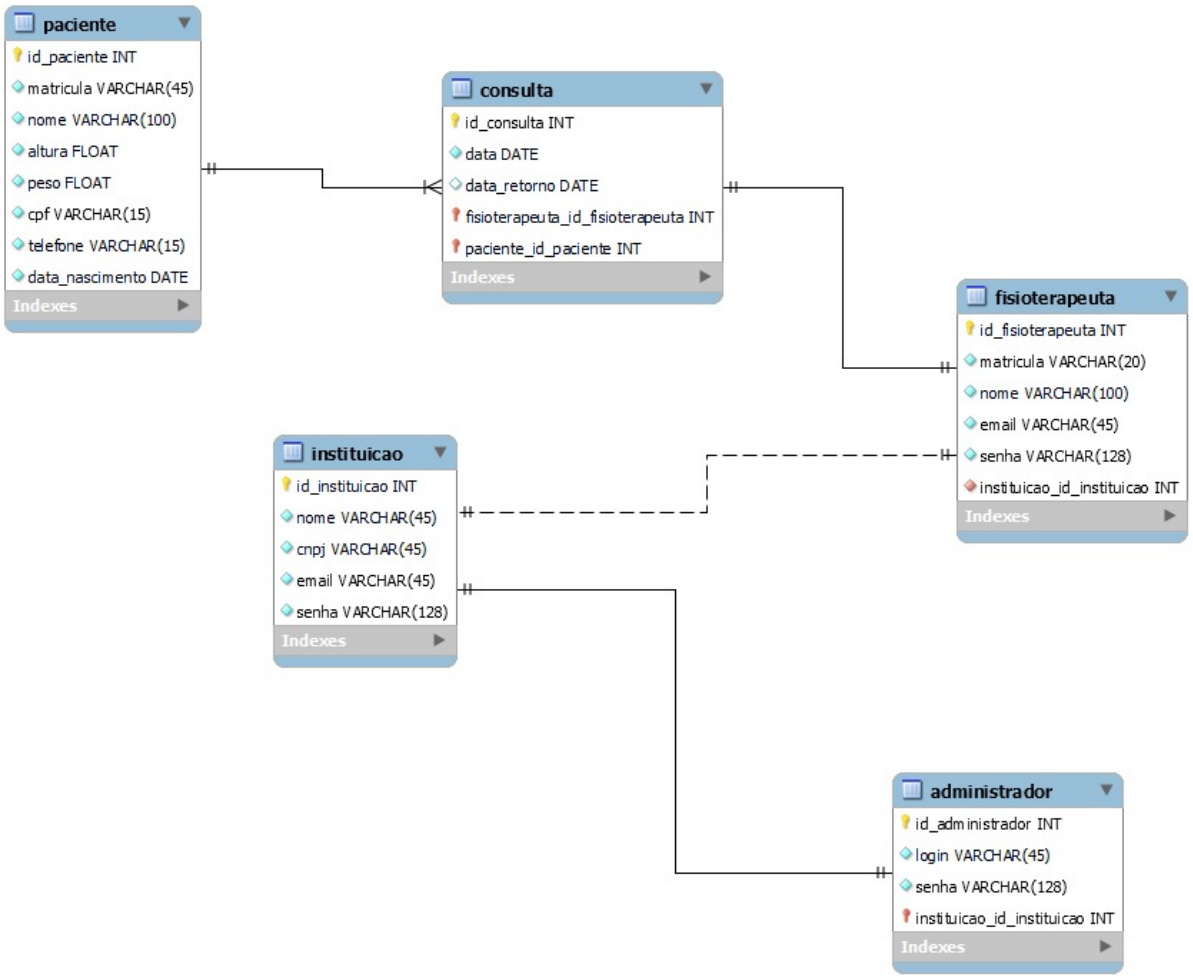
5.3.3 Diagramas de componentes

Diagramas de componentes descrevem os componentes de software, suas interfaces e dependências. Sendo mais aplicado na modelagem de um sistema num nível mais alto quando comparado com o diagrama de classes, por exemplo. Suporta a reutilização de componentes e interfaces, definindo-os e revelando problemas de configuração. No caso do OnColo, foram selecionados os componentes mais importantes do sistema, em termos de movimentação e exibição dos dados.



