Universidade de Coimbra

Faculdade de Ciências e Tecnologia

Licenciatura em Engenharia Informática

Bases de Dados – 2020/2021

3º Ano – 2º Semestre

PROJETO E BASES DE DADOS 2020/2021

META 2

Coimbra, 30 de maio de 2021

2018298731	João Pedro Pacheco Silva	joaopedro@student.dei.uc.pt
2018275587	Rui Alexandre Vale Alves Azevedo Abreu	ruiabreu@student.dei.uc.pt

Manual de utilizador

Na pasta “postman” encontrará uma coleção de pedidos HTTP que poderá importar para o postman e usar para testar as funcionalidades da aplicação. Para usar as funcionalidades os pedidos devem ter o seguinte formato:

1. **Registo de utilizadores.**

   Método: POST

   Url: http://localhost:8080/dbproj/user

   Body: {“username”: username, “email”: email, “password”:password}

2. **Autenticação de utilizadores.**

   Método: PUT

   Url: http://localhost:8080/dbproj/user

   Body: {“username”: username, “password”:password}

3. **Criar um novo leilão.**

   Método: POST

   Url: http://localhost:8080/dbproj/leilao

   Body: {“artigoId”: artigoId1, “precoMinimo”: preco, “titulo”: “Titulo do Novo Leilão”, “descricao”: “Descrição do Novo Leilão”, “fim”: “fim do leilão”, "authToken": “token de autenticação”}

