FitPersonal

Arquitetura de *Software*

Versão 1.0

Histórico de Revisões

Versão	Descrição	Data	Autor
1.0	Desenvolvimento do documento de Arquitetura de *Software*.	31/08/2024	Dayany Lima Deise Santana Graciely Duarte Lorena Avelino Warley Ramires

Índice Analítico

Documento de Arquitetura de *Software*

1. Introdução

O ***FitPersonal*** é uma plataforma voltada para a gestão personalizada de treinos e planos alimentares, permitindo que os usuários acompanhem e ajustem suas rotinas de maneira prática e eficiente. A plataforma é projetada para atender a três tipos principais de usuários: **Alunos**, **Personais** e **Nutricionistas**.

Os **Alunos** podem utilizar o aplicativo para monitorar seus treinos e planos alimentares, recebendo orientações personalizadas para melhorar seu desempenho físico e alcançar suas metas de saúde. Os **Personais**, por sua vez, têm a capacidade de criar e gerenciar os treinos dos Alunos, personalizando cada rotina de acordo com as necessidades individuais de seus clientes. Já os **Nutricionistas** podem prescrever planos alimentares personalizados para os Alunos, alinhando a dieta aos objetivos de cada pessoa, seja para ganho de massa, emagrecimento ou manutenção da saúde.

Ao integrar funcionalidades de acompanhamento em tempo real, o ***FitPersonal*** permite que todas as categorias de usuários mantenham uma comunicação contínua e ajustem seus planos, proporcionando uma experiência completa e personalizada para o gerenciamento de saúde e bem-estar.

1.1 Finalidade

Este documento oferece uma visão abrangente da arquitetura do Aplicativo de Monitoramento *Fitness*. Ele captura e comunica as decisões arquiteturais significativas tomadas para garantir que o sistema atenda aos requisitos funcionais e não funcionais estabelecidos. O público-alvo deste documento inclui desenvolvedores, arquitetos de *software*, analistas de sistemas e gerentes de projeto, que usarão o documento para guiar o desenvolvimento e garantir a conformidade com a arquitetura proposta.

1.2 Escopo

Este documento aborda a arquitetura de *software* do Aplicativo de Monitoramento Fitness, incluindo suas funcionalidades principais, como monitoramento de atividades físicas e nutricionais, gerenciamento de planos alimentares, treinos, notificações, e a integração com dispositivos móveis e periféricos. Ele é aplicável ao desenvolvimento do sistema e às decisões arquiteturais que impactam sua implementação.

1.3 Definições, Acrônimos e Abreviações

- **MVC:** *Model-View-Controller*
- **UC:** Caso de Uso
- **RAM**: Memória de Acesso Aleatório
- **FAQ**: Perguntas Frequentes
- **HTTP**: *Hypertext Transfer Protocol*
- **Wi-Fi**: *Wireless Fidelity*
- **DTO's**: *Data Transfer Objects*
- **3G**: Terceira geração da tecnologia de dados
- **4G**: Quarta geração da tecnologia de dados
- **5G**: Quinta geração da tecnologia de dados
- **API:** *Application Programming Interface*
- **iOS:** *iPhone Operating System*

1.4 Referências

Documentação do Projeto:

- Visão de Negócio (DVN001)
- Casos de Uso (DCU002)
- Diagrama de Casos de Uso (DGCU002)
- Diagrama Modelo de Domínio (DGMD003)
- Diagrama de Sequência (DGS004)
- Diagrama de Classe (DGC005)
- Arquitetura de *Software* (ARS006)

1.5 Visão Geral

Este documento está organizado em seções que cobrem diferentes aspectos da arquitetura do sistema. A Seção 2 descreve a representação arquitetural do sistema. A Seção 3 discute as metas e restrições da arquitetura. As Seções 4 a 9 detalham as várias visões arquiteturais, incluindo Casos de Uso, Lógica, Processos, Implantação, Implementação e Dados. As Seções 10 e 11 abordam os aspectos de Tamanho, Desempenho e Qualidade do sistema.

2. Representação Arquitetural

A arquitetura do Aplicativo de Monitoramento *Fitness* segue o padrão MVC, garantindo uma separação clara entre a *interface* do usuário, a lógica de negócios e a gestão de dados. As principais visões arquiteturais incluem a Visão de Casos de Uso, Visão Lógica, Visão de Processos, Visão de Implantação e Visão de Implementação, cada uma delas abordando aspectos críticos do *design*, funcionamento e distribuição do sistema.

Na camada *Model*, são definidas as entidades que representam os principais objetos do domínio, como Aluno, Personal, Nutricionista, Treino e PlanoAlimentar. Essas classes, localizadas na pasta *entities*, refletem a estrutura de dados do sistema. Além disso, são utilizados DTOs (*Data Transfer Objects)* para transferir informações entre as camadas de forma eficiente, facilitando o tráfego de dados sem expor diretamente às entidades.

A camada *Controller* lida com as interações externas, processando as requisições que chegam do *frontend* e retornando as respostas apropriadas. Localizados na pasta *controllers*, os controladores são responsáveis por gerenciar as operações relacionadas às entidades, tratando as requisições HTTP e invocando os serviços necessários para cumprir as regras de negócio.

