Projeto Pronto Afeto api

Trabalho de Conclusão de Curso

Vinícius Rodrigues dos Santos Moreira

Manoel Neto
Orientadora

Beltrano de tal
Coorientador

Instituto Federal da Bahia - IFBA
Curso de Análise e Desenvolvimento de Sistemas
Campus Salvador

Salvador, Bahia, Brasil
21 de janeiro de 2025

Sumário

1 Visão geral

1.1 Objetivo

Este projeto tem como objetivo desenvolver uma API REST voltada para o telecuidado de pessoas, com o intuito de conectar famílias a cuidadores de forma eficaz e apoiada pela Pronto Afeto. A plataforma será projetada para oferecer uma ampla gama de funcionalidades, possibilitando um gerenciamento de cuidados mais eficiente e transparente.

A API permitirá o cadastro completo e detalhado de cuidadores, cuidados e famílias, incluindo todos os membros familiares relacionados ao cuidado que está sendo monitorado. Através dessa API, será possível registrar informações cruciais, como o histórico completo de acompanhamento, ocorrências diárias e rotinas de cuidados, garantindo que todas as necessidades dos pacientes sejam atendidas de maneira organizada e acessível.

Além disso, a plataforma será equipada para gerenciar prontuários de cuidados, proporcionando uma visão abrangente e em tempo real do progresso e das necessidades de cada indivíduo. Com essa abordagem, a API não só facilita a conexão entre cuidadores e famílias, mas também assegura um acompanhamento contínuo e detalhado das atividades e dos cuidados prestados. [1]

1.2 Definições, Siglas e Abreviações

1. CLIENTE - Responsável financeiro pelo contratação
2. CUIDADO - Aquele a quem se destina a contratação.
3. CUIDADOR - Aquele que presta o serviço de acompanhamento ao CUIDADO.
4. PRONTUÁRIO - Registro periódico de anotações relacionadas a um CUIDADO e que servem para acompanhamento do CLIENTE da saúde e bem estar do CUIDADO e das intervenções feitas pelo CUIDADOR

1.3 Declaração do Problema

A solução de software desenvolvida para este projeto é uma API REST robusta e modular, projetada para facilitar o gerenciamento de cuidados a distância, integrando famílias e cuidadores de maneira eficiente. A API permite a criação de um ecossistema colaborativo em que as famílias podem monitorar o cuidado recebido por seus entes queridos em tempo real, enquanto os cuidadores podem registrar e compartilhar informações cruciais de maneira organizada e acessível.

Um dos principais aspectos da solução é a capacidade de gerenciamento completo e detalhado de dados. Cuidadores podem cadastrar informações pessoais e profissionais, além de manter um registro contínuo das atividades diárias e ocorrências no cuidado prestado. Da mesma forma, os familiares têm acesso a esses dados, o que assegura transparência e visibilidade em todas as etapas do acompanhamento. A API também permite o registro de

prontuários de cuidados, oferecendo uma visão consolidada das necessidades e progressos dos pacientes. Alguns problemas que o sistema pode resolver seriam:

1. Falta de Transparência no Cuidado Remoto: Em muitos casos, as famílias têm pouca visibilidade sobre o cuidado prestado aos seus entes queridos. A API propõe resolver essa questão ao permitir que os familiares acompanhem, em tempo real, as atividades e o bem-estar do paciente, proporcionando maior confiança e segurança.

2. Dificuldade em Registrar e Organizar Informações de Cuidado: O registro de informações importantes, como atividades diárias, ocorrências e histórico de cuidados, pode ser desorganizado ou incompleto. A API facilita o gerenciamento desses dados, permitindo que cuidadores registrem atividades de forma organizada e consistente.

3. Falta de Integração e Comunicação entre Cuidadores e Famílias: A comunicação eficaz entre cuidadores e familiares é crucial para o bem-estar do paciente. O projeto resolve essa necessidade ao criar um ecossistema colaborativo que centraliza a comunicação e promove o alinhamento de informações entre as partes interessadas.

4. Gerenciamento Ineficiente de Dados de Pacientes: A API permite o gerenciamento completo e detalhado dos dados dos pacientes, incluindo histórico de cuidados e prontuários, proporcionando uma visão abrangente e consolidada que facilita o acompanhamento da saúde e o progresso de cada paciente.

