Projeto de Base de Dados - Trabalho Prático

Base de Dados II - Sistema de Gestão de Reservas

Escola Superior de Tecnologia e Gestão
Politécnico de Coimbra

Introdução:

O objetivo deste trabalho é desenvolver um sistema completo de reservas para um hotel, utilizando PostgreSQL (PL/pgSQL) para a lógica de base de dados e uma API em Flask para interagir com o sistema. Além disso, serão abordadas práticas de segurança, auditoria, backup e deployment da aplicação.

Este projeto visa proporcionar aos alunos experiência no desenvolvimento de sistemas de base de dados. O projeto é orientado às necessidades do mercado de trabalho; assim, os alunos experimentam todas as principais etapas de um projeto comum de desenvolvimento de software, desde o início até a entrega.

Objetivos:

Depois de concluir este projeto, os alunos devem ser capazes de:

- Compreender como um projeto de desenvolvimento de aplicativo de base de dador é organizado, projetado e executado
- Executar a criação de modelos de dados conceituais e físicos para suportar e persistir dados de aplicações
- Projetar, implementar, testar um sistema de base de dados
- Instalar, configurar e gerir um DBMS relacional moderno
- Compreender a programação do lado do cliente e do servidor em SQL e PL/pgSQL.

Grupo:

- O projeto deve ser realizado por grupos de 2 alunos.
- Os grupos devem fazer a sua inscrição até ao dia 1/3 no ficheiro disponível na descrição da submissão de trabalhos.

Requisitos:

O projeto final deve corresponder os seguintes requisitos:

- Usar um DBMS transacional PostgreSQL
- Uma arquitetura de base de dados distribuída com uma API REST
- SQL e PL/pgSQL
- Triggers, funções e procedimentos adequados e relevantes executados no lado do DBMS
- Estratégias para gestão transações, concorrência e segurança
- Estratégias de prevenção, deteção e mitigação de erros
- Implementação de todas as funcionalidades abordadas durante as aulas com documentação detalhada.
- Documentação detalhada

Gestão de Reservas– Descrição Funcional

O sistema de gestão de reservas permitirá que clientes realizem reservas em quartos de hotel de forma segura e eficiente. A aplicação fornecerá uma interface para que administradores, rececionistas e clientes possam interagir com os dados, respeitando diferentes níveis de permissões. Os administradores terão o controlo total sobre o sistema, enquanto os rececionistas poderão gerir reservas e pagamentos. Os clientes poderão criar e cancelar reservas, além de visualizar seu histórico de estadias.

A API permitirá operações como registo e autenticação de clientes, criação e consulta de reservas, processamento de pagamentos e upload de imagens de quartos. Para garantir a integridade e desempenho da base de dados, serão utilizadas técnicas avançadas como procedures, triggers, cursores, particionamento e indexação. Além disso, o sistema implementará auditoria de operações e mecanismos de backup e recuperação, assegurando a segurança dos dados.

A API será alojada na plataforma Vercel, garantindo acessibilidade e disponibilidade contínua. A base de dados é alojada no servidor *aid.estgoh.ipc.pt*. O sistema deve ser disponibilizado através de uma API REST que permita ao utilizador aceder ao sistema através de pedidos HTTP (quando conteúdo for necessário, deve ser usado JSON).

O objetivo da UC está na criação de uma base de dados e das funcionalidades associadas, o desenvolvimento de aplicações frontend não está no foco deste trabalho. Para usar ou testar a sua API REST, deve ser usado o software POSTMAN.

Importante: *A lógica do sistema deve ser implementada totalmente em SQL e não no web server!*

Funcionalidades a desenvolver:

- Registo de utilizadores (clientes, recepcionistas, administradores)
- Login com autenticação segura (JWT)
- Armazenamento seguro de senhas (bcrypt)
- Controlo de permissões e privilégios na base de dados
- Registo, atualização e remoção de quartos
- Upload e armazenamento de imagens dos quartos (BYTEA)
- Consulta de disponibilidade de quartos
- Criar reservas verificando disponibilidade
- Cancelamento de reservas com regras de penalidade
- Consulta de reservas ativas de um cliente
- Listagem de todas as reservas para administradores
- Registo e processamento de pagamentos
- Atualização automática do status da reserva após pagamento
- Consulta de histórico de pagamentos
- Registo de ações no sistema (auditoria)
- Backup e recuperação da base de dados
- Implementação de índices para acelerar consultas
- Uso de particionamento na tabela de reservas
- Uso de cursores para aumentar o preço das reservas de acordo com critérios dinâmicos (por exemplo até atingir um determinado valor)
- Desenvolvimento de endpoints (explorar arquitetura em *microservices)*
- Proteção dos endpoints com autenticação JWT

- Upload e recuperação de imagens dos quartos
- Implementação da API Flask na Vercel
- Documentação detalhada da API e instruções de instalação

Para a entrega do trabalho é necessário entregar o seguinte:

1. Definição das principais operações da base de dados, transações e problemas de concorrência
2. Plano de trabalho: tarefas, divisão inicial do trabalho por aluno e cronograma
3. Diagrama ER
4. Dicionário de dados.
5. Modelo de dados relacionais (tabelas) – script de criação das tabelas
6. Criação das views, rules, cursores, excepções, procedimentos, funções e triggers necessários para o funcionamento da aplicação.
7. Tarefa em aula: 18/3.
8. Tarefa em aula: 18/3.
9. Tarefa em aula: 18/3.
10. Criar índices em colunas chave para otimizar as buscas mais frequentes, como: Índice no email dos clientes para login rápido, índice nas datas de reserva para acelerar consultas de disponibilidade e índice no status das reservas para otimizar filtros de pesquisa. (Demonstrar com *EXPLAIN*)
11. Implementar particionamento na tabela de reservas com base no ano da reserva, melhorando a performance de consultas em grandes volumes de dados.
12. Criar um mecanismo para armazenar imagens dos quartos na base de dados usando a coluna BYTEA e fornecer um endpoint na API para upload e recuperação de imagens.
13. Criação da API REST com recurso ao Flask de forma a criar os serviços que permitem a interação com a base de dados através de pedidos HTTP.
    a. (POST) /auth/register – Registar um novo cliente.
    b. (POST) /auth/login – Autenticação de cliente com e-mail e senha.
    c. (POST) /reservas – Criar uma nova reserva (chamando o procedure PL/pgSQL).
    d. (GET) /reservas/{id} – Consultar detalhes de uma reserva.
    e. (POST) /pagamentos – Processar pagamento de uma reserva.
    f. (PUT) /reservas/{id}/cancelar – Cancelar uma reserva, aplicando regras de negócio.
    g. (POST) /upload-imagem – Upload de imagens dos quartos (salvando no PostgreSQL em BYTEA).
    h. (GET) /quartos/{id}/imagem – Recuperar a imagem de um quarto.
14. Implementar tokens JWT para autenticação segura.
15. Endpoints protegidos exigem autorização via token JWT.
16. Criação de dois utilizadores com privilégios diferentes. Um com privilégios de consulta e outro com privilégios de consulta/alteração de dados
17. As palavras-passe devem ser guardadas com encriptação
18. Devem entregar um backup completo da base de dados.
19. Senhas dos clientes devem ser armazenadas com hashing seguro (bcrypt).
20. Criar um sistema de permissões no PostgreSQL para diferentes perfis de utilizadores:

a. Admin: Acesso total (CRUD em todas as tabelas, gestão de utilizadores).
b. Rececionista: Pode criar reservas, verificar pagamentos, consultar clientes.
c. Cliente: Apenas pode consultar e cancelar as próprias reservas.

Nota: Os alunos devem criar:

1. Três roles (`admin_role`, `rececionista_role`, `cliente_role`).
2. Utilizadores fictícios (`admin_user`, `rececionista_user`, `cliente_user`) e atribuir roles.
3. GRANT para definir permissões de acesso às tabelas.
4. View segura para que os clientes possam consultar apenas suas próprias reservas.

21. Criar uma tabela de auditoria para monitorizar operações críticas, registando:
    a. Utilizador responsável, timestamp e detalhes da ação.
    b. Triggers para capturar eventos como criação, alteração e cancelamento de reservas.

Considerações finais:

A documentação deve ser técnica e justificar as opções selecionadas.

O código deve ser comentado.

O relatório deve ser estruturado na seguinte forma:

- Introdução
- Métodos
- Resultados
- Discussão
- Conclusão
- Referências
- Anexo 1 – lista de tarefas implementadas com a indicação do elemento do grupo que as executou de acordo com a numeração 1-21
- Restantes Anexos

Além disso, o relatório deve incluir obrigatoriamente a seguinte informação:

- Manual de utilizador descrevendo como os utilizadores podem interagir com a aplicação
- Manual de instalação descrevendo como implementar e executar o software desenvolvido
- ER final e modelos de dados relacionais
- Todas as informações que considerar relevantes para entender como o software está construído.
- Código fonte e Scripts:
    - Inclua o código-fonte, scripts, ficheiros executáveis e bibliotecas necessárias para compilar e executar o software (identifique o DBMS usado, não faça upload dos ficheiros binários)
    - Scripts de criação da base de dados contendo as definições de tabelas, restrições, sequências, utilizadores, funções, permissões, triggers, funções e procedimentos.

Nota importante: Os projetos são realizados em grupo de dois elementos e posteriormente será realizada uma defesa oral individual. O projeto é realizado em grupo, mas a avaliação é feita individualmente em defesa técnica.

Os grupos são construídos por dois elementos. Para desenvolver o trabalho individualmente, o aluno terá que informar o professor do motivo e fica sujeito à aceitação do professor. A submissão deste trabalho sem defesa oral não será avaliada.