A lógica de negócios propriamente dita é implementada na camada *Service*, onde estão centralizadas as operações mais complexas que envolvem as entidades. As classes de serviço, armazenadas na pasta *services*, são responsáveis por processar os dados, aplicar validações e garantir que as regras de negócio sejam respeitadas em cada operação antes que qualquer mudança seja realizada no banco de dados ou retornada ao usuário.

A persistência dos dados é gerenciada pela camada *Repository*, que cuida do acesso ao banco de dados. Localizadas na pasta *repositories*, as classes de repositório fornecem uma *interface* consistente para leitura e gravação de dados, encapsulando toda a lógica de interação com o banco e garantindo a separação entre a lógica de negócio e as operações de armazenamento.

Isso permite que o desenvolvimento aconteça de maneira organizada e clara. Garantindo que as responsabilidades de cada camada sejam atendidas para contribuir na manutenção e escalabilidade da aplicação.

3. Metas e Restrições da Arquitetura

3.1 Requisitos e Objetivos

- **Segurança**: Garantir que os dados dos usuários sejam protegidos, com suporte a autenticação segura e criptografia de dados sensíveis.
- **Portabilidade:** Suporte a múltiplos dispositivos e sistemas operacionais (iOS 12+ e Android 8.0+).
- **Reusabilidade:** Estruturar o código para facilitar a reutilização em componentes futuros.
- **Escalabilidade:** Capacidade de suportar um número crescente de usuários sem degradação do desempenho.

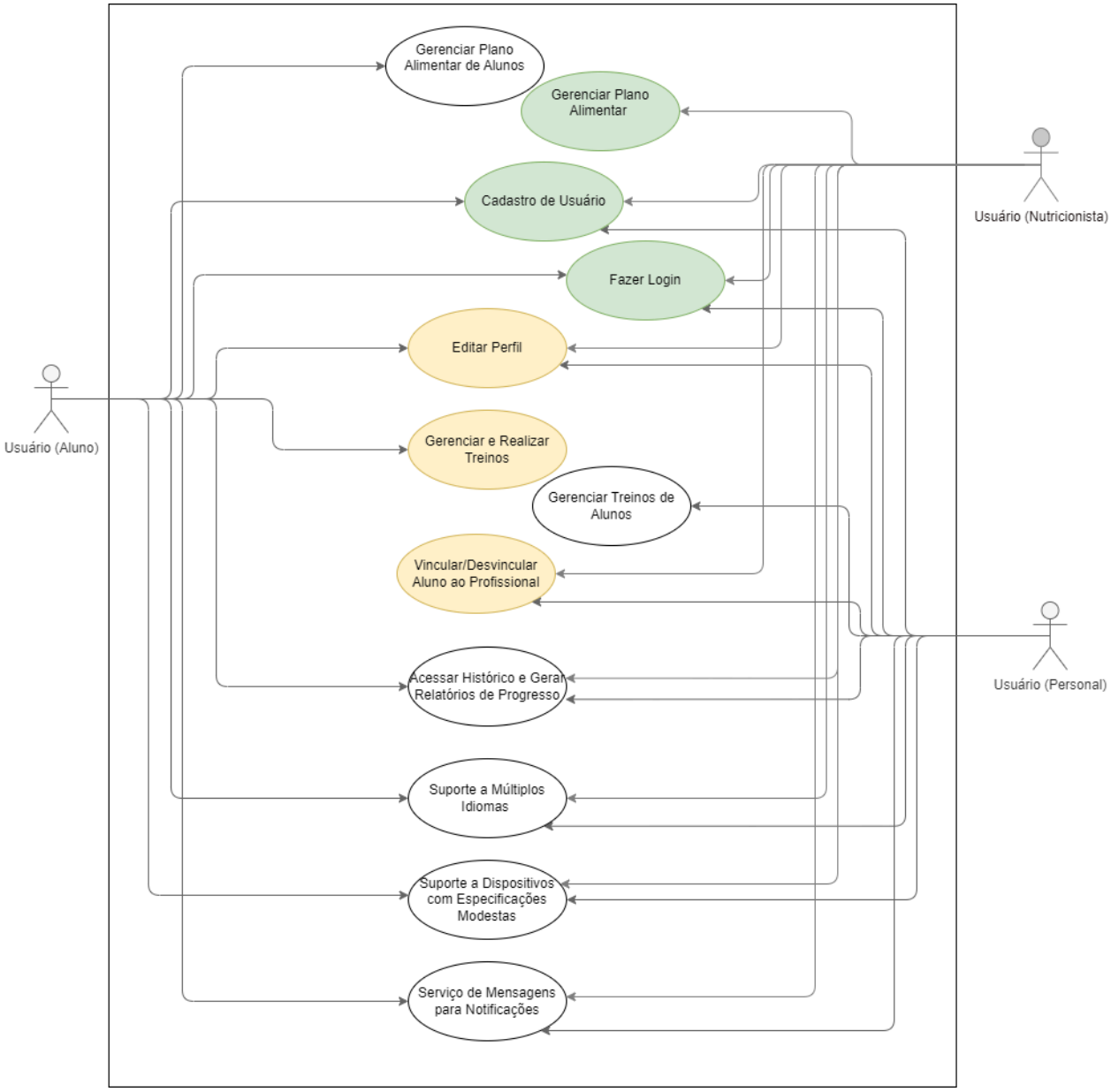
3.2 Restrições Especiais

- **Estratégia de *Design*:** Adoção do padrão MVC e práticas ágeis de desenvolvimento.
- **Ferramentas de Desenvolvimento:** Uso de *frameworks* e ferramentas que suportem integração contínua e desenvolvimento ágil.
- **Cronograma:** Implementação dentro de prazos restritos, exigindo entregas incrementais.

