Trabalho Final: Desenvolvimento de um Sistema com Base em Arquitetura de Software

1. Introdução

- **Objetivo**: Desenvolver uma proposta arquitetural completa para um sistema funcional, utilizando uma arquitetura de software adequada que atenda aos requisitos especificados.
- **Descrição Geral**: O sistema deve simular um cenário real, onde o aluno terá que escolher e justificar a adoção de um padrão arquitetural, além de elaborar uma documentação detalhada da arquitetura proposta.

2. Cenário de Desenvolvimento

- O desenvolvimento da proposta arquitetural deve considerar o seguinte cenário:
    - **Contexto do Negócio**: A startup atua no setor de **e-commerce especializado em produtos sustentáveis**. O objetivo é oferecer uma plataforma online que facilite a venda de produtos ecologicamente corretos, como itens recicláveis, orgânicos e de impacto ambiental reduzido. A empresa está crescendo rapidamente e precisa de uma solução que seja escalável para atender a um aumento considerável de usuários.
    - **Equipe** A empresa valoriza a agilidade e flexibilidade na entrega contínua de novas funcionalidades. Além disso, a equipe deve considerar a possibilidade de futuros desenvolvedores integrarem o time, o que demanda uma arquitetura bem documentada e fácil de escalar.
    - **Infraestrutura**: O sistema será hospedado em uma infraestrutura baseada em **cloud computing**, utilizando serviços de containers (como Docker e Kubernetes) e bancos de dados em nuvem. A escolha da arquitetura precisa considerar a utilização eficiente de recursos e a integração com serviços de terceiros.
    - **Requisitos de Segurança**: Como a plataforma lida com dados pessoais e transações financeiras, a arquitetura precisa implementar boas práticas de segurança, como criptografia de dados, controle de acesso adequado e conformidade com regulações como a **LGPD** (Lei Geral de Proteção de Dados) ou **GDPR**.

3. Requisitos Funcionais e Não Funcionais

- Definir requisitos mínimos do sistema, como:
    - Funcionalidade principal de gerenciamento de catálogo de produtos sustentáveis.
    - Funcionalidade de carrinho de compras e sistema de checkout seguro.
    - Suporte para multiusuários simultâneos, com perfis de cliente e vendedor.
    - Sistema de recomendação de produtos com base em comportamentos de compra e preferências sustentáveis.

- Escalabilidade horizontal, para que a aplicação possa crescer de forma flexível.
- Implementação de boas práticas de segurança (como autenticação e autorização).
- Utilização de cache para melhorar a performance.

4. Etapas do Trabalho

1. **Análise de Requisitos**
   - Realizar uma análise detalhada dos requisitos funcionais e não funcionais.
   - Desenhar os casos de uso principais do sistema, considerando o cenário de desenvolvimento e as funcionalidades de e-commerce.
2. **Escolha da Arquitetura**
   - Com base no cenário apresentado, os alunos devem escolher uma arquitetura que atenda aos seguintes aspectos:
     - Suporte à escalabilidade e modularidade.
     - Integração fácil com serviços em nuvem.
     - Facilidade de manutenção e expansão, com uma base de código clara para novos desenvolvedores.
     - Implementação de boas práticas de segurança, especialmente no que diz respeito a dados pessoais e transações financeiras.
     - Exemplos de arquiteturas a serem consideradas:
       - Arquitetura de Microservices.
       - Arquitetura em Camadas.
       - Arquitetura Event-driven.
       - Arquitetura Hexagonal.
   - Justificar a escolha com base nos requisitos do projeto e no cenário de desenvolvimento.
3. **Desenho da Arquitetura**
   - Criar diagramas UML que representem a arquitetura:
     - Diagrama de componentes.
     - Diagrama de sequência.
     - Diagrama de classes.
4. **Detalhamento de Classes e Funções**
   - Utilizando **pseudocódigo**, detalhar as principais **classes e funções** do sistema, dentro do contexto da arquitetura escolhida.
   - **Pseudocódigo** é uma forma de descrever algoritmos de maneira simples e estruturada, sem a necessidade de seguir a sintaxe rigorosa de uma linguagem de programação específica. Ele permite que o desenvolvedor foque na lógica do algoritmo.