5.3.4 Diagrama ou Modelo Entidade Relacionamento

Um diagrama de entidade relacionamento determina quais são os dados armazenados e tratados pelo sistema. Sendo assim, leva em consideração os dados independentemente do processamento que é feito sobre os mesmos. No caso do OnColo, utilizou-se o diagrama gerado pelo gerenciador de banco de dados relacional utilizado, através do qual foram definidos as multiplicidades das relações.

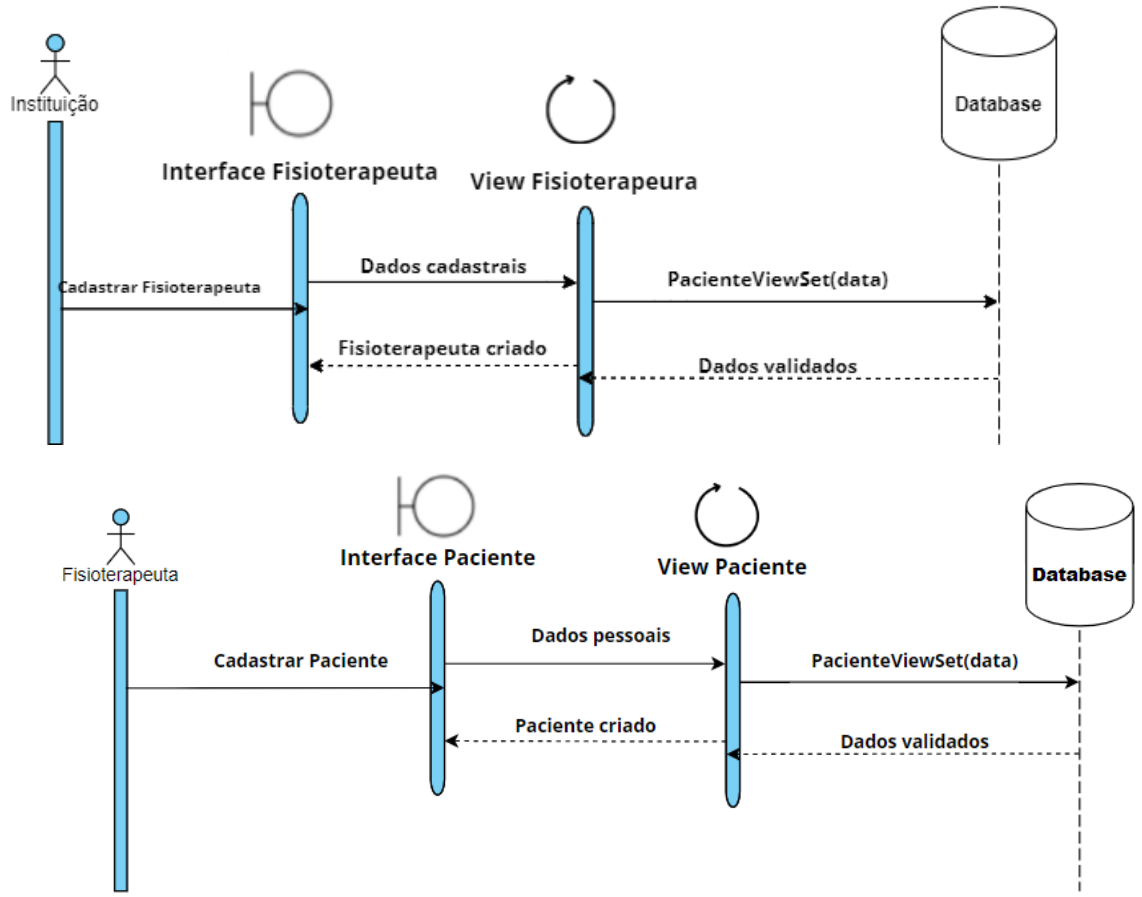


5.4 Visão de Processos

A visão de processos do modelo 4+1 promove a visualização dos aspectos dinâmicos de um sistema, delineando os processos e sua comunicação. Essa visão é responsável por atributos como distribuição, integração, performance, escalabilidade, etc.