4. Visão de Casos de Uso

Os detalhes de cada caso estão disponíveis no documento Casos de Uso (DCU002), com seus fluxos ilustrados no Diagrama de Sequência (DGS004) e o modelo de domínio correspondente no Diagrama Modelo de Domínio (DGMD003).

Na imagem a seguir listamos os casos de uso do sistema, cujo os casos de uso em verde foram os que já estão totalmente prontos, os que estão em amarelo estão em desenvolvimento.



4.1 Realizações de Casos de Uso

A visão lógica do sistema é projetada para fornecer uma estrutura organizada e coerente para os principais componentes do aplicativo. A arquitetura foi desenvolvida seguindo o padrão MVC (*Model-View-Controller*) para garantir uma separação clara entre a lógica de apresentação, a lógica de negócio e a manipulação de dados. A estrutura lógica é dividida em três camadas principais:

Camada de Apresentação (*View*): Responsável por gerenciar a *interface* do usuário, esta camada interage diretamente com o usuário, exibindo as informações necessárias e capturando as entradas. Aqui, os usuários podem realizar ações como cadastro (UC01), *login* (UC09), edição de perfil (UC02), e navegação entre funcionalidades como gerenciar e realizar seus treinos (UC03), gerenciar treinos de seus alunos (UC04), gerenciar seu plano Alimentar (UC05) e gerenciar plano alimentar de seus alunos (UC06).

Camada de Controle (*Controller*): Esta camada atua como intermediária entre a camada de apresentação e a camada de negócio, recebendo as entradas dos usuários e decidindo quais ações tomar. Por exemplo, ao tentar realizar *login* (UC09), o controlador verifica as credenciais fornecidas, comunica-se com a camada de modelo para validar as informações, e dependendo do resultado, autentica o usuário ou exibe uma mensagem de erro. Controladores especializados são criados para funcionalidades específicas, como o gerenciamento e realização de treinos (UC03), gerenciar treinos de seus alunos (UC04), gerenciar seu plano Alimentar (UC05) e gerenciar plano alimentar de seus alunos (UC06).

Camada de Negócio (*Model*): Aqui reside a lógica de negócio do sistema, onde os dados são processados e as regras de negócio são aplicadas. Esta camada gerencia as entidades do sistema, como Usuário, Treino, Plano Alimentar, entre outras, garantindo que todas as operações realizadas, como a criação de novos treinos ou a vinculação de alunos a profissionais (UC10), sejam consistentes e sigam as regras estabelecidas. As entidades também armazenam e manipulam os dados persistentes, refletindo as ações tomadas pelos usuários, como no caso da atualização do perfil (UC02) ou no gerenciamento de históricos e relatórios de progresso (UC08).

4.2 Interações e Comunicação entre Componentes

A comunicação entre as diferentes camadas segue um fluxo bem definido:

- A Camada de Apresentação captura as ações do usuário e as encaminha para a Camada de Controle.
- A Camada de Controle processa a solicitação, validando-a se necessário e acionando a Camada de Negócio para realizar as operações necessárias.
- A Camada de Negócio manipula as entidades conforme necessário e retorna os resultados para a Camada de Controle, que então decide a resposta adequada a ser exibida ao usuário.

Por exemplo, ao editar um perfil (UC02), a solicitação do usuário é recebida pelo controlador de perfil, que então interage com a entidade de Usuário na camada de modelo para atualizar as informações. Após a atualização, o controlador confirma a operação e exibe uma mensagem de sucesso na *interface* do usuário.

4.3 Fluxos Alternativos e Extensões

Os fluxos alternativos e extensões dos casos de uso são também suportados por essa arquitetura. Por exemplo, se durante o cadastro (UC01) o usuário optar por se cadastrar como *Personal* ou Nutricionista, a camada de controle ajusta o fluxo, solicitando informações adicionais como o Registro Profissional, antes de prosseguir.

Da mesma forma, se ocorrerem erros durante o processo, como uma falha na criação de conta (UC01) ou na atualização de um plano alimentar (UC05), a camada de controle é responsável por interceptar o erro e encaminhar a resposta adequada para a *interface*, garantindo que o usuário seja informado e possa tentar novamente ou tomar outra ação.