4. **Listar todos os leilões existentes.**

   Método: GET

   Url: http://localhost:8080/dbproj/leiloes

5. **Pesquisar leilões existentes.**

   Método: GET

   Url: http://localhost:8080/dbproj/leiloes/{keyword}

6. **Consultar detalhes de um leilão.**

   Método: GET

   Url: http://localhost:8080/dbproj/leilao/{leilaoId}

7. **Listar todos os leilões em que o utilizador tenha atividade.**

    Método: POST

    Url: http://localhost:8080/dbproj/user/leiloes

    Body: { "authToken": “token de autenticação”}

8. **Efetuar uma licitação num leilão.**

    Método: GET

    Url: http://localhost:8080/dbproj/licitar/{leilaoId}/{licitacao}

    Body: { "authToken": “token de autenticação”}

9. **Editar propriedades de um leilão.**

    Método: GET

    Url: http://localhost:8080/dbproj/leilao/{leilaoId}

    Body: { "authToken": “token de autenticação”}

10. **Escrever mensagem no mural de um leilão.**

    Método: POST

    Url: http://localhost:8080/dbproj/mensagem/{leilaoId}

    Body: { "authToken": “token de autenticação”}

11. **Listar versões anteriores do leilão.**

    Método: GET

    Url: http://localhost:8080/dbproj/versoes/{leilaoId}

    Body: { "authToken": “token de autenticação”}

12. **Listar notificações do utilizador.**

    Método: GET

    Url: http://localhost:8080/dbproj/notificacoes

    Body: { "authToken": “token de autenticação”}

13. **Terminar leilão e determinar o vencedor.**

    Método: GET

    Url: http://localhost:8080/dbproj/terminar/{leilaoId}

14. Registar um produto.

Método: POST

Url: http://localhost:8080/dbproj/artigo

Body: {“ean”: “código EAN”}

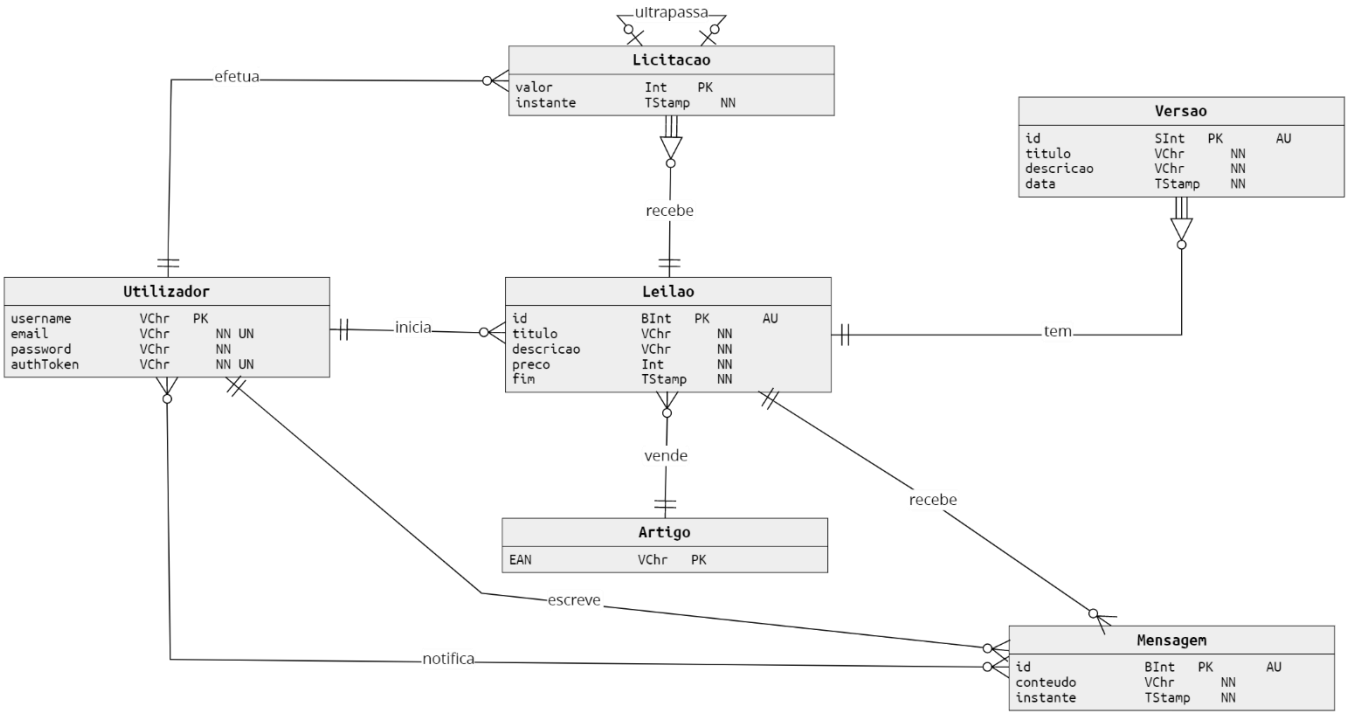
Todos os pedidos HTTP que necessitem da autenticação do utilizador, como por exemplo editar um leilão, escrever uma mensagem ou fazer uma licitação, devem incluir o token de autenticação fornecido na etapa de autenticação (request 2).