5. Dificuldade em Monitorar o Progresso do Paciente e Identificar Necessidades Emergentes: Sem acesso fácil a informações de prontuários e ocorrências diárias, pode ser desafiador avaliar o progresso e as necessidades emergentes de um paciente. A solução proposta resolve isso com relatórios atualizados que ajudam familiares e cuidadores a identificar rapidamente mudanças e adaptar os cuidados conforme necessário.

6. Ausência de Documentação e Continuidade no Cuidado: Em situações de troca de cuidador, a falta de documentação clara pode comprometer a continuidade dos cuidados. Com a API, o histórico e as informações do paciente ficam centralizados e acessíveis, facilitando a transição entre cuidadores e garantindo a continuidade dos cuidados.

7. Necessidade de Gerenciamento Centralizado e Acessível de Dados Sensíveis: Cuidadores e familiares precisam de uma plataforma segura e acessível para gerenciar informações sensíveis dos pacientes. A API oferece uma solução centralizada com medidas de segurança robustas, como autenticação e autorização, garantindo a proteção dos dados pessoais e de saúde.

1.4 Proposta de Solução de Software

conectar cuidadores, famílias e a empresa Pronto Afeto de forma eficiente e organizada. A API será o núcleo central do sistema, permitindo o gerenciamento completo dos dados relacionados aos cuidados prestados, incluindo o cadastro de cuidadores, famílias e informações detalhadas dos pacientes sob supervisão.

Concebida para atender à crescente demanda por telecuidado, a API oferece uma plata-

forma que facilita o acompanhamento contínuo e em tempo real das atividades de cuidado, gerando relatórios completos sobre rotinas diárias, ocorrências e prontuários de cuidados. Esse nível de detalhamento garante que todos os envolvidos no processo – tanto os familiares quanto os cuidadores – possam acessar dados atualizados e tomar decisões informadas rapidamente.

Além disso, a API proporciona transparência e segurança nos processos de gestão, assegurando que todas as informações sensíveis, como dados médicos e pessoais, estejam protegidas por camadas de segurança como autenticação via tokens JWT e criptografia de dados. A flexibilidade da arquitetura da solução permite a integração com outras plataformas, possibilitando expansões futuras e maior adaptabilidade a novos requisitos.

Um diferencial importante da solução é o uso de geocoding, que permite conectar cuidadores e clientes com base na proximidade geográfica. Esse recurso facilita a busca por cuidadores que estejam próximos do local do cliente, otimizando a logística e garantindo um atendimento mais rápido e eficaz. Ao integrar geocoding à API, a plataforma melhora a eficiência na alocação de cuidadores, diminuindo o tempo de resposta e ampliando a satisfação tanto dos clientes quanto dos cuidadores.

Essa abordagem garante que o sistema seja capaz de resolver o problema da falta de visibilidade e comunicação entre as famílias e os cuidadores, oferecendo uma solução que facilita o monitoramento remoto de cuidados e melhora significativamente a experiência tanto dos pacientes quanto de seus familiares. A API torna-se assim uma ferramenta crucial para o acompanhamento detalhado e eficiente, respondendo à necessidade crescente de gestão de cuidados personalizados e conectados.

1.5 Tecnologias

1. Java 17: A versão 17 do Java foi escolhida por sua estabilidade e suporte de longo prazo (LTS). Ela oferece novas funcionalidades de linguagem e aprimoramentos de desempenho que tornam o desenvolvimento mais eficiente e seguro.

2. Spring Boot: O Spring Boot é o framework principal utilizado para construir a aplicação. Ele oferece uma estrutura simplificada para o desenvolvimento de APIs REST, além de uma configuração inicial fácil e rápida. As dependências principais de Spring Boot usadas no projeto são:

   (a) spring-boot-starter-web: Fornece o suporte necessário para a construção de APIs RESTful, incluindo a configuração do servidor web embutido (Tomcat) e as ferramentas essenciais para lidar com requisições HTTP.

   (b) spring-boot-starter-security: Essencial para implementar autenticação e autorização na API, garantindo que apenas usuários autorizados possam acessar os recursos.

   (c) spring-boot-starter-validation: Inclui as bibliotecas de validação necessárias para assegurar a integridade dos dados recebidos nas requisições, aplicando anotações para validação dos DTOs.