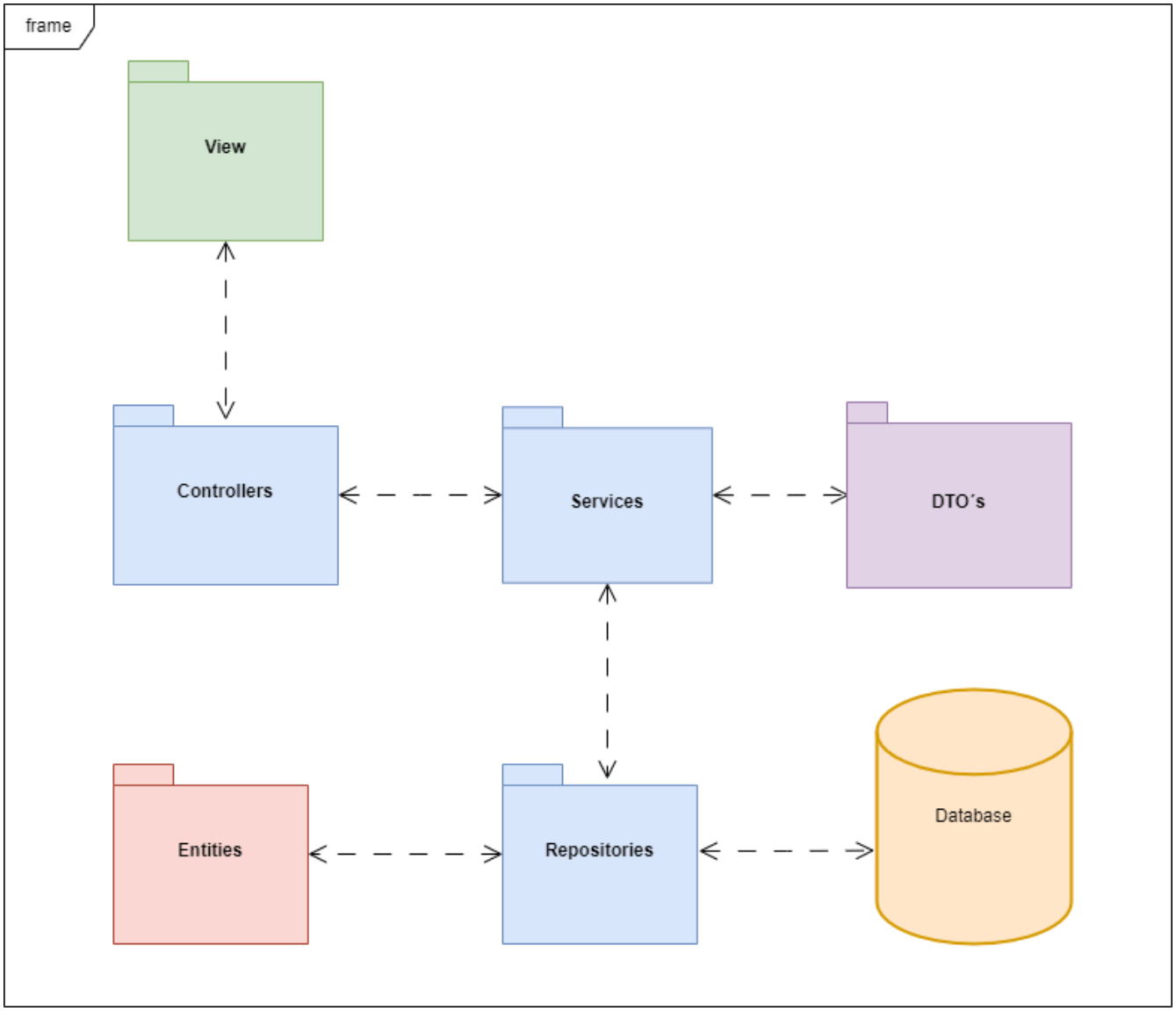
4.4 Considerações Especiais

A arquitetura lógica também acomoda requisitos específicos, como o suporte a múltiplos idiomas (UC07), onde a camada de controle verifica a preferência de idioma do usuário e ajusta a *interface* conforme necessário. Além disso, o Serviço de Mensagens para Notificações (UC11) é implementado para gerenciar a comunicação assíncrona entre o sistema e os usuários, enviando notificações conforme as preferências definidas. Essa estrutura lógica e bem definida garante que o sistema seja escalável, modular e fácil de manter, permitindo a adição de novas funcionalidades e a

