Databases Project
2020/2021

(2021/04/09: versão 2: clarificação das mensagens/notificações; pequenas correções)
(2021/03/09: versão draft do enunciado; pequenas correções/alterações/ajustes podem ainda a vir a ser realizadas nos próximos dias)

Introduction

The aim of this project is to provide students with experience in developing database systems. The project is oriented by the industry's best practices for software development; thus, students will experience all the main stages of a common software development project, from the very beginning to delivery.

Objectives

After completing this project, students should be able to:

- Understand how a database application development project is organized, planned and executed
- Master the creation of conceptual and physical data models for supporting and persisting application data
- Design, implement, test, and deploy a database system
- Install, configure, manage and tune a modern relational DBMS
- Understand client and server-side programming in SQL and PL/pgSQL (or similar).

Groups

The project is to be done by groups of 2 or 3 students. Depending on the size of the group, different functionalities should be developed (as defined later in this document). IMPORTANT: the members of each group **must** be enrolled in PL classes of the same Professor.

Quality attributes for a good project

Your application must make use of:

(a) A transactional DBMS (the use of PostgreSQL is optional)
(b) A distributed database application architecture, providing a REST API
(c) SQL and PL/pgSQL, or similar
(d) Adequate and relevant triggers, functions and procedures running on the DBMS side
(e) Good strategies for managing transactions and concurrency conflicts, and database security
(f) Good error avoidance, detection and mitigation strategies
(g) Good documentation

Your application must also respect the functional requirements defined in the companion document (in annex A) and execute without "visible problems" or "crashes". To fulfill the objectives of this assignment, you can be as creative as you want, provided you have included this list of defined features in your solution.

Milestones and deliverables

Midterm presentation (20% of the grade) – week of April 12th

The group must present their work in the PL classes (privately, to the professor of the class). The following artifacts should be developed and uploaded at *inforestudante* before the presentation at the PL class (each

group must select a member for performing this task, and all submissions must clearly identify the team and the students working in the project):

- **Presentation** (e.g., power point) with the following information:
  - Name of the project
  - Team members and contacts
  - Brief description of the project
  - Definition of the main database operations, transaction and potential concurrency conflicts
  - Description of a potential solution (if you already have one)
  - Core technologies: programming language, DBMS, Libraries, etc. The group is free to select the technologies to be used.
  - Development plan: planned tasks, initial work division per team member, timeline
- **ER diagram**
  - Description of entities, attributes, integrity rules, etc.
- **Relational data model** (tables)

Final delivery (80% of the grade) – May 31st

The project outcomes must be submitted to *inforestudante* until 23:55 in the day of the deadline. Each group must select a member for performing this task. All submissions must clearly identify the team and the students working in the project. Upload into *inforestudante* the following materials and documents:

- Document with:
  - **User manual** describing how users can interact with the application
  - **Installation manual** describing how to deploy and run the software you developed
  - **Final ER and relational data models**
  - **Development plan:** make sure you specify which tasks were done by each team member and the effort involved (e.g., hours)
  - **All the information you consider relevant to understand how the application is built**
- **Source code and Scripts:**
  - Include the source code, scripts, executable files and libraries necessary for compiling and running the software (identify the DBMS used, do not upload its binaries)
  - DB creation scripts containing the definitions of tables, constraints, sequences, users, roles, permissions, triggers, functions, and procedures

Defense – June 1st to 4th

- Prepare a 10 min live presentation of your software
- Prepare yourself to answer questions regarding all deliverables and implementation details
- Sign up for any available time slot for the defense in *inforestudante*
- The list of available slots will be released in before the deadline for the final delivery
- Sign up until the final delivery due date

Assessment

- This project accounts for 8 points (out of 20) of total grade in the Databases course
- Midterm submission corresponds to 20% of the grade of the project
- Final submission accounts for the remaining 80% of the grade of the project
- The minimum grade is 35%

Notes

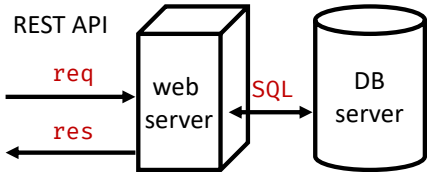
- Do not start coding right away - take time to think about the problem and to structure your development plan and design
- Always implement the necessary code for **error detection and correction**
- Aspects related to **concurrency management**, **security**, and **best coding practices** will be valued
- Assure a **clean shutdown of your system** (no memory-leaks)
- **Plagiarism or any other kind of fraud will not be tolerated**

Leilões Online – Descrição Funcional

Este projeto consiste em desenvolver um sistema típico de leilão, suportado por um sistema de gestão de base de dados. Um leilão é iniciado por um vendedor que define um artigo, indica o preço mínimo que está disposto a receber, e decide o momento em que o leilão vai terminar (data, hora e minuto). Os compradores fazem licitações que vão sucessivamente aumentando o preço até ao término do leilão. Ganha o comprador que licitar o valor mais alto.

