Respira

**Documento de Arquitetura de Software**

**Versão 1.**2

**Histórico da Revisão**

| **Data** | **Versão** | **Descrição** | **Autor** |
| --- | --- | --- | --- |
| 23/04/2025 | 1.0 | Versão inicial do documento | Pietro do Couto Freitas |
| 24/04/2025 | 1.1 | Escrita do documento e adição de diagramas | Pietro do Couto Freitas |
| 24/04/2025 | 1.2 | Refinamento de seções, escrita de parte da seção arquitetural e adição de diagramas | Lucas Rohr Carreño |
|  |  |  |  |

**Índice Analítico**

1. Introdução 4

1.1 Finalidade 4

1.2 Escopo 4

1.3 Definições, Acrônimos e Abreviações 4

1.4 Referências 4

2. Representação Arquitetural 4

3. Metas e Restrições da Arquitetura 4

4. Visão de Casos de Uso 5

5. Visão Lógica 5

5.1 Visão Geral 5

5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura 5

6. Visão de Implantação 5

7. Visão da Implementação 5

7.1 Visão Geral 5

7.2 Camadas 6

8. Visão de Dados 6

8.1 Modelo de objetos persistentes 6

8.2 Estratégias 6

8.3 Modelo Relacional 6

**Documento de Arquitetura de Software**

# **Introdução**

## **Finalidade**

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquitetônicas para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquitetônicas significativas que foram tomadas em relação ao sistema.

## **Escopo**

O documento trata da Arquitetura de Software do aplicativo **Respira**, voltado ao monitoramento de qualidade do ar.

## **Definições, Acrônimos e Abreviações**

Definição de palavras-chave utilizadas ao longo desse documento, essenciais para a compreensão do que está aqui exposto.

**JavaScript** – Linguagem de programação

**NodeJS** – Ferramenta para rodar JavaScript fora do navegador

**Mobile** – Dispositivos móveis como celulares

**Framework** – Ferramenta contendo todos os artifícios necessários para desenvolver uma aplicação

**React Native** – Framework em JavaScript para desenvolvimento mobile

**Play Store** – Loja de aplicativos do Android

**App** **Store** – Loja de aplicativos do iOS

**Docker** – Sistema para isolamento de processos em uma máquina

**Arduino** – Microcontrolador com grande popularidade e adesão

**SQL** – Structured Query Language, linguagem de banco de dados

**App** – Abreviação de aplicativo

**Linux** – Sistema operacional comum em servidores

## **Referências**

| **Título** | **Link** | **Data** |
| --- | --- | --- |
| Diagrama de implantação | [diagrama\_de\_implantação.drawio](https://app.diagrams.net/?state=%7B%22ids%22:%5B%221G3wTm7mbA6BlfVklKOvf0utEueDSFNyq%22%5D,%22action%22:%22open%22,%22userId%22:%22103928204842177296090%22,%22resourceKeys%22:%7B%7D%7D) | 23/04/2025 |
| Diagrama de componentes | [diagrama\_de\_componentes.drawio](https://app.diagrams.net/?state=%7B%22ids%22:%5B%22141wech3fwMi4hgB0pR853idcJFKax9G6%22%5D,%22action%22:%22open%22,%22userId%22:%22103928204842177296090%22,%22resourceKeys%22:%7B%7D%7D) | 23/04/2025 |
| Diagrama relacional | [diagrama\_relacional.drawio](https://app.diagrams.net/#G1fYJd04FuiNJDOkveQrjGHmP3XUrP8lWP) | 11/24/2024 |
| Realização de casos de uso | [realização\_de\_casos\_de\_uso.docx](https://docs.google.com/document/d/1hdwn7Sy4Tjk3ts_1ti8xihvRMYlbHvIX/edit) | 23/04/2025 |

