Otavio Augusto Thomas Trevisan

DOCUMENTO DE ARQUITETURA DO TESTE TÉCNICO - GERENCIADOR DE TICKETS

CASCAVEL
NOVEMBRO / 2025

Otavio Augusto Thomas Trevisan

DOCUMENTO DE ARQUITETURA DO TESTE TÉCNICO - GERENCIADOR DE TICKETS

Este documento é parte integrante da avaliação do teste técnico que estou fazendo, apresentando a arquitetura e o porquê das decisões técnicas feitas em relação ao projeto realizado.

Cascavel, 2025

SUMÁRIO

INTRODUÇÃO

O presente documento técnico tem como objetivo descrever o desenvolvimento do sistema Gerenciador de Tickets, criado como solução para o desafio proposto na prova prática de programação. O projeto contempla desde o planejamento da estrutura interna até a implementação completa das funcionalidades, seguindo boas práticas de organização de código, arquitetura e documentação.

O sistema foi desenvolvido em Java 17, utilizando Maven para gerenciamento de dependências e SQLite como banco de dados embarcado, garantindo portabilidade, simplicidade de execução e facilidade de manutenção. A interface gráfica foi construída com Java Swing, proporcionando uma experiência intuitiva para o usuário final.

Este documento apresenta a estrutura do projeto, decisões técnicas, diagramas, explicações sobre o funcionamento interno e instruções para execução. O objetivo é demonstrar não apenas o funcionamento correto do sistema, mas também as escolhas arquiteturais adotadas ao longo do desenvolvimento.

DESENVOLVIMENTO

ARQUITETURA DO PROJETO:

O projeto foi organizado seguindo a estrutura padrão de aplicações Java com Maven, que separa o código-fonte principal dos arquivos de teste, garantindo melhor organização, manutenção e escalabilidade.

Dentro da pasta principal do código (src/main/java), o projeto foi dividido em cinco pacotes, cada um responsável por uma parte específica da aplicação:

config, contém as classes responsáveis pela configuração do banco de dados e inicialização da conexão via JDBC. dao, reúne as classes de acesso a dados, responsáveis por realizar operações de CRUD diretamente no banco SQLite. model, abriga as classes que representam as entidades do sistema, como Funcionario e TicketEntrega. service, concentra as regras de negócio e validação da aplicação, atuando como intermediária entre a interface e a camada DAO. view, contém todas as telas construídas com Swing, incluindo formulários, listagens e relatórios.

Essa separação por responsabilidades segue boas práticas de desenvolvimento, facilita a manutenção do sistema e permite evolução futura com menor acoplamento entre as partes.

Um exemplo da utilidade dessa estrutura e o porquê de usar o acoplamento é que, se a qualquer parte do processo de desenvolvimento fosse necessário modificar alguma regra de negócio, bastava modificar dentro de service, caso o banco de dados fosse mudar, poderíamos saber facilmente onde modificar as configurações de inicialização, é interessante que o princípio de SOLID foi possível ser aplicado aqui, com o S em ênfase, com cada módulo de funcionamento com uma funcionalidade clara definida.

TECNOLOGIAS DO PROJETO:

As principais tecnologias utilizadas no desenvolvimento do projeto foram: Java 17, linguagem de programação moderna, robusta e orientada a objetos. A escolha dessa versão se deu por sua estabilidade, recursos atualizados e pela proficiência prévia no ecossistema Java, Maven, ferramenta de gerenciamento e automação que organiza o projeto, controla dependências e simplifica o processo de build, garantindo padronização na estrutura do código, Java Swing, biblioteca utilizada para construção da interface gráfica, permitindo o desenvolvimento de janelas, formulários e componentes visuais de maneira

simples e integrada ao ambiente Java, JUnit 5, framework de testes unitários utilizado para validar funcionalidades críticas da aplicação e garantir maior confiabilidade no código, SQLite, banco de dados leve e integrado ao projeto através de JDBC, escolhido por ser prático, portátil e adequado ao escopo de um sistema de pequeno porte.

SQLite E PORQUE DE SUA ESCOLHA:

O banco de dados utilizado no projeto é o SQLite, um sistema de gerenciamento de banco de dados relacional leve. A escolha desse banco de dados se deu por quatro motivos principais, o banco é armazenado em um único arquivo (database.db), permitindo que o projeto funcione em qualquer máquina sem necessidade de instalar um servidor de banco de dados, baixa complexidade, por ser um banco local, elimina a necessidade de configurações complexas, usuários, permissões e serviços externos, adequado para pequenos sistemas, para um projeto de avaliação ou aplicações desktop com poucos usuários simultâneos, o SQLite atende perfeitamente.

DESAFIOS ENCONTRADOS:

Por ser a primeira vez utilizando o Maven, foi necessário compreender sua estrutura de organização de projetos, sua filosofia e o motivo por trás dos diretórios padrão, como src/main/java e src/test/java, além disso, entender como funcionam o pom.xml, as dependências, os plugins e o ciclo de vida de build demandou tempo e dedicação, apesar do desafio inicial, o aprendizado proporcionou uma visão mais clara sobre boas práticas e padronização de projetos Java modernos.

Outro ponto desafiador foi a implementação do algoritmo de validação de CPF, por nunca ter trabalhado com essa regra de negócio de forma completa, foi necessário estudar como funcionam os dígitos verificadores, como aplicar o cálculo e como garantir que as validações fossem consistentes e seguras, a inserção desse algoritmo na camada de serviço, juntamente com o tratamento de erros e casos extremos, proporcionou uma experiência prática muito relevante.

A adoção de uma arquitetura dividida em várias camadas, como model, service, dao, view e config, exigiu um entendimento mais profundo sobre responsabilidades, isolamento de lógica e boas práticas de organização, encontrar a forma ideal de estruturar o projeto, separar corretamente as responsabilidades e montar um fluxo limpo entre as camadas foi desafiador inicialmente, mas extremamente recompensador, esse processo ajudou a consolidar o entendimento sobre padrões de projeto e arquiteturas escaláveis.

CONCLUSÃO

O desenvolvimento deste projeto proporcionou uma oportunidade valiosa para aplicar conceitos fundamentais de programação orientada a objetos, arquitetura em camadas e boas práticas de desenvolvimento Java, ao longo do processo, foi possível construir uma aplicação completa, com interface gráfica, banco de dados, validações e testes automatizados, integrando diferentes tecnologias.

De forma geral, o projeto cumpriu todos os requisitos propostos e serviu como uma excelente experiência prática, consolidando conhecimentos e ampliando a capacidade de estruturar, implementar e testar aplicações profissionais em Java.