(d) spring-boot-devtools: Facilita o desenvolvimento com ferramentas como o auto-reload da aplicação, otimizando o processo de desenvolvimento e teste.

(e) spring-boot-starter-data-jpa: Fornece suporte para acesso e manipulação de dados, facilitando o uso do JPA (Java Persistence API) para realizar operações em banco de dados.

3. Lombok: O Lombok é utilizado para reduzir o boilerplate de código, gerando automaticamente getters, setters, construtores, entre outros. Ele aumenta a produtividade dos desenvolvedores ao reduzir a quantidade de código necessário para tarefas rotineiras.

4. flyway-core e flyway-database-postgresql: Flyway é uma ferramenta de migração de banco de dados que permite versionar e controlar alterações na estrutura do banco. Ele facilita a criação e manutenção de scripts de migração, garantindo consistência nos ambientes de desenvolvimento, teste e produção.

5. java-jwt: Utilizada para implementar autenticação com tokens JWT (JSON Web Token), essa biblioteca facilita a criação e validação de tokens, que são essenciais para garantir que apenas usuários autenticados possam acessar certos endpoints.

6. springdoc-openapi-starter-webmvc-ui: Essa dependência é usada para gerar automaticamente a documentação da API usando o padrão OpenAPI, o que torna a API mais acessível aos desenvolvedores e usuários. A integração com o Swagger UI permite visualizar e testar os endpoints diretamente no navegador.

7. slf4j-api: SLF4J é uma biblioteca de logging que permite um registro centralizado e consistente de logs, auxiliando no monitoramento e na depuração de problemas durante o desenvolvimento e em produção.

8. mockito-core e spring-security-test: Essas dependências são essenciais para a criação de testes automatizados. Mockito permite a criação de mocks para simular comportamentos e testar funcionalidades, enquanto o Spring Security Test oferece ferramentas específicas para testar endpoints protegidos por autenticação e autorização.

9. jackson-databind: Esta biblioteca é usada para a conversão de objetos Java em JSON e vice-versa, facilitando a serialização e desserialização de dados em requisições e respostas HTTP, o que é fundamental para APIs RESTful.

10. Thymleaf: O Thymeleaf é um mecanismo de template eficiente e flexível que será utilizado para gerar PDFs de contratos. Ele permite criar templates HTML dinâmicos que podem ser convertidos para PDF, facilitando a personalização de documentos com informações específicas dos usuários. Integrado ao Spring Boot, o Thymeleaf oferece uma maneira prática de gerar e formatar PDFs diretamente a partir da aplicação.

11. nominatim-api: A API do Nominatim permite geocodificação, ou seja, a conversão de endereços em coordenadas geográficas. Esse recurso pode ser útil para funcionalidades que envolvem localização, como a determinação de endereços de cuidadores e pacientes.

12. Postgres: O PostgreSQL é um sistema de gerenciamento de banco de dados relacional robusto e de código aberto. Ele será utilizado para armazenar e gerenciar de forma segura os dados dos usuários, cuidadores, e prontuários. A escolha do PostgreSQL se deve à sua confiabilidade, escalabilidade e suporte avançado para operações complexas de dados, garantindo o bom desempenho da aplicação em produção.

13. Docker: O Docker é uma plataforma de contêineres que será usada para empacotar e isolar a aplicação, facilitando o ambiente de desenvolvimento e a implantação. Ao encapsular todos os serviços necessários em contêineres separados, o Docker proporciona consistência e portabilidade, permitindo que o sistema seja executado em qualquer ambiente com o mínimo de configuração.

14. Docker Compose: O Docker Compose é uma ferramenta complementar ao Docker que simplifica a definição e o gerenciamento de múltiplos contêineres para a aplicação. Com ele, será possível configurar e orquestrar facilmente a API, o banco de dados PostgreSQL, e outras dependências, assegurando uma execução coordenada de todos os serviços em ambientes de desenvolvimento e produção.

15. Git: O Git é um sistema de controle de versão distribuído que permitirá o gerenciamento eficiente do código-fonte do projeto, facilitando a colaboração e o rastreamento de mudanças.