# **Representação Arquitetural**

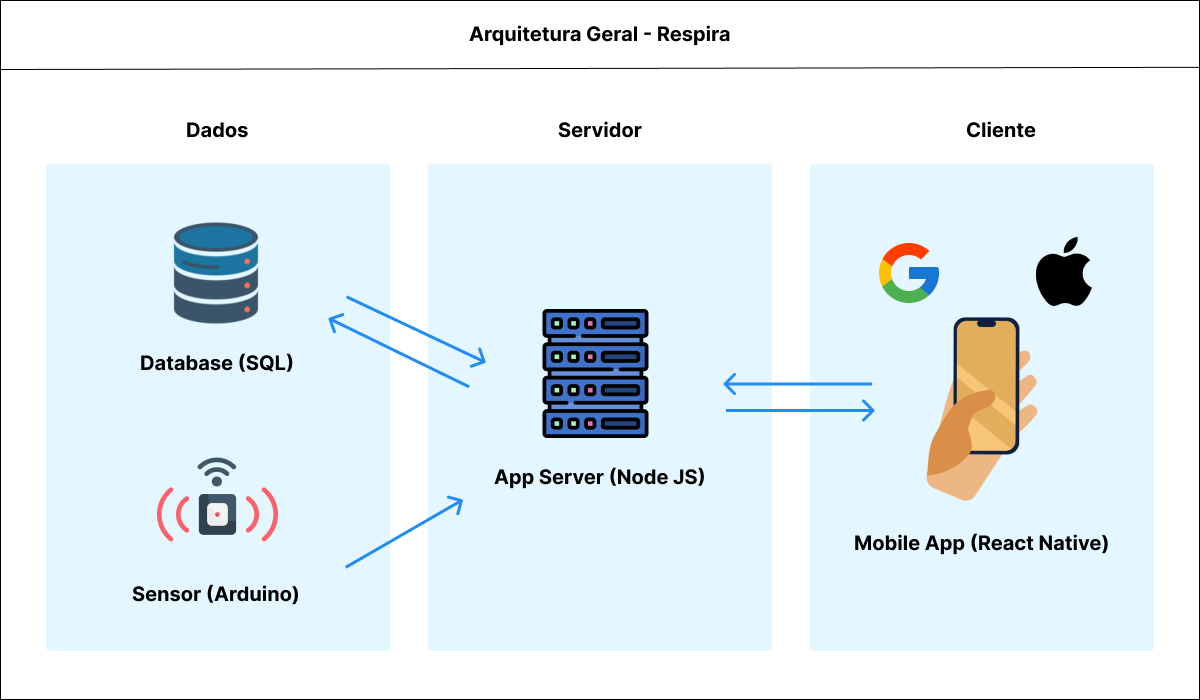
A arquitetura utilizada para planejamento e concepção do Respira é a MVC. Nesse padrão, trabalha-se com a ideia de um modelo, responsável pela estruturação dos dados; uma view, que exibe a interface gráfica ao usuário e um controller, que no caso do Respira é o servidor responsável pelo gerenciamento do que ocorre no sistema. Para representar a modelagem do sistema, utiliza-se das seguintes visões:

**Visão de Casos de Uso** – Representada em diagramas de caso de uso

**Visão Lógica** – Representada por diagramas de classe

**Visão de Implantação** –Representada por um diagrama de implantação

**Visão de Implementação** –Representada por um diagrama de componentes

****

# **Metas e Restrições da Arquitetura**

Esta seção descreve os requisitos de software e os objetivos que têm um impacto significativo na arquitetura, como proteção, segurança, privacidade, uso de um produto desenvolvido internamente e adquirido pronto para ser usado, portabilidade, distribuição e reutilização.

As seguintes restrições de requisito e de sistema impactam a arquitetura restringindo ela de alguma maneira. São elas:

* O sistema deverá ser acessado por meio de aplicativo móvel;
* O sistema requisitará acesso à internet para funcionar devidamente;
* A linguagem de desenvolvimento utilizada para o sistema será o JavaScript com TypeScript no App e no Servidor;
* O Servidor será construído utilizando Node JS;
* O Sistema Operacional que dará suporte ao servidor em um cenário real será LINUX;
* Em um cenário real, o sistema isolaria o servidor em containers usando Docker;
* O Sistema Gerenciador de Banco de Dados escolhido para suportar a aplicação será o PostgreSQL;
* Cliente (app) e servidor irão conversar via REST;
* Em um cenário real, os dados de qualidade do ar virão de um Sensor Arduino. Já no MVP de entrega, serão cadastrados diretamente em Banco de Dados devido ao escopo reduzido da disciplina.

