
# Engenharia de Software

# Padrões de Arquitetura de Software

Prof. Jacson Barbosa

Aluno(s):
Diana Aparecida Osorio Barros
Fernando Brito Rodrigues
Josenilton Santos de Jesus

**Documento de Arquitetura de Software**

**PTM Center (Patient Transfer Management Center)**

## 1. Introdução

O presente documento tem como objetivo descrever o documento de arquitetura do PTM Center. Este sistema tem como finalidade o gerenciamento de transferências hospitalares, que auxilie em aspectos logísticos e burocráticos dos principais stakeholders envolvidos em transferências inter-hospitalares

### 1.1 Finalidade

Este documento oferece uma visão arquitetural geral do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do mesmo. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas no decorrer do processo de desenvolvimento.

### 1.2 Escopo

Este documento auxilia os envolvidos no projeto a captar aspectos arquiteturais do sistema que são necessários para o desenvolvimento de uma solução que atenda às necessidades dos usuários finais. Além de auxiliar no entendimento do sistema por novos membros da equipe.

### 1.3 Definições, Acrônimos e Abreviações

Camada de dados (Data) usando API REST, interage com os dados salvos e inserção de dados.
Core: Implementação regras de negócios, processamento de dados, serviços (processamento de imagens) usando micro serviços.
API Gateway: Serviço de terceiro de abstração para abstrair para comunicar com micro serviços. (abstrair as funcionalidades do sistema)
Regulator Center: Página web com as funções implementadas em
**SaaS**: Software as a Service
**API Gateway**: É a ferramenta de gerenciamento de interfaces de programação de aplicações.

### 1.4 Visão Geral

São apresentados ainda neste documento diferentes visões arquiteturais de como o sistema deve se comportar em diferentes processos, como deve ser implantado é implementado e restrições de desempenho e qualidade.

## 2. Contexto da Arquitetura

### 2.1 Funcionalidades e Restrições Arquiteturais

| Id. | Tipo | Descrição |
|---|---|---|
| RAS_1 | RNF | Sistema clientes utilizado pelos médicos |
| RAS_2 | RNF | Transferência de armazenamento dos dados |
| RAS_3 | RNF | Disponibilidade do Sistema |
| RAS_4 | RNF | Tratamento de erros do Sistema |
| RAS_5 | RNF | Comunicação de usuários do Sistema |
| RAS_6 | RNF | Sistema intuitivo de utilização |
| RAS_7 | RNF | Sistema com cores por longos períodos |
| RAS_8 | RNF | Sistema fácil de aprender |
| RAS_9 | RNF | Informações do sistema restritivo por usuário |

Os RAS citados na tabela acima são referentes a requisitos não-funcionais (ou restrições). Os RAS serão os responsáveis por guiar as decisões sobre quais estilos arquiteturais serão adequados para favorecer os atributos de qualidade priorizados. O RAS_1 propõe um sistema cliente utilizado pelos(as) médicos(as), que será desenvolvido na plataforma mobile. O RAS_2 é a transferência e armazenamento dos dados usando protocolos seguros (ex. protocolos e regulações de saúde, criptografia e salting). O RAS_3 relata sobre a disponibilidade do sistema, tendo como restrição de 24h por dia e 7 dias por semana, não podendo passar por períodos de

indisponibilidade. O RAS_4 o sistema mobile deverá tratar erros de comunicação com o servidor e apresentá-los para os usuários de forma amigável. RAS_5 o sistema deverá realizar as comunicações entre médicos de origem/destino e médico regulador (tempo de resposta) em até 10 segundos para no mínimo 99% dos casos. RAS_6 o sistema deve ser fácil de usar. O RAS_7 sistema deve ter cores agradáveis para o uso por longos períodos. O RAS_8 sistema deve ser fácil de aprender. O RAS_9 sistema não deve disponibilizar informações e funcionalidades para usuários não autorizados.

