ENCONTRE SEU PET

Documento de Arquitetura de Software

GUSTAVO HENRIQUE RODRIGUES SANTOS SILVA

Índice

1. Introdução

1.1. Finalidade

Este documento tem como objetivo fornecer uma visão geral da arquitetura que será usada no desenvolvimento do projeto e permitir um maior entendimento do módulo ESP, e de como ele irá se comportar e se comunicar com as outras aplicações que compõem o projeto. Ele deve mostrar de forma clara e objetiva as decisões arquiteturais que foram tomadas em relação ao projeto. Além disso, o mesmo é direcionado para aos stakeholders do software a ser desenvolvido, tais como: Gerentes do Projeto, Clientes e equipe técnica, possuindo grande foco para os Desenvolvedores e a Equipe de implantação.

1.2. Escopo

Este documento se baseia no documento de requisitos do ESP para definir os atributos de qualidade a serem priorizados, bem como, os estilos arquiteturais que favorecem tais atributos e as representações das visões arquiteturais e seus sub-produto

1.3. Definições, Acrônimos e Abreviações

- **ESP:** Encontre Seu Pet
- **HTTP:** Protocolo de Transferência de Hipertexto;
- **Atributos de qualidade:** São atributos que impactam diretamente na concepção de um software, são definidos conforme a ISO-IEEE 9126;
- **Id.:** Identificador;
- **Software:** Conjunto de documentações, guias, metodologias, processos, códigos e ferramentas para a solução de um problema;
- **UML:** Sigla para Linguagem de Modelagem Unificada;
- **Visão Arquitetural:** Produto resultante da interpretação de um Stakeholder do sistema;
- **Ponto de Vista Arquitetural:** Produto resultante da execução de uma Visão Arquitetural;
- **Arquitetura de Software:** Forma como os componentes são agrupados com o objetivo de construir um software ou sistema;
- **Nó-físico:** Termo para representar um componente físico de modo geral, como um navegador ou um banco de dados, por exemplo.

1.4. Referências

Id.	Nome do Artefato

AAS_1	Especificação de Necessidades

1.5. Visão Geral

Os próximos tópicos descrevem quais serão os requisitos e restrições utilizados para definir a arquitetura a ser implementada, bem como, quais atributos de qualidades serão priorizados e o porquê da escolha.

Quais os padrões arquiteturais serão utilizados conforme os atributos de qualidade selecionados e como funcionará o trade-off entre esses padrões arquiteturais, bem como o porquê da escolha dos padrões arquiteturais.

Quais e como as visões arquiteturais serão detalhadas e quais os pontos de vista da arquitetura serão utilizados para descrever as visões.

2. Contexto da Arquitetura

2.1. Funcionalidades e Restrições Arquiteturais

Os requisitos deixam explícito que a aplicação deve estar em contexto mobile e de interoperabilidade, deixando claro a existência de componentes, um cliente e um servidor. Desse modo, fica claro a utilização do estilo arquitetura **Cliente-Servidor**.

Além disso, a necessidade do requisito não funcional de manutenibilidade mostra que é necessário implementar uma arquitetura modularizada, como definido pelo estilo de **Camadas**. Desse modo, caso um servidor fique indisponível, a informações do tempo ainda serão consultadas e armazenadas.

2.2. Atributos de Qualidade Prioritários

Conforme definido no tópico anterior (2.1), os estilos arquiteturais cliente-servidor e camadas também favorecem o atributo de qualidade **Manutenibilidade**, pois permitem que os componentes possam ser facilmente substituídos caso haja necessidade (modularidade), devido a divisão das responsabilidades.

Além disso, os estilos arquiteturais cliente-servidor e camadas também favorecem o atributo de qualidade **Manutenibilidade**, pois permitem que os componentes possam ser facilmente substituídos caso haja necessidade (modularidade), devido a divisão das responsabilidades.

Portanto, definimos como os atributos de qualidade, que serão priorizados durante o desenvolvimento da arquitetura que será implementada, a Confiabilidade e a Manutenibilidade, respectivamente.

3. Representação da Arquitetura

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

Ponto de vista	Visão	Diagrama(s)

Projetista	Desenvolvimento	Componentes
Implantador	Física	Implantação

Os tópicos a seguir detalham esses pontos de vista arquiteturais, bem como as visões e os metamodelos utilizados para representá-los.