Manual de instalação

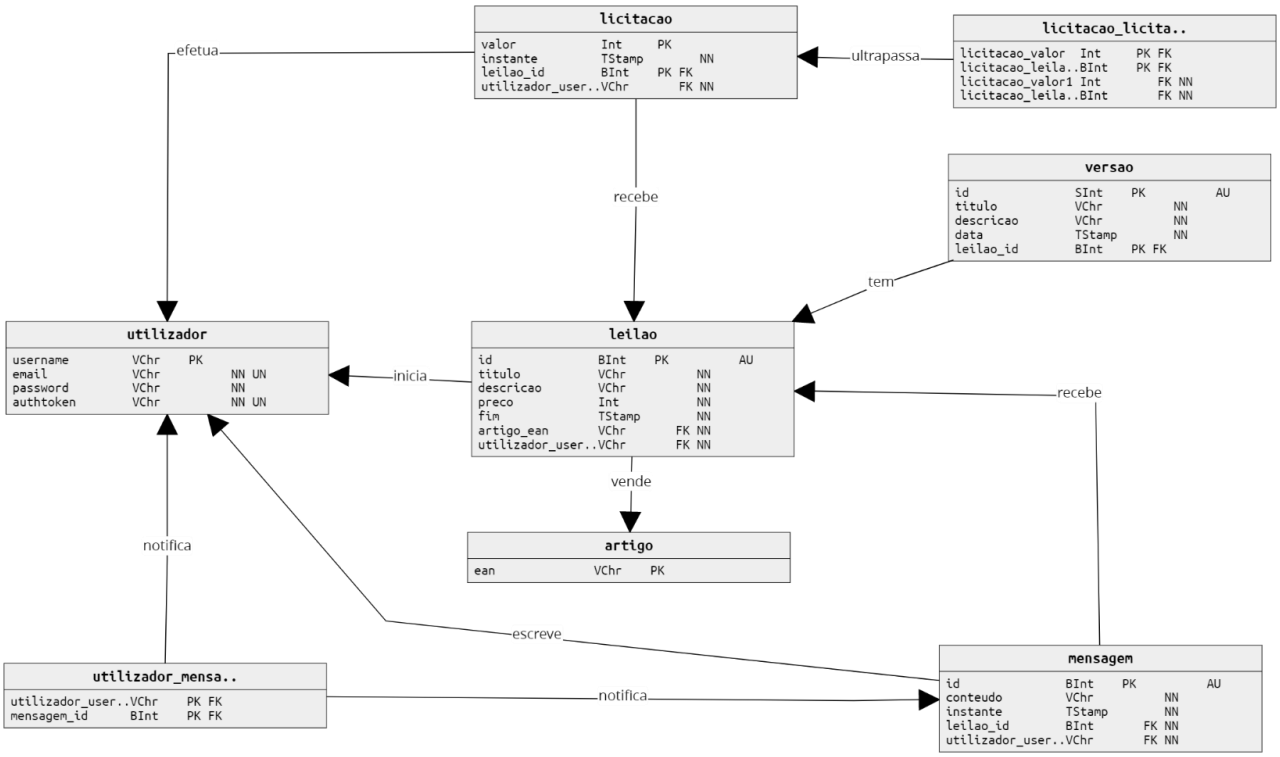
Para realizar a instalação do software pode usar uma das seguintes abordagens:

1. Windows: Abra o ficheiro "setup.bat"
2. Mac: Abra o ficheiro "setup.command"
3. Abra um terminal na pasta do projeto e execute o comando "docker compose -f docker-compose-python-psql.yml up --build"

Modelo ER



Tabelas



Plano de desenvolvimento

Tarefa	Realizador	Esforço
Registo de utilizadores	Rui	30m
Autenticação dos utilizadores (login e validação dos tokens)	João	2h
Criação de um leilão	João	30m
Listar os leilões existentes	Rui	30m
Pesquisar os leilões existentes	Rui	1h
Consultar os detalhes do leilão	Rui	2h
Listar os leilões que o utilizador tenha atividade	João	2h
Efetuar licitação num leilão	João	2h30m
Notificar licitação ultrapassada	João	1h
Editar propriedades de um leilão	Rui	1h30m
Guardar versões anteriores de um leilão	Rui	1h30m
Listar versões anteriores do leilão	Rui	30m
Escrever mensagens no mural do leilão	João	30m
Notificar mensagens novas	João	1h
Listar notificações do utilizador	João	1h30m
Terminar leilão e determinar o vencedor	Rui	1h30m
Registar um produto	Rui	30m
Manual de utilizador	Rui	30m
Plano de desenvolvimento	Rui	30m
Detalhes da implementação	João	1h
Análise das queries e escolha dos índices	João	2h
Controlo de concorrência	João	1h
testes	Rui + joão	5h

Detalhes da implementação

Arquitetura:

A arquitetura do projeto divide-se em dois grandes componentes, um Data Base Server onde é guardada e gerida toda a informação do sistema de leilões através do DBMS PostgreSQL e um Web Server que recebe pedidos HTTP e interage com o Data Base Server através de uma interface SQL implementada em python.

O Web Server é o intermediário entre os utilizadores e o Data Base Server, sendo este responsável por traduzir cada pedido HTTP na query SQL adequada. Para além do tratamento de pedidos, o Web Server é também responsável pela implementação dos mecanismos de segurança, nomeadamente a encriptação das passwords e criação de tokens de autenticação. A encriptação das passwords é feita recorrendo a uma função de hash do modulo "hashlib". Esta é usada sobre a password de forma gerar um código hash que será guadado no seu desta. Sempre que houver um pedido de autenticação a password é usada como input da função hash e o código resultante é comparado com o conteúdo armazenado no Data Base Server. Relativamente aos tokens de autenticação, estes são gerados e guardados na base dados de dados sempre que o utilizador faz login. Para gerar cada token é usa-se o username seguido de um número aleatório como input de uma função hash. O output da função será o token a ser utilizado nos pedidos subsequentes.

