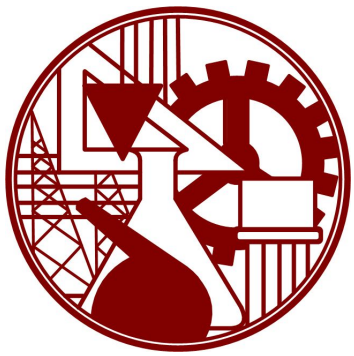
CADEIA DE RESTAURANTES

Interface de acesso a base de dados da cadeia de restaurantes.

ISEL | INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

LEIM | LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA

SBD | SISTEMAS DE BASES DE DADOS | T2



ANDRÉ FONSECA | 39758

a39758@alunos.isel.pt ou afbfonseca@gmail.com

27/01/2017

ÍNDICE

INTRODUÇÃO

O presente relatório apresenta uma descrição geral do projecto final proposto na disciplina de Sistemas de Bases de Dados que sucede ao primeiro projecto de modelação de uma base de dados para um cadeia de restaurantes. Foi implementada uma estrutura de dados para o armazenamento de dados para sustentar o fluxo de operações de vários restaurantes e funcionalidades desejadas.

Este projecto tem como base a implementação de uma solução através de uma aplicação funcional que sirva não só a população em geral na realização encomendas a restaurantes, mas como também no auxílio da gestão interna dos mesmos.

Para o desenvolvimento da aplicação foi proposto o uso da tecnologia Java e da interface JDBC[1] que permite realizar o acesso aos sistemas de gestão de bases de dados.

De forma a que o conteúdo aplicacional possa ser legível por qualquer pessoa, com conhecimentos técnicos para tal, a língua utilizada no desenvolvido da aplicação foi o Inglês.

[1] Java database connectivity [link]

OBJECTIVOS

A base de dados do projecto anterior foi realizada através da concepção de um modelo entidade-associação e de um modelo relacional tendo como base os requisitos funcionais. Com a estrutura de dados definida e implementada é possível desenvolver uma aplicação em que qualquer utilizador, com as devidas permissões de acesso, possa consultar dados indo de encontro aos requisitos funcionais.

Esta implementação consolida os conceitos de acesso a uma base de dados por uma aplicação através de tecnologias SQL[2] e JDBC.

[2] Structured query language [link]

FUNCIONALIDADES

A aplicação pretende servir não só a população em geral, mas como também auxiliar a gestão interna da cadeia de restaurantes. Para tal, foram estabelecidos vários perfis de acesso e diferentes funcionalidades.

O perfil de cliente (client) trata-se de qualquer pessoa que pretenda realizar um pedido ou uma encomenda a um determinado restaurante. Esta pode ser uma pessoa anônima ou previamente registada. Cada cliente tem ao seu dispor um conjunto de funcionalidades:

- Visualização de um cardápio;
- Realizar ou alterar um pedido. Este pedido pode ser feito presencialmente ou em serviço de take-away;
- Consultar o estado actual do pedido;
- Consultar os produtos ou pratos que contêm ou não um determinado ingrediente;
- Visualizar a receita de um prato;
- Localizar o restaurante mais próximo da sua localização.

O funcionário da cozinha (chef) que pode alterar o estado actual de pedidos já realizados.

O funcionário das mesas (waitress) que pode alterar o estado actual de pedidos prontos a serem entregues.

O gestor (owner) ou responsável pela gestão de um resturantes tem acesso as seguintes funcionalidades:

- Visualizar o stock actual de ingredientes e realizar as respectivas encomendas a fornecedores;
- Consultar os três produtos mais vendidos numa determinada semana;
- Consultar os três produtos mais lucrativos num determinado mês;
- Consultar os clientes que apresentem mais gastos num determinado trimestre;
- Visualizar os colaboradores responsáveis pela preparação de um determinado prato.

DESENVOLVIMENTO DA APLICAÇÃO

AMBIENTE DE DESENVOLVIMENTO

A implementação da base de dados foi realizada através do software MySQL versão 5.6.35-0 de 64-bit. O MySQL foi implementado por uma imagem de virtualização fornecida pela empresa Bitnami[3] através do software de virtualização VirtualBox versão 5.1.14.

Esta imagem de virtualização inclui o sistema operativo Ubuntu na versão 14.04 minimizada, incluindo apenas os serviços necessários ao funcionamento do MySQL.

O desenvolvimento da aplicação Java foi realizado na plataforma OpenJDK 8 (1.80_111). Esta escolha foi tomada devido à componente open-source e não comercial da tecnologia face à tecnologia Oracle JDK. A sua ligação é estabelecida pela interface MySQL-JDBC na versão 5.1.40.

Para produção de código foi utilizado o editor de texto Atom na versão 1.13 e Eclipse Neon sob o sistema operativo Fedora[4] 25.

DEPENDÊNCIAS

A única dependência presente na aplicação é a API JDBC[5], aqui implementada na versão 5.1.40 e que estabelece a ligação com a base de dados MySQL. O ficheiro .jar é adicionado ao projecto através do Eclipse, nas bibliotecas presentes no build path do projecto.

ARQUITECTURA DE SOFTWARE

A aplicação consiste na implementação de um objecto global 'Bot' que gere o processo de selecção do perfil a ser iniciado com base na resposta do utilizador.

Cada um dos perfis comunica com o objecto de acesso à base de dados que, por sua vez, realiza os pedidos através da execução de *stored procedures* definidos no objecto 'SBD DAO'.

O objecto MySQL é responsável por estabelecer a ligação com a base de dados, sendo possível futuras implementações de outros tipos de sistemas de gestão de bases de dados.