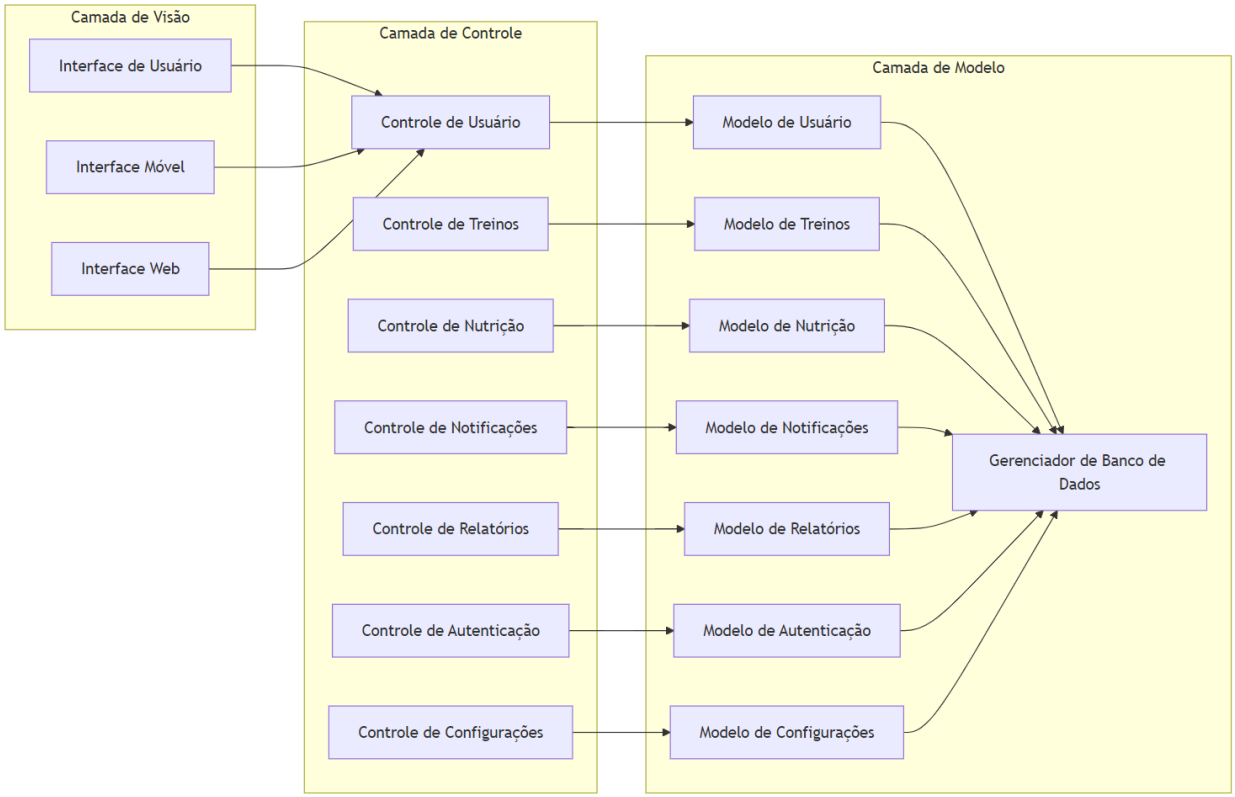
evolução contínua do *software* para atender às necessidades dos usuários.

5. Visão Lógica

A Visão Lógica descreve a estrutura da arquitetura do sistema em termos de seus subsistemas e pacotes. Essa visão organiza o modelo de *design* em camadas e pacotes, como demonstrado abaixo, destacando as principais classes e utilitários de classe que são essenciais para a arquitetura do sistema. A organização em camadas segue a arquitetura MVC (Model-View-Controller), facilitando a separação de responsabilidades e a modularidade do código.



5.1 Visão Geral



A camada de controle gerencia a lógica de negócios, a camada de visão cuida da *interface* do usuário, e a camada de modelo lida com a persistência de dados.

Camada de Visão (*View*): A Camada de Visão é responsável pela interação direta com o usuário, fornecendo interfaces amigáveis e intuitivas para diferentes plataformas:

- Interface de Usuário (*UI*): Componentes gerais da interface que podem ser reutilizados entre plataformas.
- Interface Móvel (*MobileUI*): Componentes específicos para aplicativos móveis em iOS e Android.
- Interface Web (*WebUI*): Componentes específicos para a versão web do aplicativo.

Relações: As interfaces da camada de visão interagem diretamente com os Controladores correspondentes na camada de controle para enviar solicitações e exibir dados.

Camada de Controle (*Controller*): A Camada de Controle contém a lógica de negócios do

aplicativo, processando as solicitações vindas da camada de visão e interagindo com a camada de modelo para acessar e manipular dados. Principais Pacotes:

- Controle de Usuário (UsuarioController): Gerencia operações relacionadas ao perfil e gerenciamento de usuários.
- Controle de Treinos (TreinoController): Gerencia a criação, edição e acompanhamento de planos de treino.
- Controle de Exercício (ExercicioController): Gerencia a criação, alteração e exclusão de um exercício.
- Controle de Plano Alimentar (PlanoAlimentarController): Gerencia planos alimentares e acompanhamento nutricional.
- Controle de Refeição (RefeicaoController): Gerencia a criação, alteração e exclusão de uma refeição.
- Controle de Login (LoginController): Gerencia processos de login, logout e segurança de acesso.

Relações: Cada controlador se comunica com seu respectivo Modelo na camada de modelo para realizar operações de dados.

