Relatório Técnico - Hairstyle

1. Introdução

O Hairstyle é um sistema desenvolvido para agendamento e gerenciamento de serviços de beleza e bem-estar. O objetivo é fornecer uma solução robusta e escalável, com foco em segurança, integração e facilidade de uso. O sistema foi desenvolvido utilizando Java 17, Spring Boot, PostgreSQL, Swagger e Docker, seguindo os princípios da Clean Architecture.

2. Arquitetura do Sistema

A arquitetura do Hairstyle é baseada nos princípios da Clean Architecture. Ela é composta por quatro camadas principais:

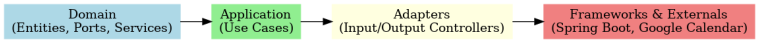
1. Domínio: Contém as entidades e regras de negócio principais.

2. Aplicação: Implementa os casos de uso e lógica de aplicação.

3. Adaptadores: Integram a aplicação com APIs externas, repositórios e controladores.

4. Frameworks e Externos: Inclui as bibliotecas e ferramentas externas, como Spring Boot e integração com Google Calendar.

A seguir, está o diagrama representando a Clean Architecture do sistema.



3. Tecnologias Utilizadas

- Java 17: Linguagem principal utilizada para desenvolvimento.

- Spring Boot: Framework para simplificar a criação de aplicações Java.

- PostgreSQL: Banco de dados relacional.

- Swagger: Ferramenta para documentação interativa da API.

- Docker: Para containerização e deploy em múltiplos ambientes.

- Maven: Gerenciamento de dependências e execução de testes.

- Google Calendar API: Integração para sincronização de agendamentos.

4. Execução do Sistema

O sistema pode ser executado tanto localmente quanto em um ambiente de nuvem. Seguem os passos detalhados:

Execução Local:

1. Certifique-se de ter Java 17 e Docker instalados.
2. Clone o repositório do GitHub:

   git clone https://github.com/MatheusFDS/hairstyle.git

   cd hairstyle
3. Construa e inicie o sistema usando Docker:

   docker-compose up --build
4. Acesse a API e o Swagger na URL: http://localhost:8080/swagger-ui/index.html

Execução na Nuvem:

1. Acesse o sistema hospedado em: http://34.230.19.130:8080
2. Use Postman ou cURL para interagir com a API, pois o CORS pode bloquear chamadas via navegador.

5. Funcionalidades Implementadas

- Cadastro de Estabelecimentos: Nome, endereço, serviços e horários.
- Cadastro de Profissionais: Especialidades, tarifas e horários disponíveis.
- Agendamento de Serviços: Visualização de disponibilidade e notificações.
- Avaliações e Comentários: Feedback para estabelecimentos e profissionais.
- Busca e Filtragem Avançada: Por nome, localização, serviços e avaliações.
- Integração com Google Calendar: Sincronização de agendamentos.

6. Testes

O sistema inclui testes unitários e de integração, cobrindo os seguintes cenários principais:

- Autenticação e geração de tokens JWT.
- Validação de agendamentos (conflitos e horários).
- Integração com o banco de dados e Google Calendar.

Para rodar os testes:

1. Execute o comando: mvn test
2. Verifique os relatórios de cobertura gerados no diretório target.

7. Desafios Técnicos

- Integração com Google Calendar: Configuração de credenciais OAuth.
- CORS: Configuração de headers para permitir acesso seguro.
- Dockerização: Configuração de serviços para aplicação e banco de dados.
- Clean Architecture: Garantir separação clara entre camadas.

8. Próximos Passos

- Implementar métricas de desempenho e monitoramento.
- Criar frontend integrado com a API.
- Adicionar notificações por SMS e e-mail.
- Suporte multilíngue para maior acessibilidade.