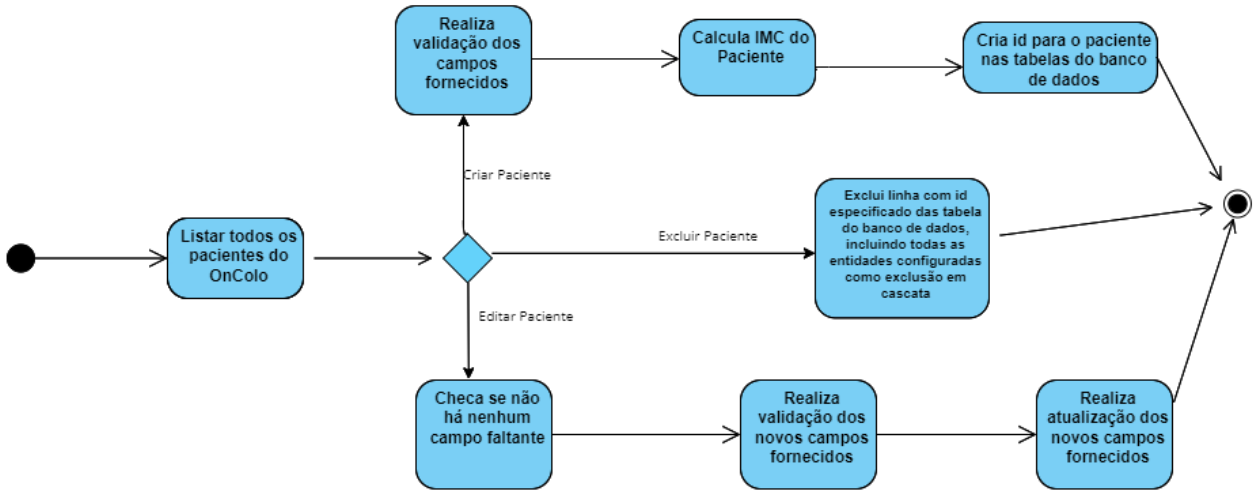
5.4.1 Diagramas de sequência ou comunicação

O diagrama de sequência está diretamente relacionado aos casos de uso de um sistema, exibindo as sequências de mensagens trocadas entre os objetos, demonstrando quais são as estruturas de controle responsáveis por cada etapa da comunicação. Para o Oncolo, são trazidos dois diagramas, um relacionado ao caso de uso CSU01 – Cadastrar paciente e outro para o CSU01 – Cadastrar paciente.



5.4.2 Diagramas de atividades.

Os diagramas de atividades lidam com o comportamento dos fluxos do sistema, retratando a sequência exata de ações em um determinado processo. Para o OnColo foi escolhida a entidade Paciente e seus possíveis fluxos de movimentação de dados na aplicação.