# **Visão de Casos de Uso**

Nesta seção, encontram-se os casos de uso fundamentais para atender as demandas de usuários. Note que funcionalidades como registro de usuários não estão aqui registradas, por representarem o passo mais básico para a utilização do aplicativo.

Em negrito, os casos de uso mais importantes para o Respira:

* **Caso de Uso 01 - Registrar novo alerta**
* Caso de Uso 02 - Consultar Favoritos
* **Caso de Uso 03 - Consultar qualidade de ar local**
* Caso de Uso 04 - Consultar histórico de ar local
* **Caso de Uso 05 - Pesquisar e Consultar Qualidade do Ar**
* Caso de Uso 06 - Pesquisar e Consultar Histórico de Qualidade do Ar

# **Visão Lógica**

*[Esta seção descreve as partes significativas do ponto de vista da arquitetura do modelo de design, como sua divisão em subsistemas e pacotes. Além disso, para cada pacote significativo, ela mostra sua divisão em classes e utilitários de classe. Apresente as classes significativas do ponto de vista da arquitetura e descreva suas responsabilidades, bem como alguns relacionamentos, operações e atributos de grande importância.]*

## **Visão Geral**

*[Esta subseção descreve toda a decomposição do modelo de design em termos de camadas e de hierarquia de pacotes.]*

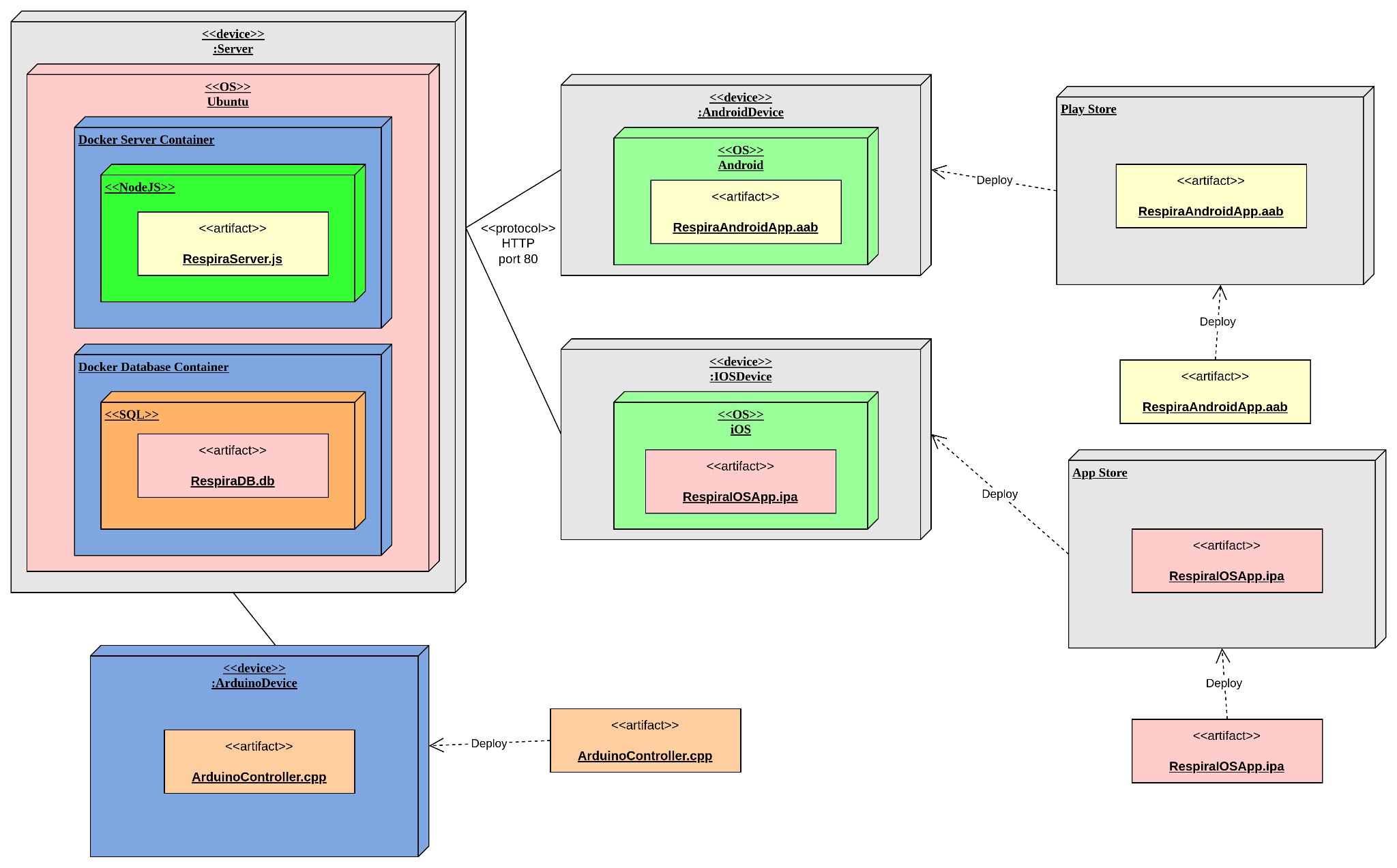
## **Pacotes de Design Significativos do Ponto de Vista da Arquitetura**