2 Requisitos funcionais

2.1 Funcionário

Nº	História de Funcionário	Módulo
RF01	Eu, como funcionário, desejo realizar login para acessar a área logada do sistema	Cadastro e Autenticação
RF02	Eu, como funcionário autenticado, desejo aprovar o cadastro de um cuidador no sistema para que ele tenha acesso à aplicação, notificando ele por email	Cadastro e Autenticação
RF03	Eu, como administrador autenticado, desejo acessar a página home para visualizar o status das propostas	Proposta
RF04	Eu, como funcionário autenticado desejo alterar o status de uma proposta, notificando o cliente	Proposta
RF05	Eu, como funcionário autenticado, desejo visualizar o histórico de atividades de um cuidador para monitorar seu desempenho e qualidade de atendimento	Monitoramento
RF06	Eu, como funcionário autenticado, desejo registrar ocorrências relacionadas a um cuidador para manter um histórico de incidentes ou feedbacks recebidos	Monitoramento

RF07	Eu, como funcionário autenticado, desejo visualizar uma lista de cuidadores disponíveis em uma determinada localização para facilitar a alocação de cuidadores próximos aos clientes	Alocação Geográfica
RF08	Eu, como funcionário autenticado, desejo criar relatórios mensais de atendimento para avaliar a qualidade e a frequência dos cuidados prestados	Relatórios e Estatísticas
RF09	Eu, como funcionário autenticado, desejo acessar o prontuário de um cuidado para verificar o andamento e a qualidade dos cuidados prestados	Prontuário
RF10	Eu, como funcionário autenticado, desejo enviar notificações para os cuidadores sobre atualizações ou lembretes de procedimentos	Comunicação
RF11	Eu, como funcionário autenticado, desejo visualizar o feedback de clientes sobre cuidadores para avaliar a satisfação com o serviço prestado	Avaliação e Feedback
RF12	Eu, como funcionário autenticado, desejo acessar um painel de controle com métricas gerais, como número de cuidadores ativos, clientes atendidos, e quantidade de ocorrências registradas	Painel de Controle
RF13	Eu, como funcionário autenticado, desejo cadastrar novos serviços ou atividades na plataforma para que cuidadores e clientes possam acessar informações atualizadas	Gerenciamento de Serviços
RF14	Eu, como funcionário autenticado, desejo revisar e editar informações de cuidadores e clientes para garantir que os dados estejam corretos e atualizados	Gerenciamento de Cadastro
RF15	Eu, como funcionário autenticado, desejo encerrar o cadastro de um cuidador ou cliente quando não for mais necessário para manter a base de dados organizada e atualizada	Gerenciamento de Cadastro
RF16	Eu, como funcionário autenticado, desejo poder alterar meu próprio perfil e configurações, como preferências de notificações e senha	Configurações de Usuário

2.2 Cliente

Nº	História Cliente	módulo
RF01	Eu, como cliente devo ser capaz de me cadastrar no sistema usando senha, email e nickname	Autenticação.
RF02	Eu, como cleinte devo ser capaz de me autenticar no sistema usando meu email e senha.	Autenticação.

RF03	Eu, como cleinte devo ser capaz de cadastrar um cuidado em meu nome.	Cuidado.
RF04	Eu, como cleinte logado devo ser capaz de fazer uma proposta para o sistema, onde serão passado os dados de cuidado, endereco e anamnese.	Proposta.
RF05	Eu, como cleinte logado devo ser capaz de acompanhar o status de minha proposta.	Proposta.
RF06	Eu, como cleinte logado devo ser capaz de cadastrar um cuidado em meu nome.	Cuidado.
RF07	Eu, como cleinte logado devo ser capaz de visuzalizar todos os dados dos meus cuidados.	Cuidado.
RF08	Eu, como cleinte logado devo ser capaz de visuzalizar todos os dados dos prontuários dos meus cuidados.	Cuidado.
RF09	Eu, como cliente logado, devo ser capaz de receber notificações sobre atualizações no status das minhas propostas e dos meus cuidados	Notificações
RF10	Eu, como cliente logado, devo ser capaz de avaliar o serviço prestado por um cuidador após o término do cuidado	Avaliação e Feedback
RF11	Eu, como cliente logado, devo ser capaz de visualizar o histórico de avaliações dos cuidadores disponíveis para escolher com base na experiência de outros clientes	Avaliação e Feedback
RF12	Eu, como cliente logado, devo ser capaz de atualizar minhas informações pessoais, como endereço, email e telefone	Gerenciamento de Conta
RF13	Eu, como cliente logado, devo ser capaz de solicitar a alteração de dados específicos de um cuidado já cadastrado, como endereço ou instruções especiais	Cuidado
RF14	Eu, como cliente logado, devo ser capaz de acessar um painel de resumo dos cuidados, incluindo quantidade de cuidados ativos, pendentes e concluídos	Painel de Controle
RF15	Eu, como cliente logado, devo ser capaz de cancelar uma proposta ou cuidado antes que eles sejam iniciados, com a opção de fornecer uma justificativa	Proposta
RF16	Eu, como cliente logado, devo ser capaz de visualizar um resumo financeiro com os custos associados aos meus cuidados e propostas	Financeiro