### 2.2 Atributos de Qualidades Prioritários

Visto que o principal requisito da arquitetura proposta é tolerância à falhas, o atributo de qualidade **Disponibilidade** onde os servidores serão distribuídos com foco em tolerância a falhas fornecendo **Confiabilidade.** O software terá uma arquitetura que vai reparar e mascarar falhas de modo a cumprir seu trabalho sem que o cliente seja prejudicado. Os estilos arquiteturais, camadas, componentes e cliente-servidor foram escolhidos por favorecer escalabilidade, disponibilidade, também favorecem o atributo de qualidade **Manutenibilidade**.

**Confiabilidade**: Tolerância a falhas - redundância, replicação e backup.
**Segurança**: Autenticação de dois fatores (2FA) com métodos seguros, como OTP, Segurança na transferência de dados sensíveis (Ex. Protocolos de saúde, LGPD)
**Escalabilidade:** escalabilidade horizontal automática usando containers.
**Manutenibilidade:** Modularização dos casos de uso, APIs, micro-serviços.

## 3. Representação da Arquitetura

Conforme definido pelos tópicos anteriores (2.1 e 2.2), a arquitetura do software a ser desenvolvido será uma arquitetura híbrida e independente que une as principais características dos estilos arquiteturais: Camadas, API REST, Micro-Serviços, API Gateway Saas e aplicação Web, e prioriza os Atributos de qualidade: Segurança, Manutenibilidade e **Confiabilidade**.

Para representar as decisões arquiteturais definidas ao findar da análise, serão utilizados os pontos de vista, que são:

| **Ponto de Vista** | **Visão** | **Diagrama(s)** |
|---|---|---|
| **-** | **Casos de Uso** | Casos de Uso |
| **Projetista** | **Desenvolvimento** | Componentes |

| | | |
|---|---|---|
| **Desenvolvedor** | **Segurança** | Pontos-fracos |
| **Implantador** | **Física** | Implantação |

## 4. Ponto de vista dos Casos de Uso

### 4.1 Descrição

Para fornecer uma base para o planejamento da arquitetura e de todos os outros artefatos que serão gerados durante o ciclo de vida do software, é gerada, na análise de requisitos, uma visão chamada visão de casos de uso. Só existe uma visão de casos de uso para cada sistema. Ela ilustra os casos de uso e cenários que englobam o comportamento, as classes e riscos técnicos significativos do ponto de vista da arquitetura. A visão de casos de uso é refinada e considerada inicialmente em cada iteração do ciclo de vida do software.

### 4.2 Visão de Casos de Uso

Cada requisito foi considerado um caso de uso e analisado de forma a gerar o diagrama de casos de uso do software a ser desenvolvido. Sendo:

1. Visualizar estatísticas: Tem como função visualizar o dashboard de estáticas dos pacientes, hospitais e médicos, incluindo informações de internações e quantidade de referências.
2. Visualizar prontuário: Tem como função visualizar todas as informações médicas do paciente.
3. Inserir Informações do paciente: Tem como função inserir informações mais recentes do paciente e simular possíveis soluções para o problema, após análise das informações recentes
4. Visualizar Paciente: Ler as informações imputadas durante a transferência (Médico regulador, médico de destino) e simular possíveis soluções para o problema, após análise das informações recentes