Camada de Modelo (*Model*) ou Entidades(*Entities*): A camada de Modelo é responsável pelo gerenciamento de dados, incluindo a definição de estruturas de dados e a comunicação com o banco de dados.Principais Pacotes:

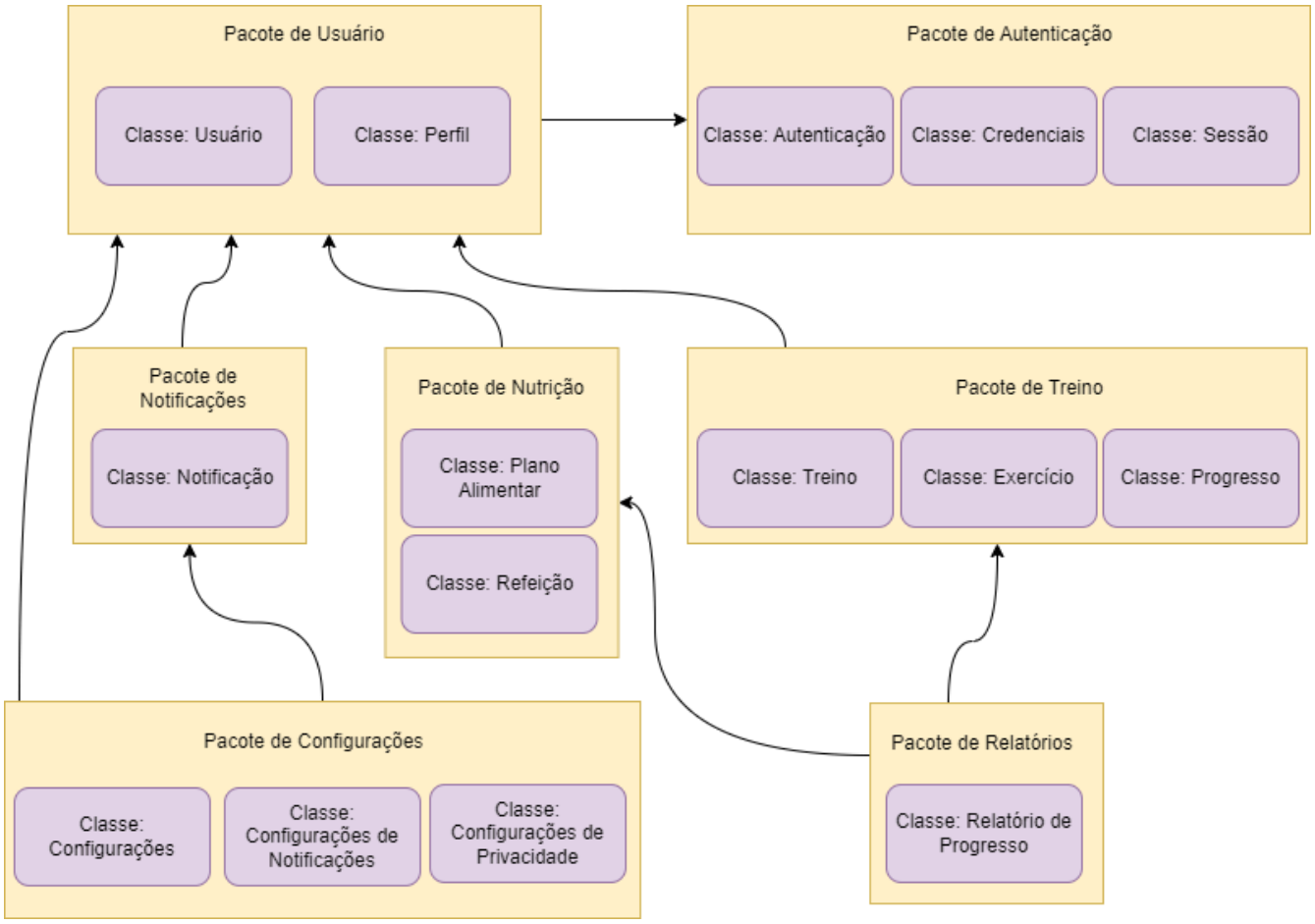
- Modelo de Usuário (Usuário): Representa dados e operações relacionadas aos usuários.
- Modelo de Treinos (Treino): Representa dados e operações relacionadas aos planos de treino.
- Modelo de Personal(Personal): Representa dados dados e operações relacionadas ao Personal
- Modelo de Refeição(Refeição): Representa dados dados e operações relacionadas a uma Refeição
- Modelo de Nutricionista(Nutricionista): Representa dados dados e operações relacionadas a um Nutricionista
- Modelo de Nutrição (PlanoAlimentar): Representa dados e operações relacionadas aos planos alimentares.
- Modelo de Notificações (Notificações): Representa dados e operações relacionadas às

notificações.

- Modelo de Relatórios (Relatório): Representa dados e operações para geração e armazenamento de relatórios.
- Modelo de Autenticação (Autenticação): Representa dados e operações relacionadas à autenticação e segurança.
- Modelo de Configurações (Configurações): Representa dados e operações relacionadas às configurações do aplicativo.
- Gerenciador de Banco de Dados (*Repositores*): Abstrai a interação direta com o banco de dados, permitindo operações de armazenamento, atualização, exclusão e recuperação de dados.