2.3 Cuidador

Nº	História de Cuidador	Módulo
RF01	Eu, como cuidador, desejo me cadastrar no sistema para poder oferecer meus serviços de cuidado.	Cadastro e Autenticação

RF02	Eu, como cuidador, desejo realizar login para acessar minha conta e visualizar as atividades disponíveis.	Cadastro e Autenticação
RF03	Eu, como cuidador autenticado, desejo atualizar meu perfil com informações como especialidades, experiência, endereço e disponibilidade, para que os clientes conheçam melhor meu trabalho.	Perfil do Cuidador
RF04	Eu, como cuidador autenticado, desejo visualizar as propostas de serviço recebidas para avaliar se estão dentro das minhas qualificações e aceitar ou recusar.	Proposta
RF05	Eu, como cuidador autenticado, desejo acompanhar o status das propostas que aceitei para saber se já foram confirmadas pelo cliente.	Proposta
RF06	Eu, como cuidador autenticado, desejo poder acessar as informações do cuidado sob minha responsabilidade para acompanhar as necessidades específicas e o histórico de anamnese.	Cuidado
RF07	Eu, como cuidador autenticado, desejo registrar observações e atualizações no prontuário do cuidado, para manter o cliente informado sobre o andamento do serviço.	Prontuário
RF08	Eu, como cuidador autenticado, desejo notificar o cliente em caso de emergências ou situações inesperadas, permitindo comunicação direta e rápida.	Comunicação
RF09	Eu, como cuidador autenticado, desejo visualizar o histórico de cuidados anteriores para auxiliar no tratamento e atendimento contínuo dos cuidados recorrentes.	Prontuário
RF10	Eu, como cuidador autenticado, desejo confirmar a conclusão de uma atividade ou cuidado, notificando o cliente para finalização do serviço.	Conclusão de Serviço
RF11	Eu, como cuidador autenticado, desejo visualizar uma agenda com meus serviços e compromissos, facilitando o gerenciamento de tempo e organização do trabalho.	Agenda

3 Requisitos não funcionais

Nº	Requisito Não Funcional	Descrição
RNF01	Desempenho	A API deve responder a 95% das requisições em menos de 200 ms.
RNF02	Escalabilidade	A API deve ser capaz de escalar horizontalmente para suportar um aumento no número de requisições.
RNF03	Disponibilidade	A API deve ter uma disponibilidade de 99.9% durante o ano.
RNF04	Segurança	A API deve usar HTTPS para todas as comunicações e implementar autenticação via OAuth 2.0.
RNF05	Manutenibilidade	O código da API deve ser modular e seguir as melhores práticas de programação para facilitar a manutenção e atualização.
RNF06	Documentação	A API deve ser documentada usando Swagger/OpenAPI para facilitar o entendimento e uso por outros desenvolvedores.
RNF07	Compatibilidade	A API deve ser compatível com as principais versões de navegadores e sistemas operacionais.
RNF08	Monitoramento	A API deve ser monitorada 24/7 usando ferramentas de monitoramento de desempenho e alertas.
RNF09	Logs	A API deve registrar todas as operações de entrada e saída para auditoria e análise posterior.
RNF10	Testabilidade	A API deve ser testável, com testes automatizados de unidade.
RNF11	Tolerância a Falhas	A API deve ser capaz de se recuperar automaticamente de falhas e continuar operando.
RNF12	Conformidade	A API deve estar em conformidade com os regulamentos de proteção de dados, como GDPR.
RNF13	Eficiência	A API deve ser otimizada para uso eficiente de recursos, como CPU e memória.
RNF14	Limites de Taxa	A API deve implementar limites de taxa para prevenir abuso e garantir a qualidade do serviço.

