1. (1,0 ponto) Por que nem todas as decisões de arquitetura de software podem ser tomadas
logo no início do projeto?

R: Porque, em grande medida, no início do projeto não temos todas as informações sobre os requisitos do sistema. Além disso, no decorrer do projeto podem surgir novos requisitos e modificações. Portanto, é importante que a arquitetura do software seja flexível para acomodar mudanças futuras e evoluir ao longo do tempo.

2. (1,0 ponto) Se você fosse um gerente de projetos e precisasse convencer um diretor não
técnico que é necessário contratar um arquiteto de software para um novo projeto (grande e
complexo), quais argumentos você usaria para convencê-lo? Considere que o diretor não tem
ideia do que um arquiteto de software faz.

R: Eu afirmaria que um profissional de arquitetura de software seria extremamente útil para a redução de custos futuros, tendo em vista que esse profissional é responsável por elaborar uma arquitetura robusta para o projeto, especialmente adaptado ao crescimento futuro do sistema. Além disso, um arquiteto de software pode elaborar e definir boas práticas e padrões de desenvolvimento que ajudam a garantir a eficiência do sistema

3. (1,0 ponto) Qual a diferença entre projeto de software e projeto de arquitetura?
R: O projeto de software é focado na criação do sistema em si, enquanto o projeto de arquitetura de software é focado na definição da estrutura e organização do software, visando garantir sua longevidade.

4. (1,0 ponto) Por que as etapas de requisitos e projeto de arquitetura apresentam uma
intersecção significativa em seus processos?

R: Porque a etapa de requisitos influencia diretamente na arquitetura. Tendo em vista que é necessário entender claramente os requisitos funcionais e não funcionais do sistema e, assim, criar uma arquitetura compatível com esses requisitos. Logo, a intersecção entre as etapas de requisitos e projeto de arquitetura é importante para garantir que o software seja bem projetado e atenda aos requisitos do cliente.

5. (1,0 ponto) Ao elaborar o projeto de arquitetura, alguns requisitos não funcionais podem ser
conflitantes. Cite um exemplo, diferente do apresentado em aula, em que dois ou mais
requisitos não funcionais são incompatíveis. Qual seria uma possível saída para esses casos?

Dê outro exemplo usando outros requisitos não funcionais.

R : Por exemplo, podemos ter dois requisitos não funcionais altamente críticos e conflitantes em uma arquitetura X e Y: desempenho e escalabilidade. Uma arquitetura oferece bastante desempenho, mas peca na escalabilidade de volume de dados e vice-versa. A meu ver, é necessário identificar um meio termo entre esses aspectos dos requisitos não funcionais dentro da arquitetura de software, pontualmente nas áreas que demandam mais escalabilidade e desempenho.

6. (1,0 ponto) Por que o projeto de arquitetura deve contemplar várias visões? Descreva com
suas palavras as visões apresentadas em aula.
R: Deve contemplar diversas visões pois, a partir delas, teremos os

subsídios necessários para identificar os aspectos fundamentais do software, incluindo o desenvolvimento satisfatório dos requisitos funcionais e não funcionais. Além disso, cada visão pode ter suas particularidades e preocupações, como questões de desempenho, escalabilidade, segurança e etc. Além disso, diversas visões auxiliam na melhor compreensão do projeto por parte dos desenvolvedores e proprietários de software.

7. (4,0 pontos) Faça uma pesquisa sobre documentos de arquitetura de software e, usando como
base a estrutura apresentada em aula, elabore um documento de arquitetura considerando
que o sistema a ser desenvolvido é um site para cotação de pacotes de viagem. Obs.: Nas
seções que descrevem as visões de arquitetura não é preciso desenhar os respectivos
diagramas UML.

Documento de Arquitetura de Software

Introdução

Este documento descreve a arquitetura do sistema de cotação de pacotes de viagem. O objetivo deste sistema é fornecer aos usuários a capacidade de pesquisar e comparar pacotes de viagem de diferentes agências de turismo.

Visão Geral

O sistema é composto por dois principais componentes: o Front-end, que é a interface do usuário e o Back-end, que é responsável pelo processamento de solicitações e gerenciamento dos dados. O Front-end é desenvolvido usando tecnologias web como HTML, CSS, JavaScript e o Back-end é implementado em Java utilizando o framework Spring.

Arquitetura do Front-end

O Front-end é baseado em uma arquitetura cliente-servidor. A interface do usuário é construída usando HTML, CSS e JavaScript e é executada no lado do cliente. O JavaScript é responsável por fazer chamadas assíncronas ao servidor para obter e enviar dados. A arquitetura do Front-end é baseada no padrão Model-View-Controller (MVC). O modelo representa a estrutura de dados, a visualização é responsável por exibir os dados e o controlador é responsável por gerenciar as interações do usuário.

Arquitetura do Back-end

O Back-end é baseado em uma arquitetura de serviços. O sistema utiliza o framework Spring para criar e gerenciar os serviços. O Spring MVC é utilizado para expor os serviços como endpoints RESTful. Os serviços são responsáveis por receber e processar as solicitações do cliente, executar as regras de negócio e retornar os resultados para o cliente. O sistema utiliza um banco de dados relacional para armazenar os dados de usuário e os pacotes de viagem.

Arquitetura do Banco de Dados

O sistema utiliza um banco de dados relacional para armazenar os dados. O banco de dados é composto por duas tabelas principais: usuários e pacotes de viagem. A tabela de usuários armazena informações do usuário como nome, endereço de e-mail e senha. A tabela de pacotes de viagem armazena informações sobre os pacotes de viagem, como destino, preço e agência de turismo.

Considerações de Segurança

O sistema implementa medidas de segurança para proteger os dados do usuário e do sistema. Todas as informações de usuário são armazenadas em formato criptografado no banco de dados. O sistema utiliza autenticação e autorização baseadas em token para garantir que somente usuários autenticados tenham acesso às informações protegidas.

Considerações de Escalabilidade

O sistema foi projetado para ser escalável. O Front-end é implementado em uma arquitetura de servidor único, mas pode ser facilmente distribuído em vários servidores para lidar com um grande número de usuários. O Back-end utiliza um modelo de arquitetura de serviços que permite a adição de mais servidores para lidar com o aumento da demanda. O banco de dados pode ser replicado para suportar a leitura de carga balanceada.

Conclusão

Este documento descreve a arquitetura do sistema de cotação de pacotes de viagem. A arquitetura foi projetada para ser escalável, segura e fácil de manter. O sistema utiliza uma arquitetura cliente-servidor para o Front-end e Back-end.