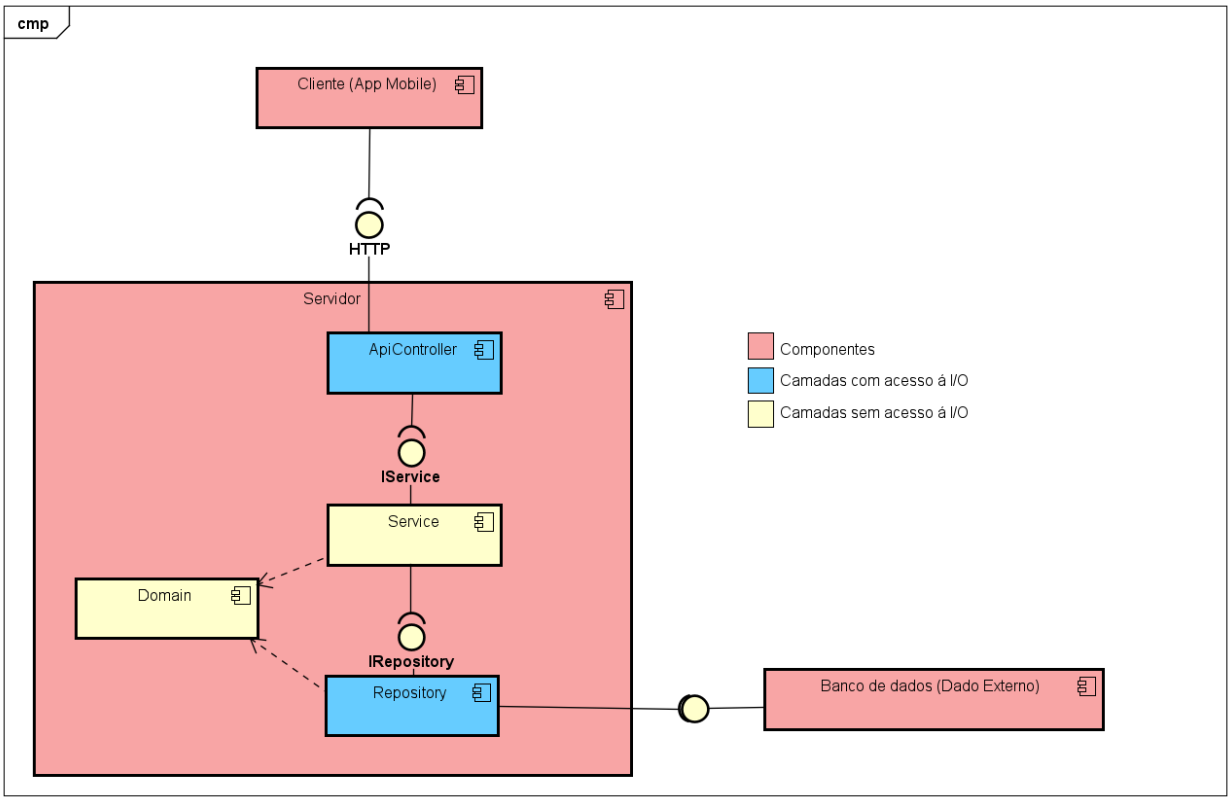
4. Ponto de vista do Projetista

4.1. 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: Um componente cliente, um componente servidor, um componente responsável por gerenciar o banco de dados e o um componente Arduino por onde são coletados os dados meteorológicos.

4.2. Visão de Componente



Metamodelo do diagrama de componentes está localizado no **Anexo B - Metamodelo de Componentes**

O componente *cliente* é responsável por interagir diretamente com o usuário, ela é será através do celular. Além disso, é responsável por realizar a comunicação com o componente servidor por meio de requisições HTTP.

O componente *servidor* é responsável pela implementação lógica presente no software, possuindo um conjunto de camadas internas para a segregação das responsabilidades. Cada camada independente, sendo responsável por garantir a segurança e execução de suas próprias funcionalidades, sendo acessadas apenas pela camada exterior e possuindo acesso apenas à camada interior, exceto camadas com acesso direto a dispositivos de entrada / saída. Também percebemos que foi dividido em dois servidores para que seja possível manter a disponibilidade do serviço.

O componente *Banco de dados* é uma representação dos dados externos ao software responsáveis por disponibilizar e armazenar os dados necessários para o funcionamento do sistema.