*[Para cada pacote significativo, inclua uma subseção com o respectivo nome, uma breve descrição e um diagrama com todos os pacotes e classes significativos nele contidos.*

*Para cada classe significativa no pacote, inclua o respectivo nome, uma breve descrição e, opcionalmente, uma descrição de algumas das suas principais responsabilidades, operações e atributos.]*

# **Visão de Implantação**

A implantação consiste em um servidor rodando um sistema operacional Linux, na distribuição Ubuntu. Dentro desse sistema, serão criados dois containers Docker, um para rodar o servidor em NodeJS, responsável pelo gerenciamento central do projeto; o outro, para rodar uma instância de um banco de dados SQL. Além disso, existirão diversos microcontroladores Arduino, que estarão distribuídos com a finalidade de ler dados de sensores e transmitir essas informações para o serviço em NodeJS que roda no servidor principal. Por fim, a implantação também contemplará o deploy do app na Play Store e App Store, de onde cada usuário pode realizar o download do aplicativo e mantê-lo atualizado.

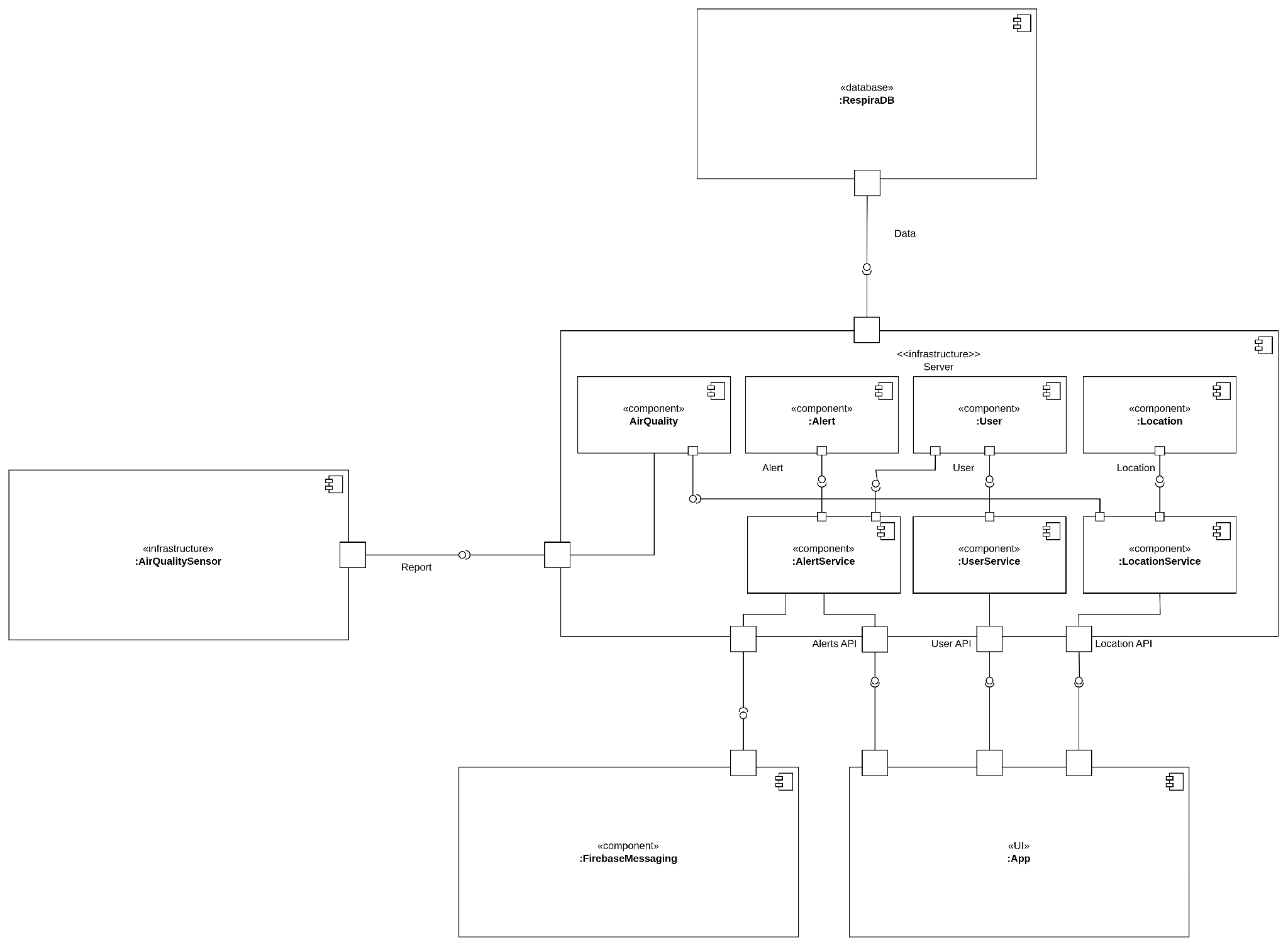
**

# **Visão da Implementação**

A visão de implementação aborda a arquitetura sobre a visão da estrutura dos componentes do sistema, assim como eles serão organizados e se relacionam uns com os outros em termos de dependência.

A visão de implementação reflete a composição física da implementação, evidenciando componentes, subcomponentes e elementos diversos (como diretórios, arquivos, dados, executáveis). Assim, pode-se dizer que a camada de implementação segue aquilo definido pela visão lógica.

## **Visão Geral**

**

## **Camadas**

A seguir, é vista uma breve descrição de cada camada do sistema Respira:

## Camada de Visualização

É representada pela “UI”, que no diagrama aparece como o componente App. É a interface visual do usuário com o sistema do Respira. Aqui, será implementado o framework React Native.