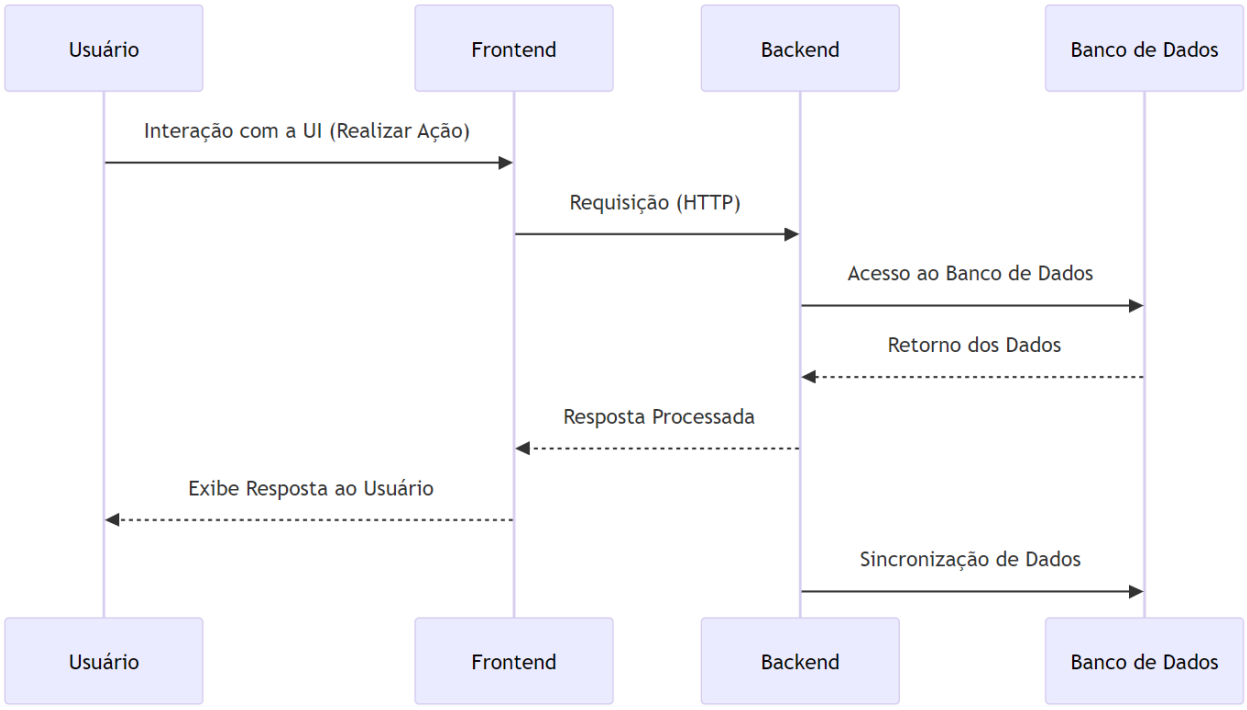
5.2 Pacotes de *Design* Significativos do Ponto de Vista da Arquitetura

Nesta subseção, cada pacote significativo do ponto de vista arquitetural é listado no diagrama em termos de suas classes.