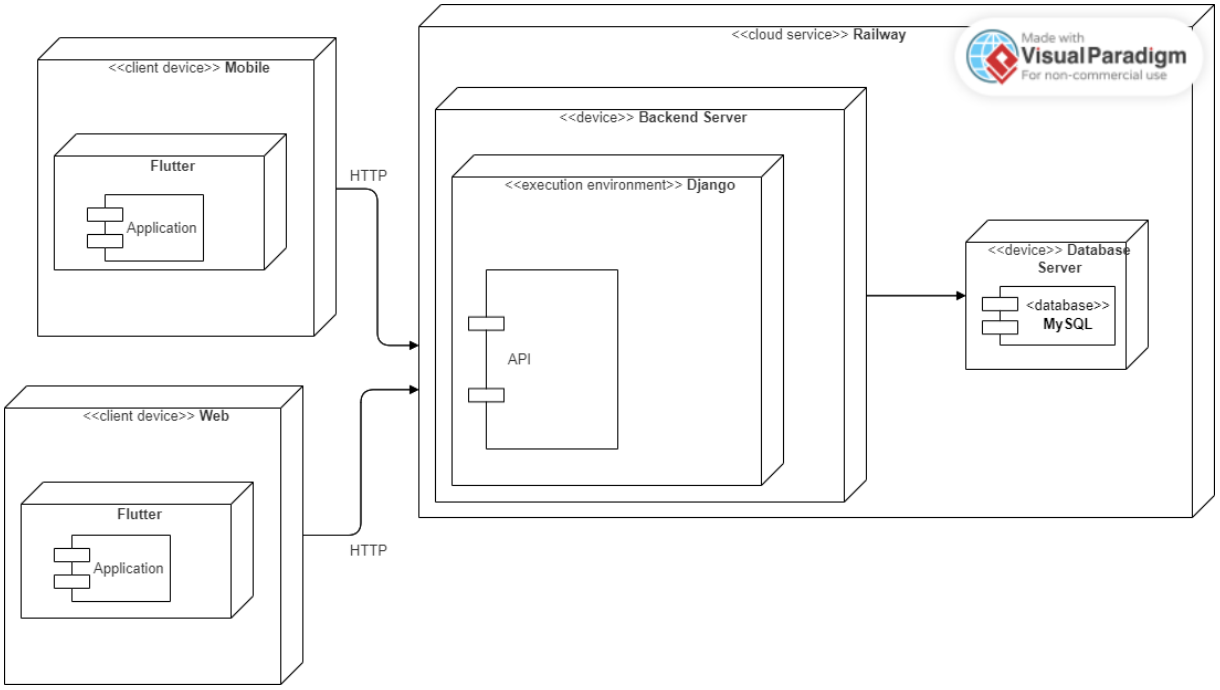


5.5 Visão de Distribuição (Física)

A visão Física leva em consideração os requisitos não funcionais do sistema e tem um impacto mínimo em todo código-fonte, sendo seu mapeamento feito em todos os nós, por isso precisa de ser altamente flexível.

5.5.1 Diagrama de Distribuição(Implantação)

Um diagrama de implementação mostra os componentes e artefatos em relação ao local em que eles são utilizados no sistema implementado. No caso do OnColo foi utilizado o Railway para fazer o deploy, onde utilizamos uma API Django REST framework com MySQL como banco de dados. E para nossa aplicação web e mobile foi utilizado o Flutter.



6. REFERÊNCIAS BIBLIOGRÁFICAS

BASTIAN, S. **Learning Django: REST Framework and MVT Architecture**. Disponível em: <https://medium.com/geekculture/learning-django-rest-framework-and-mvt-architecture-1ca173163f1c>. Acesso em: 14 fev. 2023.

CASTRO, M. **Resumo: Architectural Blueprints — The “4+1” View Model of Software Architecture**. Disponível em: <https://medium.com/@matheus_ortsac/resumo-architectural-blueprints-the-4-1-view-model-of-software-architecture-f03adf3724d2>. Acesso em: 14 fev. 2023.

FILETO, R. **O Modelo Entidade-Relacionamento**. Disponível em: <https://www.inf.ufsc.br/~r.fileto/Disciplinas/INE5423-2010-1/Aulas/02-MER.pdf>. Acesso em: 14 fev. 2023.

GUTIERREZ, F. **O modelo de visão de arquitetura 4+1**. Disponível em: <http://felipeogutierrez.blogspot.com/2017/09/o-modelo-de-visao-de-arquitetura-41.html>. Acesso em: 14 fev. 2023.

IBM. **Diagramas de Classes**. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams>. Acesso em: 14 fev. 2023.

IBM. **Diagramas de Componentes**. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=services-component-diagrams>. Acesso em: 14 fev. 2023.

IBM. **Diagramas de Sequência**.Disponível em: <https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=uml-sequence-diagrams>. Acesso em: 14 fev. 2023.

Lucidchart. **O que é um diagrama de pacotes UML?**. Disponível em: <https://www.lucidchart.com/pages/pt/diagrama-de-pacotes-uml>. Ministério da Saúde. **ESTIMATIVA 2023: Incidência de Câncer no Brasil**. Disponível em: <https://www.inca.gov.br/sites/ufu.sti.inca.local/files//media/document//estimativa-2023.pdf >. Acesso em: 14 fev. 2023.

UCHIYAMA, F. **Arquitetura Visão-Modelo 4+1**. Disponível em: <http://www.basef.com.br/old/uml/204-arquitetura-visao-modelo-41>. Acesso em: 14 fev. 2023.

UFPE. **Conceito: Arquitetura de Software**. Disponível em: <https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/concepts/software_architecture_4269A354.html>. Acesso em: 14 fev. 2023.

UFPE. **Conceito: Visão da Implementação**. Disponível em: <https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/concepts/implementation_view_E373E3B6.html>. Acesso em: 14 fev. 2023.