Para simplificar a escolha do artigo que vai a leilão considera-se que cada artigo tem um código que o identifica univocamente (por exemplo, o código EAN de 13 dígitos vulgarmente encontrado junto com o código de barras dos artigos, ou o código ISBN de 10 ou 13 dígitos habitualmente usado para identificar livros). Para iniciar um novo leilão, um utilizador escolhe um artigo, limita o preço mínimo que pretende receber, e indica a data, hora e minuto em que o leilão irá terminar.

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`). A figura representa uma vista simplificada do sistema a desenvolver. Como podemos observar, o utilizador interage com o *web server* através da troca de *request*/*response* REST e por sua vez o *web server* interage com o servidor de base de dados através de uma interface SQL (e.g., JDBC no caso do Java, Psycopg no caso de Python).



Esta é uma das arquiteturas mais utilizadas atualmente e suporta muitas das aplicações web e mobile que usamos no dia a dia. Uma vez que o foco da disciplina está nos aspetos de bom desenho de uma base de dados e das funcionalidades associadas, o desenvolvimento de aplicações web ou mobile **está fora do âmbito deste trabalho**. Para usar ou testar a sua API REST, pode usar um cliente REST tal como o `postman.com` ou o `curl.se`. Nos casos em que o formato do pedido (`req`) e da resposta (`res`) estiver especificado nos parágrafos abaixo, estes devem ser seguidos rigorosamente. Nos casos em que não estiverem definidos, o grupo deve especificá-los e incluir a sua definição no relatório.

Importante: a lógica do sistema (e.g. pesquisas) deve ser implementada nas *queries* SQL e não no web server!

Funcionalidades a desenvolver

Quando um utilizador começa a utilizar o sistema de leilões deve poder escolher entre registar uma nova conta e entrar com uma conta existente, tirando partido dos seguintes *endpoints*:

Registo de utilizadores. Criar um novo utilizador, inserindo os dados requeridos pelo modelo de dados.

Done

```
req   POST http://localhost:8080/dbproj/user
      {"username": username, "email": email, "password":password}
      /** i.e. dados do utilizador a criar, de acordo com o vosso modelo **/
res   {"userId": novoUserId} Ou em caso de erro {"erro": codigoErro}
```

Autenticação de utilizadores. Login com *username* e *password*, recebendo um *token* de autenticação em caso de sucesso, *token* esse que deve ser incluído nas chamadas subsequentes.

Done

```
req   PUT http://localhost:8080/dbproj/user
      {“username”: username, “password”:password}
      /** i.e. User data to authenticate[1] **/
res   {“authToken”: authToken} Ou em caso de erro {“erro”: AuthError}
```

Após autenticação, o utilizador poderá realizar as seguintes operações:

Criar um novo leilão. Cria-se um leilão começando por identificar o artigo que se pretende comprar. Para simplificar, considera-se que cada artigo tem um código EAN/ISBN que o identifica univocamente. Cada leilão deve igualmente ter um título, uma descrição e quaisquer detalhes adicionais que considere necessários. Para criar o leilão, o vendedor indica o preço mínimo que está disposto a receber, bem como a data, hora e minuto em que o leilão termina.

Done

```
req   POST http://localhost:8080/dbproj/leilao
      {“artigoId”: artigoId1, “precoMinimo”: preco, “titulo”: “Titulo do Novo Leilão”,
      “descricao”: “Descrição do Novo Leilão”, (...)}
res   Em caso de sucesso
      {“leilaoId”: novoLeilaoId}
      Ou em caso de erro
      {“erro”: codigoErro}
```

Listar todos os leilões existentes. Deve poder-se listar os leilões que estão a decorrer, obtendo uma lista de identificadores e descrições.

Dine

```
req   GET http://localhost:8080/dbproj/leiloes
res   [
         {“leilaoId”: leilaoId1, “descricao”: “Descrição do Leilão 1”},
         {“leilaoId”: leilaoId2, “descricao”: “Descrição do Leilão 2”},
         {“leilaoId”: leilaoId3, “descricao”: “Descrição do Leilão 3”}
      ]
```

Pesquisar leilões existentes. Deve poder-se listar os leilões que estão a decorrer, pesquisando pelo código EAN/ISBN ou pela descrição do artigo. Esta listagem apresenta o identificador e descrição de cada leilão que obedeça ao critério da pesquisa.