4 Visão Arquitetural

O sistema desenvolvido adota a arquitetura hexagonal, também conhecida como Arquitetura de Portos e Adaptadores. Este estilo arquitetural foi escolhido por sua flexibilidade e capacidade de desacoplar as camadas internas do domínio das interfaces externas, como bancos de dados, APIs e consumidores externos.

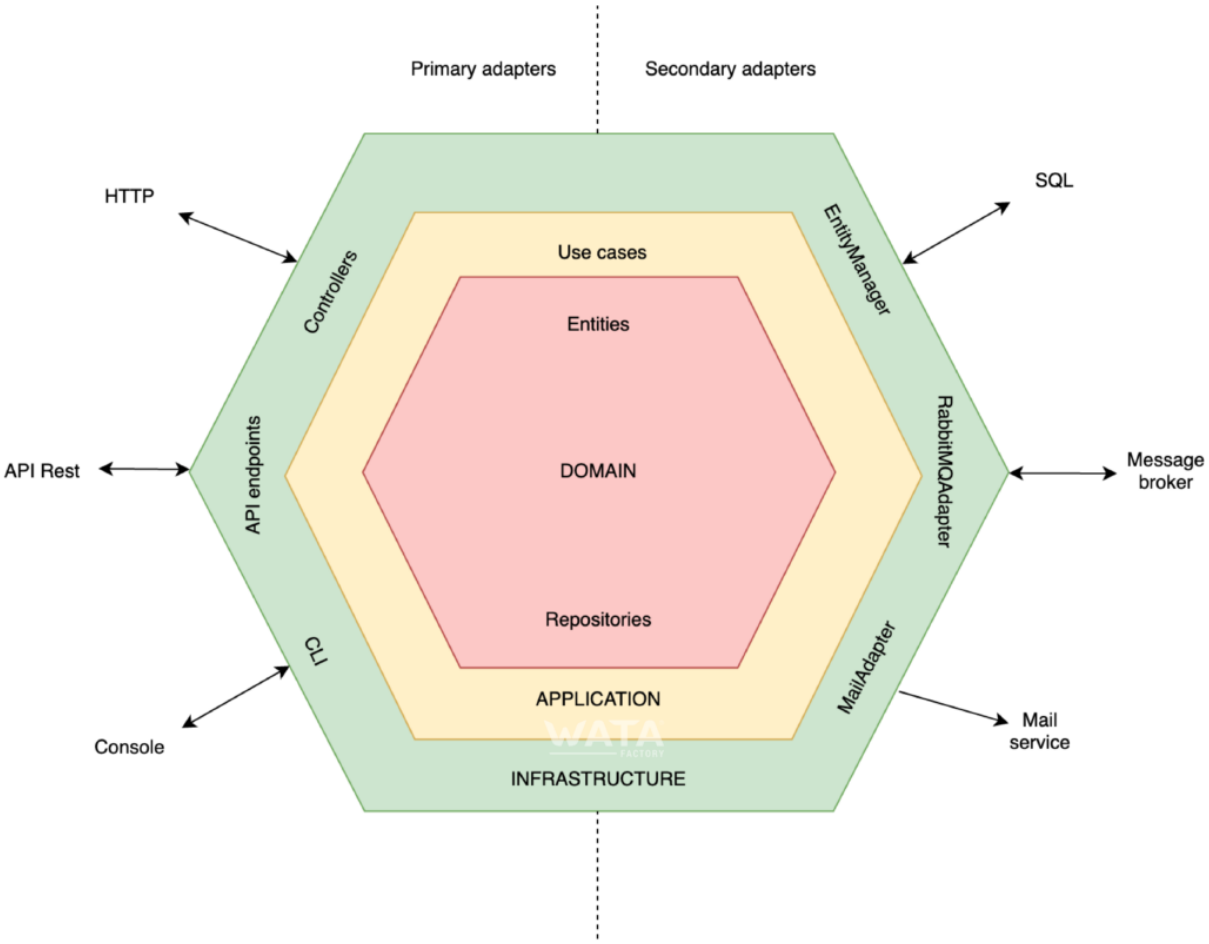

Figura 1: Diagrama ilustrando a arquitetura hexagonal. Fonte: adaptado de Wata Factory (2024).