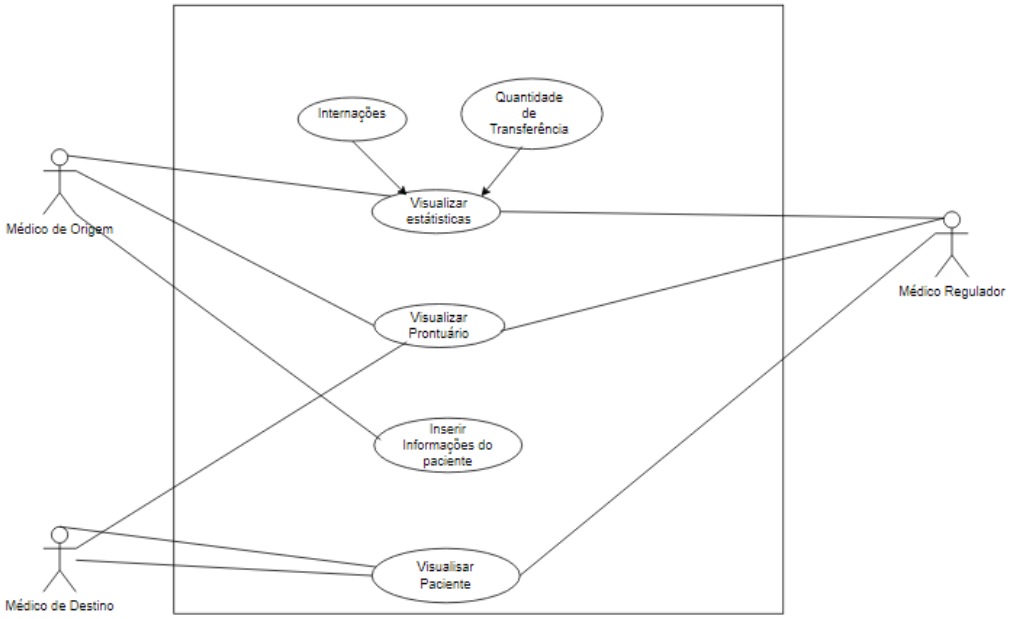


**Figura 1:** Diagrama de Caso de uso

## 5. Ponto de Vista do Projetista

### 5.1 Visão Geral

O ponto de vista do projetista é direcionado aos projetistas e desenvolvedores do software e tem como objetivo definir as principais partes que o compõem, tal como os componentes, além de definir quais as suas responsabilidades. Foi escolhida por ser uma visão primordial para a compreensão do software e de todo o seu ecossistema.

O modelo arquitetural proposto para a construção deste software será composto por 4 (quatro) componentes essenciais: *Data*, *Core*, *SaaS* e *Regulator Center*.

### 5.2 Visão de Componentes

O componente *Data* é responsável pelo gerenciamento dos dados, escalabilidade de recursos de banco, ORM. Fornece a primeira camada de segurança dos dados sendo o único ponto de acesso aos dados.

O componente *Core* é responsável pela implementação da lógica presente no software, processamento de dados externos, possuindo um conjunto de 2 camadas internas para a segregação das responsabilidades. Cada camada é independente, para garantir desenvolvimento independente, isolamento de dados, regras e deploy. Será feito o desenvolvimento sobre arquitetura de micro-serviços.

O componente *SaaS* fornecerá uma abstração de todos os recursos do software via *API Gateway* serviço de terceiro (AWS).

A interação entre os componentes citados neste tópico pode ser visualizada por meio do **Diagrama de componentes UML** abaixo:

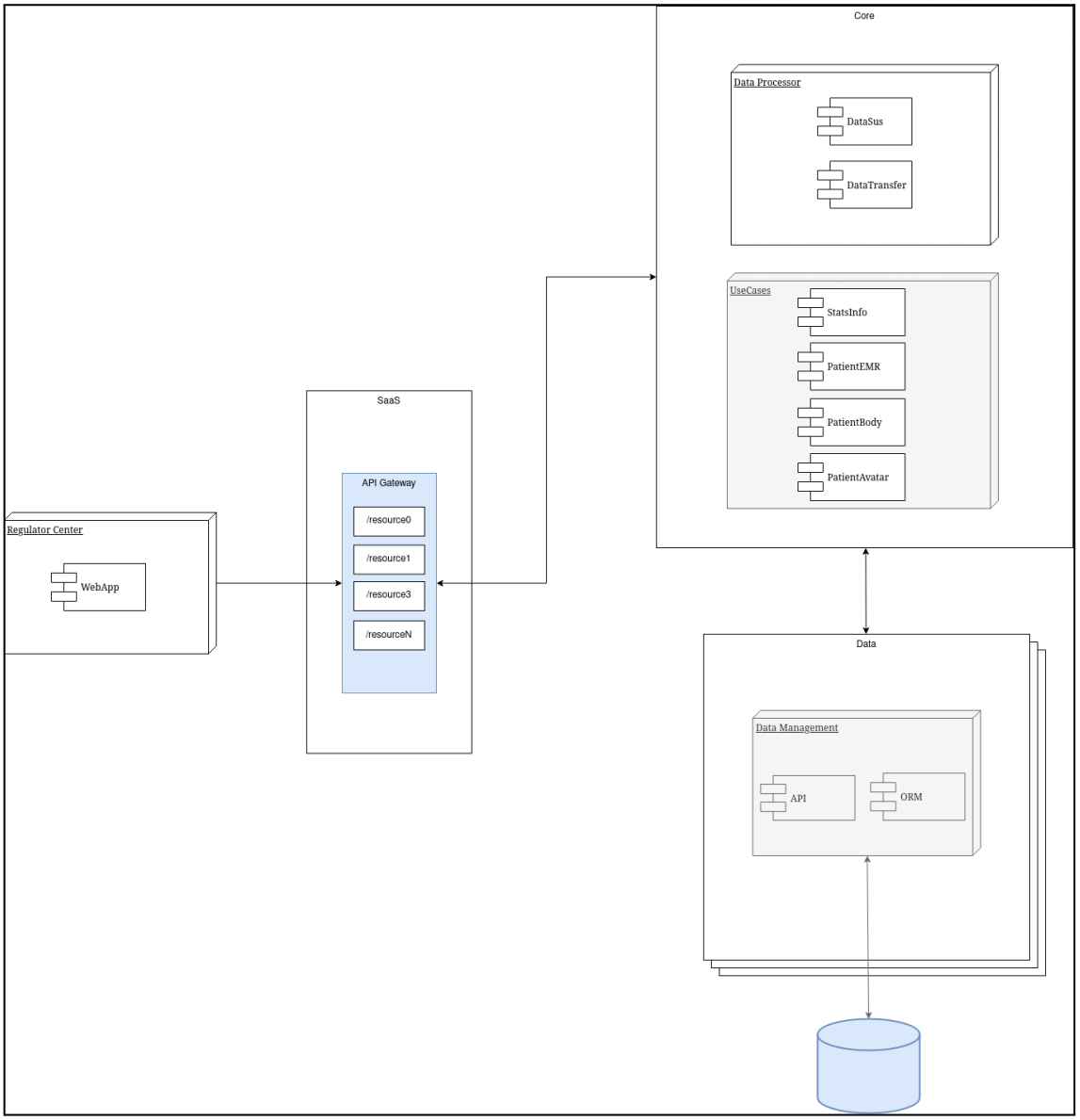


**Figura 2:** Diagrama de projetista

### 5.3 Detalhamento das Camadas

A camada *Data Management* é responsável por prover uma interface de acesso aos dados, gerenciamento e CRUD de todas as entidades. Esta interface será fornecida via API REST.

