2ª Meta do Projeto SGD

< Cloud Query >

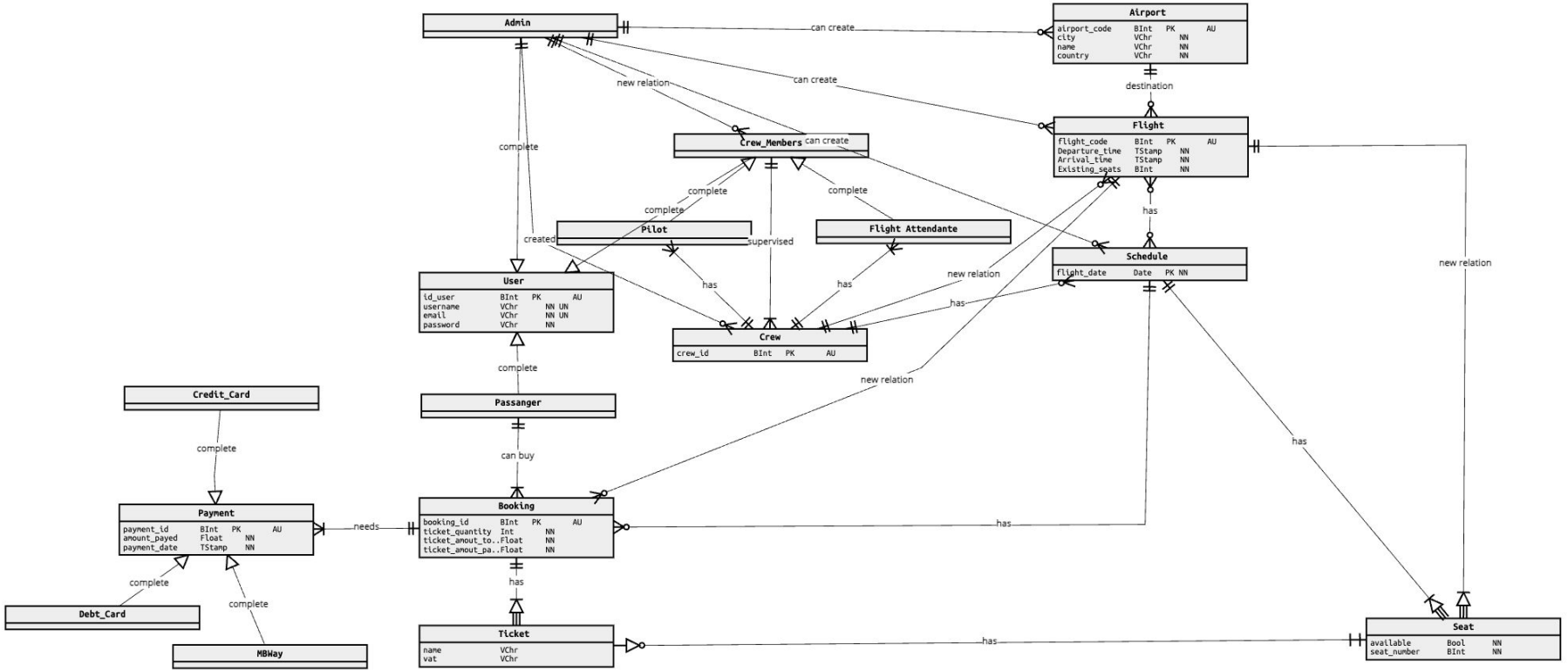
Equipa

- Francisca Mateus
  Nº estudante: 2023212096
  Email: mateusfrancisca2005@gmail.com
- Pedro Silva
  Nº estudante: 2023235452
  Email: pedrosilva222004@gmail.com
- Ramyad Raadi
  Nº estudante: 2023205631
  Email: ramyad.r.pt@gmail.com

Descrição do projeto

- Criação de endpoints REST e uma base de dados relacional para servir como backend para um sistema de uma companhia aérea de passageiros.
- A REST API será o esqueleto desta companhia aérea que vende vôos e fornecerá várias funcionalidades aos clientes.

Arquitetura ER conceptual Final



Descrição da extrutura do software

- Base de Dados
  - No projeto foi usado uma base de dados relacional que utiliza como SGBD o PostgreSQL.
  - Baseado no modelo relacional apresentado no slide anterior, foram criadas 18 tabelas.
- Postman
  - Ferramenta utilizada para testar os endpoints implementados na API RESTful.
- Docker
  - Ferramenta utilizada para garantir que a API e a Base de Dados são executados em um ambiente consistente, minimizando os problemas de compatibildade

Descrição da extrutura do software

- API RESTful
  - Tipos de user: Cada tipo de user tem funcionalidades restritas (explicitadas no manual de instruções)
    - Passageiros
    - Administradores
    - Tripulantes
  - Endpoints
    - No projeto foram implementados todos os endpoints obrigatórios do enunciado do projeto como também foram implementados outros endpoints que completam a forma como users interagem com a base de dados, por intermédio da API.
    - Foram utilizados os métodos POST, PUT e GET para realizar as operações na Base de Dados.
      - No manual de instruções podemos ver todos os endpoints implementados e o seu formato.
    - Para enviar informações nos pedidos foram utilizados ficheiros JSON e para fazer as autorizações foram utilizados Tokens.