Falta perguntar se so procura os que estão a decorrer ou se procura todos

```
req   GET http://localhost:8080/dbproj/leiloes/{keyword}
res   [
         {“leilaoId”: leilaoId1, “descricao”: “Descrição do Leilão 1”},
         {“leilaoId”: leilaoId3, “descricao”: “Descrição do Leilão 3”}
      ]
```

Consultar detalhes de um leilão. Para qualquer leilão escolhido, deve poder-se obter todos os detalhes relativos à descrição do artigo, ao término do leilão, às mensagens escritas no seu mural (ver abaixo) e ao histórico de licitações efetuadas nesse mesmo leilão.

```
req   GET http://localhost:8080/dbproj/leilao/{leilaoId}
res   {“leilaoId”: leilaoId1, “descricao”: “Descrição do Leilão 1”, /** OS RESTANTES
      DETALHES e MENSAGENS**/}
```

Listar todos os leilões em que o utilizador tenha atividade. Um utilizador deve poder listar os leilões nos quais tem ou teve alguma atividade, seja como criador do leilão seja como licitador. Esta listagem sumaria os detalhes de cada leilão.

Done

Efetuar uma licitação num leilão. Um comprador pode licitar com um preço mais alto num determinado leilão, desde que o leilão não tenha terminado e que não haja uma sua licitação mais alta do que a que está a fazer e seja, pelo menos, superior ao preço mínimo.

Falta por que o valor nao pode ser menor que o valor minimo

[1] Num ambiente real seria recomendado reutilizar um esquema de autenticação mais bem estabelecido em vez de desenvolver um, mas neste caso é melhor simplificar. O método PUT é utilizado pois a *password* não deve ser enviada por GET.

req **GET** http://localhost:8080/dbproj/licitar/{leilaoId}/{licitacao}

res Sucesso ou código de erro

Editar propriedades de um leilão. O vendedor pode ajustar todas as descrições textuais relativas a um leilão seu, sendo que todas as versões anteriores devem ficar guardadas e poder ser consultadas posteriormente para referência. Done

req **PUT** http://localhost:8080/dbproj/leilao/{leilaoId}

```
{/** informaçãoa ser alterada e.g. **/  "titulo": "Novo titulo", (...)}
```

res *Em caso de sucesso*

```
{"leilaoId": novoLeilaoId, /** informação completa do leilao**/}
```

Ou em caso de erro

```
{"erro": codigoErro}
```

Escrever mensagem no mural de um leilão. Cada leilão deve ter um "mural" onde poderão ser escritos comentários, questões e esclarecimentos relativos ao leilão.

Entrega imediata de notificações a utilizadores. Os utilizadores recebem imediatamente na sua caixa de entrada as notificações acerca das mensagens publicadas, e deverão estar disponíveis no *endpoint* correspondente. O criador de um leilão é notificado de todas as mensagens relativas a esse leilão. Todos os utilizadores que tiverem escrito num mural passam a ser notificados acerca de mensagens escritas nesse mesmo mural.

Notificação de licitação ultrapassada. Um comprador que tenha feito uma licitação num leilão recebe uma mensagem na sua caixa de mensagens sempre que houver outra licitação melhor que a sua.

Término do leilão na data, hora e minuto marcados. No momento indicado pelo vendedor (data, hora e minuto) o leilão termina. Determina-se aí o vencedor e fecha-se a possibilidade de realizar mais licitações. Os detalhes desse leilão são atualizados e podem ser consultados posteriormente.

(só para Grupos de 3) **Um administrador pode cancelar um leilão.** Um administrador deve poder cancelar um leilão se tal for necessário. O leilão continua a poder ser consultado pelos utilizadores, mas está dado como encerrado e não podem ser feitas licitações. Todos os utilizadores interessados recebem uma notificação.

(só para Grupos de 3) **Um administrador pode banir permanentemente um utilizador.** Um administrador deve poder banir um utilizador se tal for necessário. Todos os leilões criados por esse utilizador são cancelados. Todas as licitações efetuadas por esse utilizador devem ser invalidadas (ainda que mantidas nos registos). Note que, ao invalidar uma licitação num leilão, quaisquer licitações superiores a essa devem ser igualmente invalidadas exceto a melhor delas, cujo valor se torna igual ao valor da que for invalidada. Automaticamente é criada uma mensagem no mural dos leilões afetados lamentando o incómodo e todos os utilizadores envolvidos devem receber uma notificação.

(só para Grupos de 3) **Um administrador pode obter estatísticas de atividade na aplicação.** Um administrador deve poder consultar estatísticas da utilização da aplicação: top 10 utilizadores com mais leilões criados, top 10 utilizadores que mais leilões venceram, número total de leilões nos últimos 10 dias.