4.3. Detalhamentos das Camadas

- *ApiController* - Irá conter as chamadas das nossas APIs. Ela comunica apenas com a camada interior *Service;*
- *View:* é responsável por prover uma interface para acesso ao servidor, seja pela exibição de uma página html ou pela utilização de outra tecnologia. A camada View se comunica apenas com a camada *Controller.*
- *Controller:* interpreta as entradas do mouse ou do teclado enviado pelo usuário e mapeia essas ações do usuário em comandos que são enviados para para a *View* para efetuar a alteração apropriada. A camada *Controller* se comunica internamente apenas com *View* e *Service.*
- *Service*: Onde será realizada a regra de negócio em caso necessário. Essa camada se comunica internamente apenas com o Repository e externamente com o Arduino.
- *Domain*: Irá conter as entidades do nosso sistema (Domínios Ricos);
- *Repository*: Onde será realizado às consultas/inserções às informações armazenadas.

5. Ponto de vista do Implantador

5.1. Visão Geral

A visão de implantação é direcionada à equipe de implantação e é responsável por definir ferramentas e ambientes necessários para o funcionamento do software. Ela foi escolhida por se tratar de um sistema que integra componentes independentes e modularizados que devem trabalhar concorrentemente, sendo esta modularização representada principalmente pelas placas Arduíno que serão utilizadas para a captura dos índices e que alimentarão a base de dados do sistema.

5.2. Visão Física

Conforme a visão geral da perspectiva do projetista (5.1), há quatro componentes essenciais que devem ser considerados pela visão arquitetural: um componente cliente, um componente servidor, um componente gerenciador de banco de dados e um componente Arduino. Todos esses componentes são os nós-físicos do software, cada um com suas configurações e ferramentas próprias necessárias para o funcionamento.

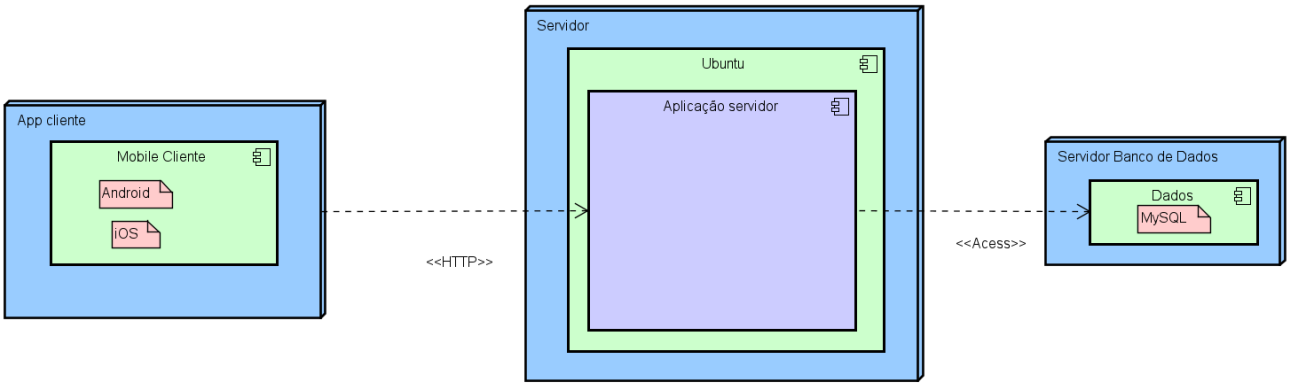
5.3. Detalhamento dos nós-físicos

Os nós-físicos do cliente é o computador do próprio usuário (desktop ou mobile) e para seu funcionamento é necessário a instalação dos aplicativos que serão disponibilizados nas lojas oficiais de aplicativos do Android e iOS.

O nó-físico do servidor é uma máquina, física ou virtual, com sistema operacional Ubuntu e com o Nginx como servidor HTTP.

O nó-físico de dados é uma abstração das aplicações externas ao software que são responsáveis por disponibilizar os dados utilizados pelo software. Será utilizado o banco MySQL.

A interação entre os nós-físicos citados pode ser visualizada por meio do Diagrama de Implantação UML abaixo:



Metamodelo do diagrama de implantação está localizado no **Anexo D - Metamodelo de Implantação**

Anexo A - Metamodelo de Casos de Uso *

**(Este metamodelo é apenas uma citação, o original se encontra na documentação do software Sempre UFG)*