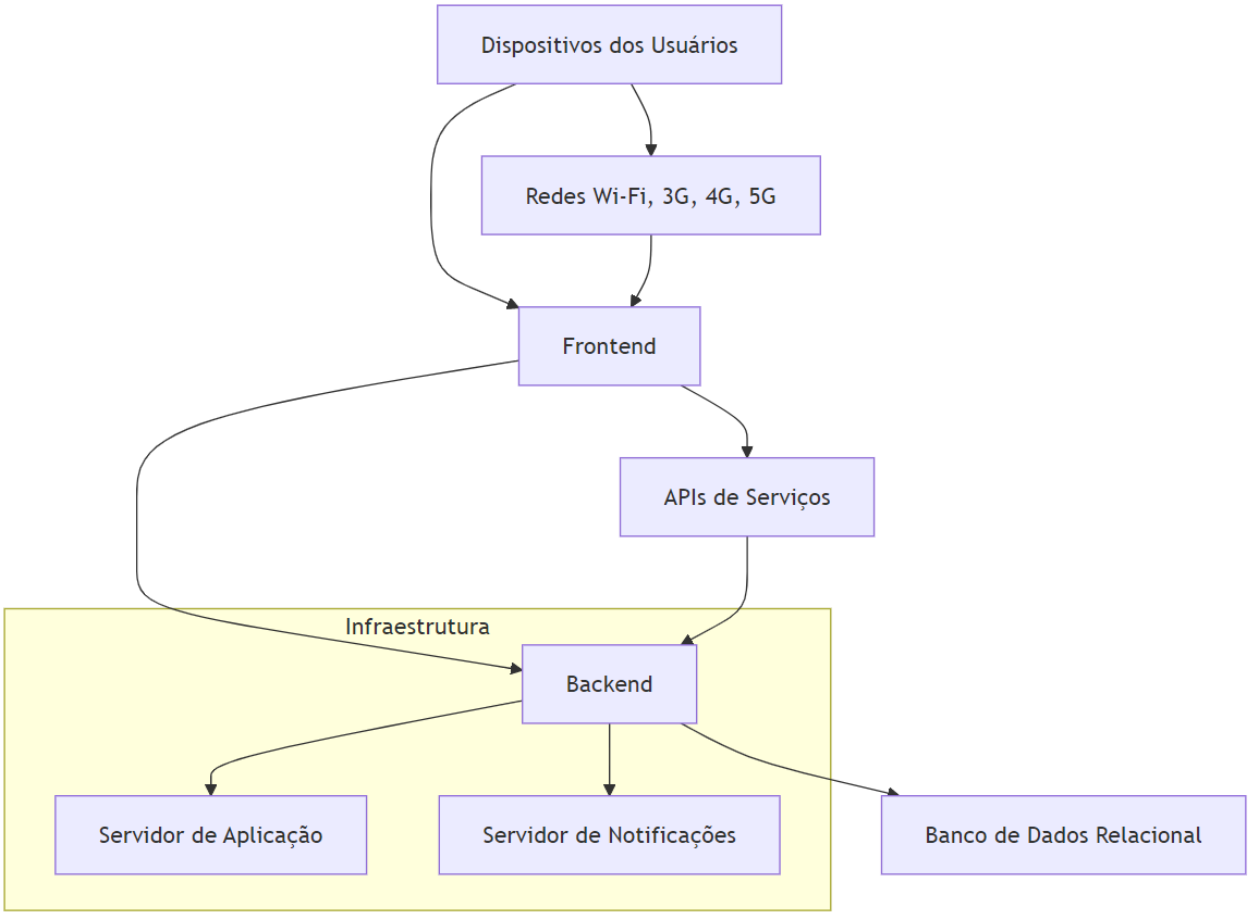
6. Visão de Processos

O sistema é composto por processos leves que gerenciam a interação do usuário em tempo real e processos pesados para tarefas como sincronização de dados e processamento de informações de monitoramento.



7. Visão de Implantação

O sistema será implantado em uma infraestrutura que suporte a execução em dispositivos móveis com diferentes sistemas operacionais, utilizando redes *Wi-Fi* e móveis (3G, 4G, 5G).



8. Visão da Implementação

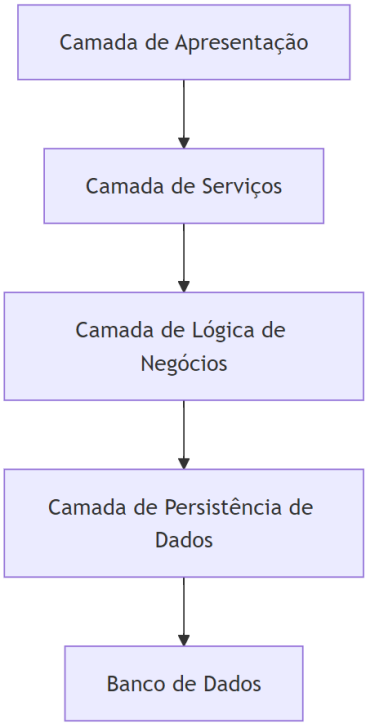
8.1 Visão Geral

O sistema será implementado em uma arquitetura em camadas, com uma separação nítida entre o *backend* e o *frontend*. Essa abordagem facilita a manutenção e a atualização de componentes individuais, permitindo que melhorias ou correções sejam realizadas isoladamente, sem impactar o restante do sistema. Além disso, essa divisão clara promove uma maior modularidade, tornando o sistema mais robusto e adaptável a futuras evoluções.

8.2 Camadas

- **Camada de Apresentação:** Implementada usando tecnologias responsivas para suportar múltiplas resoluções e dispositivos.

- **Camada de Serviços:** Inclui *APIs* que interagem com a lógica de negócios e persistência de dados.



9. Visão de Dados (opcional)

O sistema armazenará dados persistentes relacionados ao perfil do usuário, histórico de atividades e notificações em um banco de dados relacional. A conversão entre o modelo de *design* e o modelo de dados será direta e mapeada conforme as entidades do sistema.

10. Tamanho e Desempenho

O aplicativo deve ser otimizado para operar de maneira eficiente em dispositivos com 2GB de RAM, garantindo um tempo de resposta rápido, mesmo em condições de rede mais lentas, como em conexões 3G. A performance deve ser consistente, assegurando que o aplicativo funcione sem problemas e sem comprometer a experiência do usuário, independentemente das limitações de *hardware* ou da qualidade da conexão de rede.

11. Qualidade

A arquitetura do *software* foi projetada para ser altamente extensível, permitindo a adição de novas funcionalidades com impacto mínimo no sistema existente. Utilizamos a arquitetura MVC, que oferece excelente testabilidade, possibilitando que cada componente do *software* seja testado individualmente, sem a necessidade de testar o sistema como um todo. Isso simplifica a identificação e correção de problemas, garantindo maior confiabilidade e qualidade no produto final. Além disso, a estrutura clara e bem definida do MVC melhora a colaboração entre as equipes de desenvolvimento, permitindo que trabalhem em paralelo de forma mais eficiente, focando em suas áreas específicas de especialização. Isso resulta em maior produtividade, agilidade no desenvolvimento e um *software* de alta qualidade.