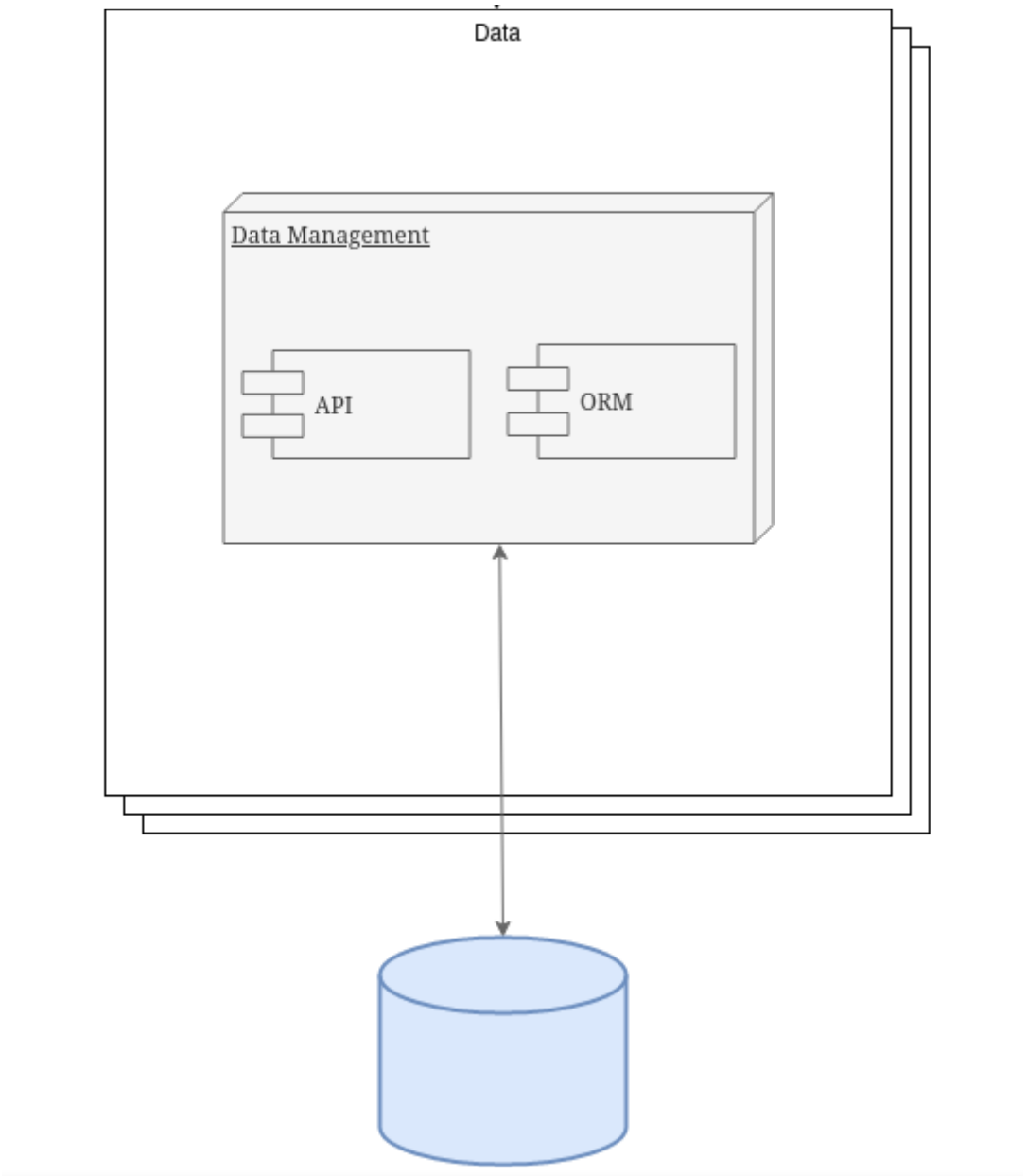


**Figura 3:** Camada de gerenciamento de dados

A camada *Data Processor* será responsável por recebimento, processamento, tratamento e interação com fontes de dados externos referente aos sistemas habilitadores. Por exemplo, o DataSus é o componente que irá interagir com dados fornecidos pelo Sistema Único de Saúde.