Exemplos de pseudocódigo:
plaintext
Copiar código

```
Classe Produto:
    Atributos:
        nome
        preco
```

```
        quantidade

    Métodos:
        função adicionarEstoque(quantidade):
            this.quantidade = this.quantidade + quantidade

        função calcularValorTotal():
            retornar this.preco * this.quantidade

Função principal:
    produto1 = Produto("Caneca Sustentável", 20.00, 10)
    produto1.adicionarEstoque(5)
    total = produto1.calcularValorTotal()
```

-
- O pseudocódigo deve descrever as principais responsabilidades das classes, incluindo interações entre componentes, e as funções críticas, com suas entradas, saídas e interações com outros módulos.

5. **Estratégia de Escalabilidade**
   - Definir e detalhar uma **estratégia de escalabilidade** que garanta o crescimento contínuo da plataforma, considerando tanto a escalabilidade horizontal (adicionar mais servidores) quanto a escalabilidade vertical (melhorar o desempenho de servidores existentes). Levar em conta os seguintes aspectos:
     - **Escalabilidade de infraestrutura**: como o sistema será escalado em termos de servidores, containers e outros recursos de cloud computing.
     - **Escalabilidade de banco de dados**: soluções para particionamento de dados (sharding), replicação e otimização de consultas.
     - **Escalabilidade de serviços**: como novos serviços podem ser adicionados ou serviços existentes podem ser replicados para lidar com a carga crescente de usuários e transações.
6. **Documentação**
   - Elaborar a documentação completa da arquitetura, considerando o cenário específico e os requisitos de infraestrutura, segurança e escalabilidade. A documentação deve incluir:
     - Visão geral do sistema (contexto e visão geral).
     - Decisões arquiteturais e padrões aplicados, com justificativa.
     - Diagramas UML (componentes, sequência e classes).
     - Pseudocódigo detalhando as classes e funções principais.
     - Descrição das interfaces e APIs (se aplicável).
     - Considerações sobre escalabilidade e segurança, em alinhamento com o cenário de desenvolvimento.

5. Entrega

- **Relatório Final**: O relatório deve ser entregue em formato de **documentação de arquitetura de software** e conter:
    - Introdução e objetivos do sistema.
    - Descrição detalhada da escolha da arquitetura, justificando-a com base no cenário.
    - Diagramas UML.
    - Explicação das decisões de design e seus impactos no projeto, considerando a infraestrutura de cloud computing e a necessidade de segurança e escalabilidade.
    - Detalhamento da **estratégia de escalabilidade**, especificando como a plataforma será escalada para lidar com o crescimento da base de usuários.
    - Pseudocódigo descrevendo as classes e funções principais do sistema dentro da arquitetura proposta.

6. Critérios de Avaliação

- **Adequação aos Requisitos e Cenário**: A proposta arquitetural atende aos requisitos funcionais e não funcionais estabelecidos e leva em consideração o cenário de desenvolvimento?
- **Qualidade da Arquitetura**: A arquitetura escolhida é adequada ao problema e ao cenário descrito? Está bem documentada e justificada?
- **Documentação de Arquitetura**: A documentação está clara, completa e segue o padrão de arquitetura de software, levando em consideração o cenário proposto?
- **Estratégia de Escalabilidade**: A estratégia de escalabilidade está bem definida e endereça adequadamente as necessidades de crescimento do sistema?
- **Pseudocódigo de Classes e Funções**: O pseudocódigo está bem estruturado, reflete corretamente a arquitetura proposta e descreve de forma clara as principais classes e funções do sistema?