Scripts de criação da base de dados:

O código usado na criação da base de dados encontra-se dividido por vários ficheiros, os quais podem ser encontrados na pasta "postgresql".

No ficheiro "leilao_triggers_and_functions.sql", é onde está todo o código relativo às funções e triggers usados sobre a tabela leilão. Nele temos uma a função chamada "close_leilao" que é responsável por terminar um leilão e determinar o vencedor do mesmo, temos também o trigger "before_close" que serve para ver se um leilão já pode ser terminado, temos o trigger "after_edit", que guarda a versão anterior de um leilão sempre que este é editado e temos o trigger "check_fim", que impede a inserção de leiloes com datas de fim anteriores à data atual.

No ficheiro "licitação_triggers.sql" é onde está todo o código relativo aos triggers usados na tabela licitação. Nele temos o trigger "before_licitacao" que faz todas as verificações necessárias para validar uma licitação antes desta ser inserida na tabela e temos o trigger "after_licitacao", responsável por notificar o utilizador com o lance ultrapassado (um utilizador não é notificado caso este ultrapasse o próprio lance).

No ficheiro "mensagem_triggers.sql" é onde está todo o código relativo aos triggers usados na tabela mensagem. Nele temos trigger "after_mensagem" responsável por notificar o vendedor e todos os utilizadores envolvidos no mural de um leilão sempre que escrita uma nova mensagem nesse mesmo mural (os utilizadores não recebem notificação das próprias mensagens).

No ficheiro "authentication.sql" é onde estão as funções relativas à validação dos tokens de autenticação.

No ficheiro "índexes.sql" é onde está todo código relativo à criação dos índices da base de dados. Estes foram escolhidos tendo como base as diferentes queries SQL usadas na aplicação, de forma a tentar melhorar a performance das mesmas.

Por fim, temos o ficheiro ”tabels.sql” onde está todo o código relativo à criação das tabelas e respetivas constraints, o ficheiro “initialize.sql” que junta todo o código dos ficheiros vistos até agora e o ficheiro “drop_tables.sql”, onde existe um conjunto de comandos para remover todas as tabelas da base de dados.

Escolha dos índices

Em relação à tabela “mensagem”, foi criado um índice sobre o conjunto (leilao_id, utilizador_username), de forma a melhorar a listagem de mensagens de um leilão e na listagem de leilões em que um utilizador esteja envolvido. Pelos mesmos motivos, criou-se também o conjunto (leilao_id, utilizador_username) para a tabela “licitacao”.

Relativamente à tabela “leilao”, foram criados 4 índices. Um sobre a coluna “fim”, para melhorar a listagem de leilões a decorrer, outro sobre a coluna “utilizador_username” para ajudar na listagem de leilões em que um utilizador esteja envolvido e um ultimo sobre o conjunto (descricao, artigo_ean), para as pesquisas de leilões por artigo ou por descrição.

Por fim, na tabela “versao” criou-se um índice na coluna “leilão_id” de forma a melhorar a listagem das versões de um leilão.

Controlo de concorrência

Para fazer o controlo de concorrência na funcionalidade de licitar, usámos um “select for update“ sobre todas as licitações do leilão. Dessa forma evitamos os casos em que a comparação feita com os lances anteriores é feita usando informação desatualizada. Pelos mesmos motivos usou-se o mesmo comando na determinação do vencedor do leilão.

Relativamente aos restantes casos o controlo de concorrência foi feita implicitamente ao chamar os comandos insert e update.

Validação dos dados

Para implementar as restrições mais complexas, fizemos uso dos triggers mencionados anteriormente. Sempre que os dados são classificados como inválidos é gerada uma exceção que anula todas as alterações realizadas durante a transação.