A camada UseCases

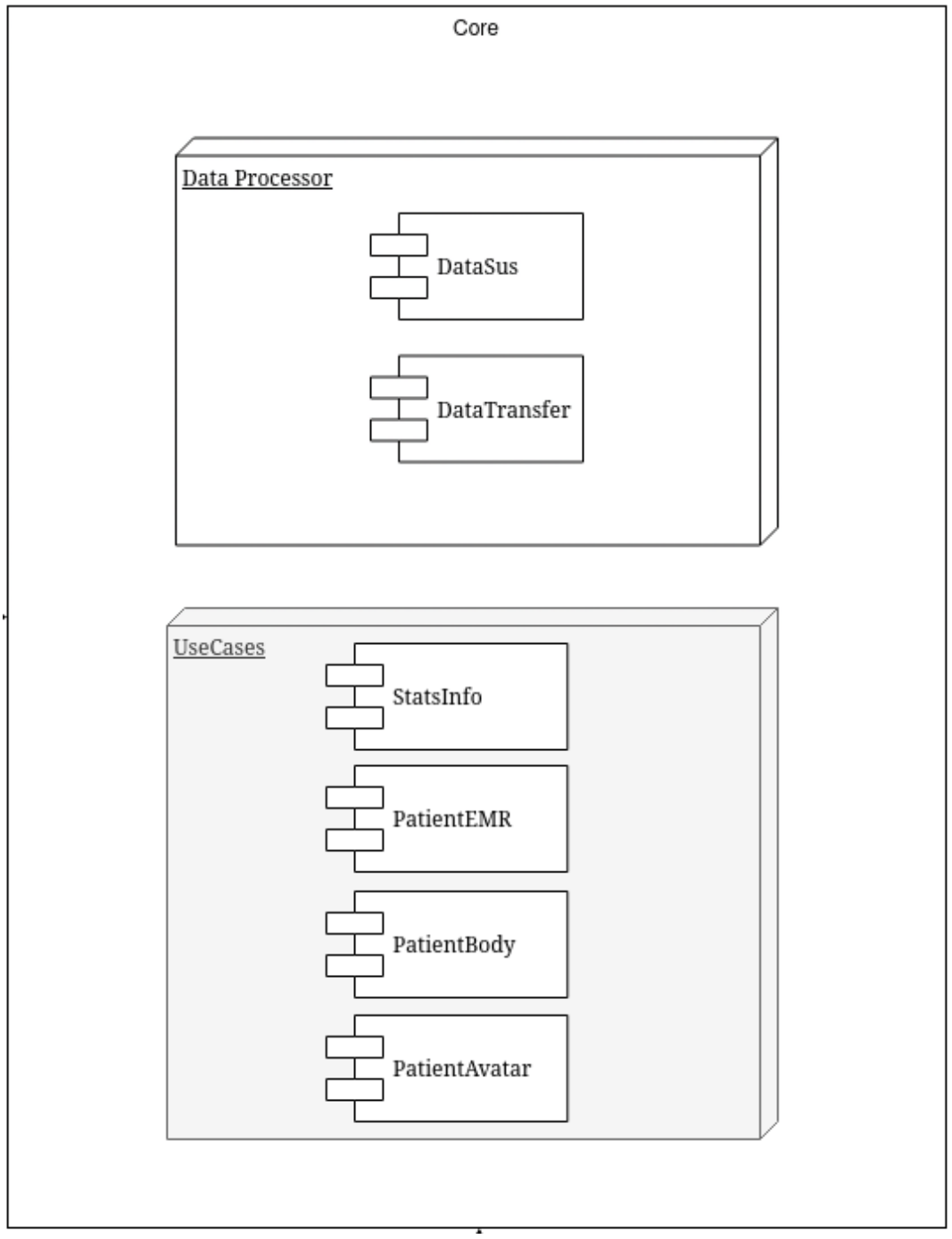


**Figura 3:** Componente de regras de negócio e serviços