Problemas de Transação

- Durante a elaboração do projeto, para garantir a inexistência de problemas de transação, tivemos de implementar métodos em cada endpoint para cumprir com os pressupostos das propriedades ACID.
    - Atomicidade: Caso exista uma falha a meio da transação, utilizamos o “Rollback” para reverter todas as operações efetuadas até esse ponto.
        - Exemplo: Em cada endpoint do tipo POST, foram criadas verificações para o input e para as operações na base de dados, que caso ocorra uma falha, vai ser efetuado um “rollback” e vai ser transmitido ao utilizador o erro.
    - Consistência: Durante a transação caso, exista dependências da entidade alterada com outras entidades, vai ser feita uma alteração nas entidades dependentes.
        - Exemplo: No momento em que é feito um pedido de reserva de assentos, para um certo voo, a entidade seats vai ser atualizada no atributo “available”, para cada lugar reservado.
    - Isolamento: Está relacionado com os problemas de concorrência.
    - Durabilidade: No momento em que acaba transação, termina com sucesso, ou seja, não existe nenhuma falha nas operações efetuadas, vai ser efetuado um “commit” no final da transação.
        - Exemplo: Em cada endpoint do tipo POST, caso a última operação seja bem sucedida, vai ser feito um commit na base de dados.

Problemas de Concorrência

No decorrer da realização deste projeto, deparámo-nos com alguns problemas de concorrência, sendo eles por exemplo, várias transações acederem aos mesmos dados em simultâneo.

Mais especificamente, podemos observar este problema quando vários passageiros tentam realizar uma reserva para o mesmo voo, que contém lugares iguais.

Para lidarmos com o problema, utilizamos o FOR UPDATE, que bloqueia os registos selecionados às outras transações, não deixando modificar ou eliminar os registos. Este método garante-nos que caso algum passageiro queira reservar o mesmo lugar, não vai ter sucesso na sua transação, sendo obrigado a escolher outro lugar.

Tendo em consideração, que o lugar só será do passageiro que fez a reserva, no momento em que completa o pagamento, o sistema só vai dar acesso aos bilhetes, no momento em que o pagamento seja concluído. Para futuras implementações, o sistema poderia disponibilizar apenas uma quantidade de reduzida de tempo , para completar o pagamento, e caso não o completasse nesse intervalo seria efetuado um rollback relativo a todas as operações daquela reserva.

Falhas e Erros no Sistema

A ocorrência de falhas e erros no sistema, pode estar relacionada com diversos fatores, nomeadamente problemas de pedidos à base de dados ou erros internos da API.

Para lidarmos com estes problemas, todos os endpoints que necessitam de receber parâmetros no seu pedido, têm implementado uma proteção relativa aos inputs, devolvendo ao utilizador o status e o erro associado, como também foi implementada para cada execução dos pedidos à base de dados, o mecanismo de try except, para que caso exista algum erro interno na API ,o mesmo, possa ser feito o Rollback da transação ,caso se aplique, e a devolução do status e erro respectivo.

Descrição dos testes de avaliação de qualidade

Para a realização dos testes foi utilizado o Postman, onde foram passados ficheiros JSON e/ou Token para cada endpoint, caso se aplique, para testar todos os endpoints.

Com o intuito de testar todas as verificações foram alterados os ficheiros JSON, utilizando token de users que não têm permissão para o endpoint em questão, como também foram inseridos valores inválidos no JSON, para avaliar a qualidade do sistema no caso de inputs inválidos.

Trabalho desenvolvido para a 2ª meta

- Modelo ER final [contribuição da equipa - 5h]
- Código API:
  - Implementação dos endpoints REST e as respetivas funcionalidades e prevenção de falhas [Francisca e Pedro - 30h]
- Documentação das implementações [contribuição da equipa - 2h]
- Criação do manual de instruções [Ramyad - 15h]
- Testar o sistema:
  - Criação de scripts [Francisca - 2h]
  - Verificação da qualidade das funcionalidades [Francisca - 5h]
- Explicação de como o sistema lida com transações, concorrência, falhas e erros [contribuição da equipa - 3h]

Webgrafia

Foram consultadas as documentações das bibliotecas utilizadas como também, a documentação do POSTgresSQL.

Para nos auxiliar na criação, verificação dos tokens como também na encriptação da password, utilizá-mos o ChatGPT, como linha de pensamento.

- https://chatgpt.com/
- https://www.w3schools.com/postgresql/index.php
- https://docs.docker.com/
- https://learning.postman.com/docs/introduction/overview/