**PADRÕES ARQUITETURAIS SGITP (SISTEMA DE GESTÃO INTELIGENTE DE TRANSPORTE PÚBLICO)**

**GUILHERME FERREIRA RIBEIRO**

1. **Introdução**
   1. **Objetivo**

Este documento tem como objetivo estabelecer os padrões arquiteturais que guiarão o desenvolvimento do Sistema de Gestão Inteligente de Transporte Público. Os padrões aqui definidos buscam garantir a qualidade, escalabilidade, segurança e facilidade de manutenção do sistema, proporcionando uma base sólida para o desenvolvimento contínuo e para futuras evoluções.

* 1. **Escopo**

Este documento abrange todos os componentes do sistema, incluindo API Application, Web Application, Mobile Application, e os serviços de suporte. A aplicação desses padrões é mandatória para todas as fases do desenvolvimento e será monitorada continuamente para garantir a conformidade.

1. **Princípios Arquiteturais**
   1. **Modularidade**

O sistema será desenvolvido com uma abordagem modular, onde os diferentes módulos do sistema serão independentes, facilitando a substituição, atualização e manutenção de cada módulo sem impactar os outros.

* 1. **Desacoplamento**

Os componentes do sistema serão projetados para minimizar o acoplamento, com comunicação através de interfaces bem definidas. Isso permitirá que os módulos interajam entre si de maneira flexível, sem dependências rígidas.

* 1. **Reutilização**

Será incentivada a reutilização de componentes e de código em todo o sistema, tanto para otimizar o tempo de desenvolvimento quanto para garantir a consistência das funcionalidades implementadas.

* 1. **Escalabilidade**

O sistema está sendo desenhado para suportar a escalabilidade horizontal e vertical, permitindo que a infraestrutura cresça conforme a demanda sem necessidade de reestruturações significativas.

* 1. **Segurança**

Todas as operações do sistema, especialmente aquelas envolvendo dados sensíveis, serão protegidas por padrões de segurança rigorosos. A comunicação entre os componentes será criptografada, e serão implementadas práticas de segurança que seguem as diretrizes da OWASP.

* 1. **Manutenibilidade**

O sistema será construído de forma a facilitar sua manutenção e evolução, com boas práticas de codificação, documentação abrangente e utilização de padrões de design que garantam flexibilidade.

1. **Padrões Arquiteturais**
   1. **Arquitetura em Microservices**

O sistema será dividido em micro serviços, cada um responsável por uma funcionalidade específica. Esses serviços serão independentes e poderão ser escalados, mantidos e implantados de forma isolada.

A arquitetura em micro serviços facilita a escalabilidade, a manutenção e a implantação contínua (CI/CD), permitindo uma maior flexibilidade no desenvolvimento e na operação do sistema.

* 1. **API Gateway**

Um API Gateway será utilizado como ponto único de entrada para todas as requisições externas ao sistema. Ele gerenciará o roteamento das requisições para os micro serviços correspondentes.

O API Gateway fornece uma camada adicional de segurança e controle, permitindo centralizar a autenticação, a autorização, e o monitoramento do tráfego. Ele simplifica a comunicação entre os clientes (front-end) e os micro serviços.

* 1. **CQRS (Command Query Responsibility Segregation)**

O padrão CQRS será implementado para separar as operações de leitura (queries) das operações de escrita (commands), possibilitando otimizações específicas para cada tipo de operação.

A segregação das responsabilidades melhora a performance do sistema, facilita a implementação de auditorias e logs, e permite a escalabilidade diferenciada de operações de leitura e escrita.

* 1. **Event-Drive Architecture**

O sistema adotará uma arquitetura orientada a eventos, onde as comunicações assíncronas entre componentes serão baseadas em eventos.

A arquitetura orientada a eventos aumenta a resiliência e a escalabilidade do sistema, permitindo que diferentes partes do sistema respondam a eventos sem depender diretamente umas das outras.

1. **Padrões de Design**
   1. **Singleton**

O padrão Singleton será utilizado para garantir que uma classe tenha apenas uma instância e forneça um ponto global de acesso a essa instância.

Esse padrão será utilizado em componentes que precisam de um estado compartilhado globalmente, como gerenciadores de configuração ou provedores de autenticação.

* 1. **Factory Method**

O padrão Factory Method será empregado para criar objetos sem especificar a classe exata a ser instanciada, promovendo flexibilidade e reutilização de código.

Será utilizado principalmente na criação de instâncias de serviços que podem variar conforme o ambiente (ex.: produção, desenvolvimento)

* 1. **Repository**

O padrão Repository abstrai a lógica de acesso a dados, permitindo que a camada de negócios interaja com uma interface simplificada, enquanto a lógica de persistência fica encapsulada.

Esse padrão será implementado em todos os módulos que interagem diretamente com o banco de dados, facilitando a troca do backend de persistência, se necessário.

* 1. **Adapter**

O padrão Adapter será utilizado para compatibilizar a interface de um componente com a interface esperada por outro componente, facilitando a integração de sistemas legados ou de bibliotecas externas.

Será usado especialmente em integrações com sistemas externos ou com componentes que não seguem o mesmo padrão de interface.

1. **Tecnologias e Ferramentas**
   1. **CI/CD**

A integração e entrega contínuas serão gerenciadas usando Gitlab CI, que automizará a compilação, os testes e a implantação do sistema.

* 1. **Versionamento de Código**

O Gitlab será utilizado para controle de versão, com práticas recomendadas para branching e naming conventions que facilitem o gerenciamento de releases e o trabalho colaborativo.

* 1. **Monitoramento e Logging**

O monitoramento do sistema será realizado através da ELK Stack que permite a coleta e visualização de métricas em tempo real, bem como o armazenamento e análise de logs centralizados.

* 1. **Segurança**

Serão utilizadas práticas de segurança avançadas para proteger o sistema contra ameaças comuns. Oauth 2.0 será utilizado para autenticação e autorização, e SSL/TLS para criptografia de dados em trânsito.

1. **Governança e Controle de Qualidade**
   1. **Revisão de Código**

Todas as alterações no código passarão por revisões obrigatórias por outros membros da equipe para garantir a qualidade e a aderência aos padrões estabelecidos.

* 1. **Testes Automatizados**

Será estabelecida uma cobertura mínima de testes automatizados, incluindo testes unitários, de integração e end-to-end.

* 1. **Documentação**

Todos os componentes do sistema devem ser bem documentados, assim como já está sendo feito, com diagramas sempre atualizados e documentação de APIs disponíveis para consulta por toda a equipe.

1. **Conclusão**

A adoção desses padrões arquiteturais proporcionará uma base sólida para o desenvolvimento do Sistema de Gestão Inteligente de Transporte Público, garantindo que o sistema seja escalável, seguro, e de fácil manutenção. A conformidade com esses padrões é essencial para o sucesso do projeto e será monitorado de perto durante todo o ciclo de vida do desenvolvimento.