## 6. Ponto de vista do Implantador

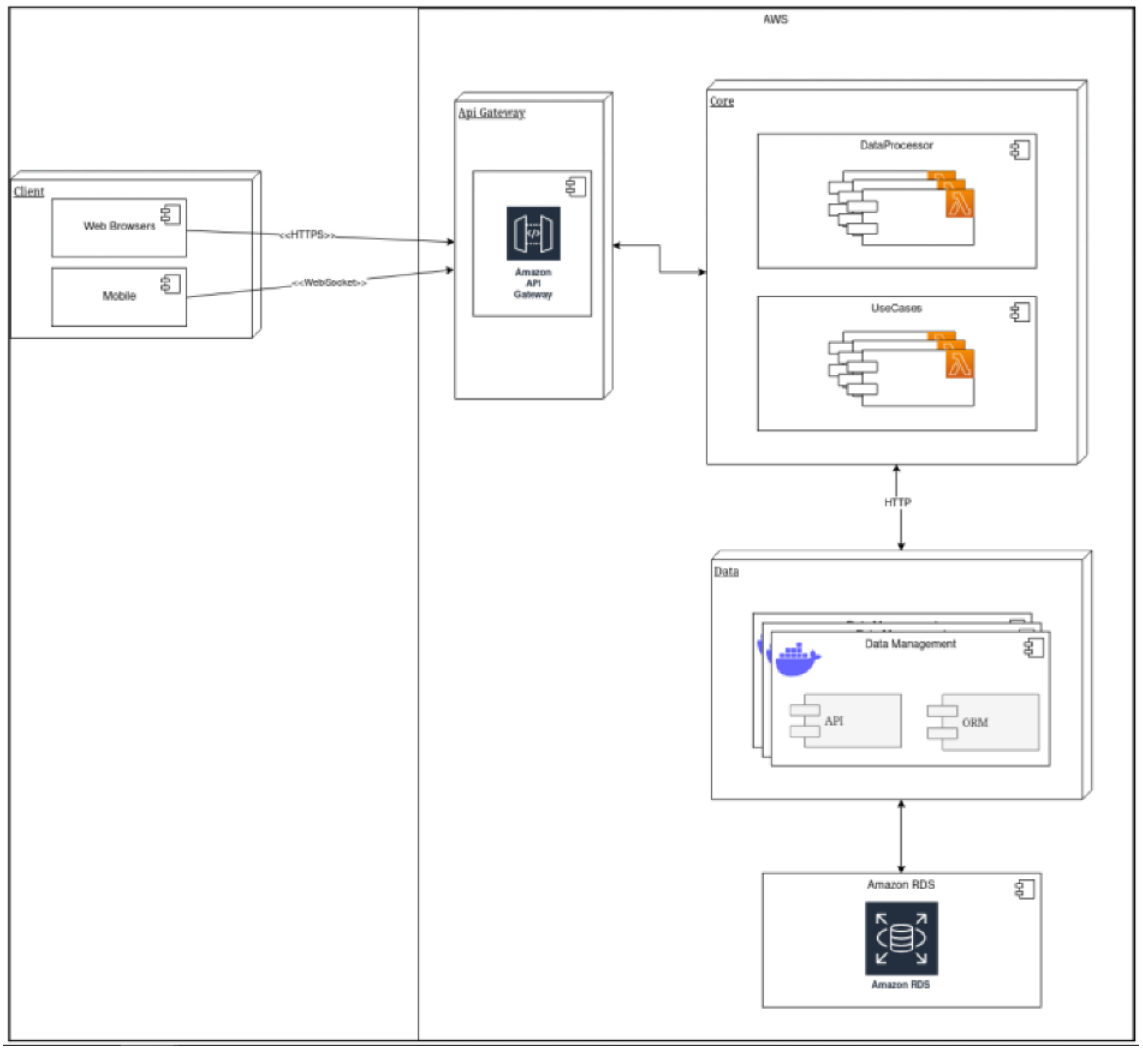


**Figura 2:** Diagrama de implantação UML