## Camada de Controle

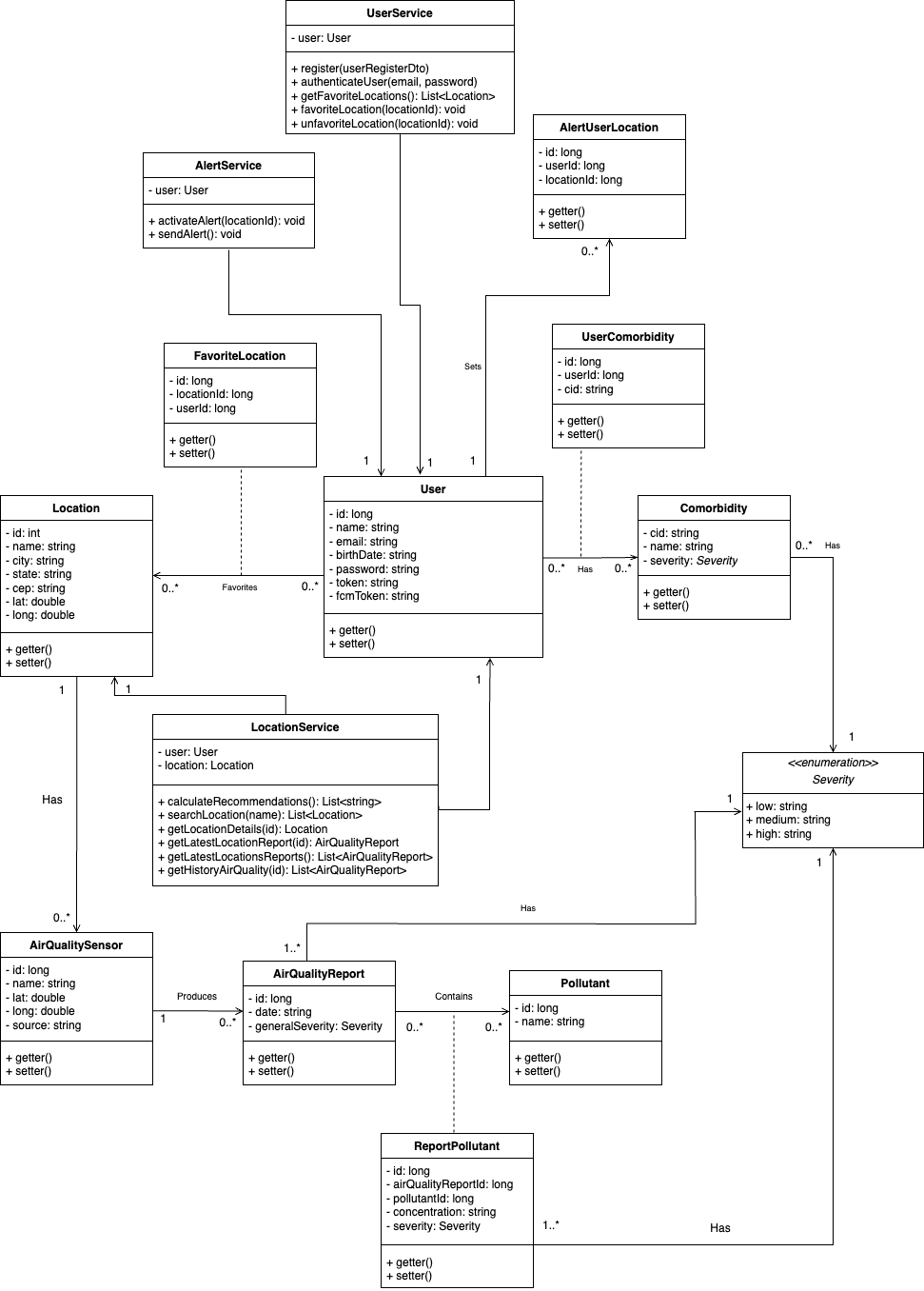
Está representada pelo Server e FirebaseMessaging no diagrama. Atua como um ponto central para gerenciar o que é lido pelos sensores, manipular o banco de dados e lidar com as requisições de usuário.

## Camada de Modelo

No diagrama, é demonstrada como RespiraDB. É a estrutura responsável por fornecer ao Server uma interface de interação com o banco de dados.