4.1 Arquitetura do Sistema

A arquitetura do sistema é dividida em quatro camadas principais:

- **Core (Domínio):** Esta é a camada central e mais importante do sistema. Contém as regras de negócio, entidades e casos de uso. Nesta camada, as decisões não dependem de frameworks ou bibliotecas externas, garantindo que as regras de negócio sejam independentes das implementações tecnológicas.
- **Camada de Aplicação (Application):** Atua como uma ponte entre o Core (Domínio) e

os Adaptadores. Contém os casos de uso do sistema, definindo a lógica de orquestração necessária para que as regras de negócio sejam executadas corretamente. Essa camada organiza as chamadas às portas de entrada e saída.

- **Adaptadores de Entrada (Portos de Entrada):** São responsáveis por receber as interações externas e traduzi-las para o formato compreendido pelo domínio. No nosso sistema, utilizamos Spring Boot para implementar controladores REST, que servem como pontos de entrada para as requisições HTTP.
- **Adaptadores de Saída (Portos de Saída):** Gerenciam as integrações com serviços e recursos externos. Exemplos incluem repositórios que interagem com o banco de dados PostgreSQL utilizando Spring Data JPA.

4.2 Benefícios da Arquitetura Hexagonal

- **Flexibilidade:** O sistema é facilmente adaptável para novas interfaces (por exemplo, troca de banco de dados ou APIs externas).
- **Testabilidade:** A separação clara entre as camadas permite testar o domínio sem a necessidade de dependências externas.
- **Manutenção:** Com o domínio independente, alterações em tecnologias externas não afetam as regras de negócio.

5 Design

5.1 Projeto UML

[Insira os seguintes Diagramas de UML para o seu projeto:

1. Diagrama de Classe
2. Diagrama de Atividades da principal atividade do sistema
3. Diagrama de Sequência da principal atividade do sistema
4. Diagrama de Estado do principal objeto do sistema
5. Diagrama de Componentes
6. Diagrama de Implantação

OBS: Como cada sistema tem particularidades específicas, a escolha de qual a principal atividade e qual o principal objeto do sistema deve ser discutida com o orientador. Consideramos que existem alguns diagramas básicos que devem existir em qualquer projeto. São eles: Diagrama de Classes, Componentes e Implantação. A elaboração dos outros diagramas devem ser discutidos com seu orientador.]
[Obrigatório - A seção é obrigatória, os diagramas de classes e diagrama de casos de uso

são obrigatórios. Os outros diagramas são opcionais a depender do tipo do sistema a ser desenvolvido. Deve ser discutido com o orientador quais diagramas devem constar no trabalho.]

5.2 Visão arquitetural

[Descreva o estilo arquitetural que melhor se encaixa no seu sistema. Descreva a arquitetura de seu sistema, incluindo quais tecnologias utilizou na sua construção. Utilize figuras para ilustrar se necessário.]
[Obrigatório]

5.3 Modelo de Banco de Dados

[Descreva o modelo de Dados Físico e Lógico]
[Obrigatório - Somente se o sistema possui banco de dados.]

6 Qualidade

6.1 Projeto de testes

[Descreva em detalhes as estratégias de Testes que utilizou para desenvolvimento do seu software. Se utilizou casos de testes, testes automatizados ou manuais, ferramentas, etc.]
[Obrigatório]

7 Implantação

7.1 Projeto de implantação

[Descreva a Plataforma de Hardware e Software requeridas para instalação e operação do seu software.]
[Obrigatório]

8 Manual do Usuário

[Descreva o Manual de Usuário, como utilizar o seu software. Sugerimos que coloque figuras com as telas do seu sistema se necessário para melhorar o entendimento do usuário.]

[Obrigatório]

Agradecimentos

[Agradecimentos a colaboradores do seu projeto]
[Opcional]

Referências

1 FREITAS, C. C. et al. Uma Ferramenta Baseada em Algoritmos Genéticos para a Geração de Tabela de Horário Escolar. 2014.

[Forneça uma lista completa de todos os documentos mencionados ou que foram utilizados como referência na elaboração deste documento. Todos os documentos devem ser identificados por título, data, nome e organização responsável por sua publicação. Especifique as fontes dessas referências. Utilize o padrão ABNT para o formato das referências.] [Obrigatório]

[Dica: insira suas referencias no arquivo ”referencias.bib”e utilize o comando ”cite”para utilizá-la]