# **Visão de Dados**

## **Modelo de objetos persistentes**

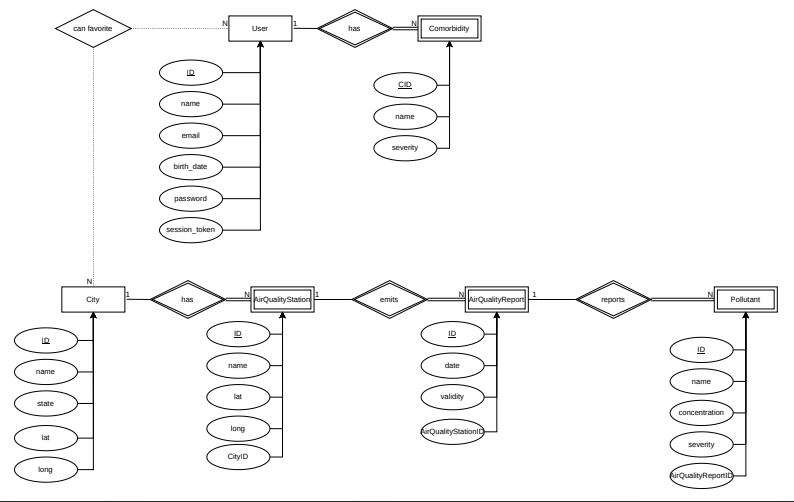


## **Estratégias**

A estratégia utilizada para realizar o mapeamento do modelo de banco de dados foi o levantamento de requisitos para o correto funcionamento do sistema, além da devida atribuição a entidades.

## **Modelo Relacional**

Abaixo, segue o modelo de entidades relacionais do banco de dados do Respira:

**