Diagrama de comunicação entre objectos

[3] Imagem de virtualização Bitnami [link]
[4] Sistema operativo baseado na kernel Linux de código aberto e produzido pela empresa Red Hat
[5] Interface de ligação JDBC [link]

INTERFACE

A interface da aplicação apresenta uma consola de texto que tem como objectivo a simulação de uma conversação entre um agente com um nível de inteligência limitado, mas suficiente para reconhecer os pedidos do utilizador e proporcionar uma interatividade o mais humana possível.

Esta escolha recai sobre o conceito de pessoas sem conhecimentos tecnológicos para que possam utilizar a aplicação sem terem de seleccionar opções e escolhas de uma forma mecânica. Desta forma, podem fazê-lo de uma forma mais pessoal e passível de interação humana, através de uma implementação minimalista de "linguagem de processamento natural".

O reconhecimento de instruções é realizado através da comparação das palavras introduzidas pelo utilizador e com as palavras que estão presentes em cada linha de instrução. Caso exista uma igualdade ou semelhança a instrução é reconhecida e processada.

De forma a estabelecer um contacto mais próximo com o utilizador o agente vai "falando" com o mesmo, como é o caso da mensagem de abertura da aplicação que é seleccionada aleatoriamente.

Cliente a interagir com o agente

CONSULTA DE DADOS

Para que seja possível invocar instruções SQL - e posteriormente modificá-las sem que haja necessidade de recompilar a aplicação - todas as operações entre a aplicação e a base de dados são realizadas através de *stored procedure's*.

O objecto MySQL contém as informações de acesso e o método estático que inicia e devolve a ligação estabelecida com a base de dados.

```
11.     private static final String ip = "192.168.1.134";
12.     private static final String port = "3306";
13.     private static final String db_name = "sbd";
14.     private static final String user = "root";
15.     private static final String pswd = "bitnami";
16.
17.     public static Connection get_connection() throws SQLException {
18.
19.         String c_string = String.format("jdbc:mysql://%s:%s/%s", ip, port, db_name);
20.
21.         return DriverManager.getConnection(c_string, user, pswd);
22.     }
```

Para testar a ligação a base de dados e execução de *stored procedure's* foram criados dois métodos estáticos em que o primeiro executa uma simples query de consulta e o último realiza a execução de uma *stored procedure* presente na base de dados.

```
23.
24.     public static boolean test_connection() {
25.         Connection connection;
26.         Statement stmt;
27.         ResultSet rs;
28.         String query = "SELECT id from client";
29.
30.         try {
31.             connection = MySQL.get_connection();
32.             stmt = connection.createStatement();
33.             rs = stmt.executeQuery(query);
34.
35.             rs.close();
36.             stmt.close();
37.             connection.close();
38.
39.         } catch (Exception e) {
40.             return false;
41.         }
42.
43.         return true;
44.     }
45.
46.     public static boolean test_stored_procedure() {
47.
48.         String query = "{CALL get_orders(?, ?)}";
49.
50.         try (
51.             Connection conn = MySQL.get_connection();
52.             CallableStatement cstmt = conn.prepareCall(query);
53.         ) {
54.             cstmt.setDate("in_date", java.sql.Date.valueOf("2017-01-20"));
55.             cstmt.setString("in_order_state", "Ready");
```

```
56.             cstmt.execute();
57.             ResultSet rs = cstmt.getResultSet();
58.
59.             while (rs.next()) {
60.
61.                 System.out.format(
62.                         "%02d %02d %02d %s\n",
63.                         rs.getInt("client_order_id"),
64.                         rs.getInt("MAX(order_state_id)"),
65.                         rs.getInt("employee_id"),
66.                         rs.getDate("date")
67.                         );
68.             }
69.         } catch (Exception e) {
70.             return false;
71.
72.         }
73.
74.         return true;
75.     }
```

A criação de todos os *stored procedure's* é realizada de igual forma; como exemplo: a consulta que permite obter os dados do dono de um restaurante.

```
1.  USE SBD;
2.
3.  CREATE PROCEDURE `get_owner` (IN in_restaurant_id int)
4.  BEGIN
5.      SELECT id, full_name, email, mobile_number FROM employee WHERE restaurant_id = in_restaurant_id
    AND owner = true;
6.  END
```

IMPLEMENTAÇÃO TÉCNICA

A implementação da bases de dados requer de uma instância de base de dados MySQL com a respectiva base de dados que pode ser montada através do script realizado no projecto anterior. Esta instância pode ser instalada nativamente ou numa máquina virtual, não existindo qualquer requisito especial quanto à sua implementação.

Quanto à execução da aplicação esta requer apenas a instalação da tecnologia Java na versão 8 ou posterior e a actualização do endereço de *ip* da máquina onde a base de dados irá estar alojada.

FUNCIONALIDADES FUTURAS

A principal funcionalidade que futuramente pode ser implementada é, sem dúvida, o melhoramento do agente, através de técnicas de inteligência artificial como “Natural language processing” - que reconheça facilmente as intenções do utilizador ao usar a aplicação.

Outras melhorias passam por exportar a aplicação desktop para uma aplicação móvel e integrar a mesma com outras aplicações de interacção com o utilizador, como é o caso do equipamento Amazon Alexa e Google Home ou, até mesmo, em sistemas de controlo de bordo de automóveis.

CONCLUSÃO

A primeira fase do projecto teve como principal objectivo a modelação de uma base de dados com os respectivos scripts de criação, o preenchimento de dados e a eliminação dos mesmos.

A segunda fase consolida os conceitos anteriores e dá a conhecer a implementação de como uma aplicação Java comunica com a base de dados, e, também, como é possível aceder, consultar e modificar dados e manter as relações de dados estabelecidos previamente.

Apesar de existirem outras alternativas, a utilização da linguagem SQL e as bases de dados relacionais permanecem como as opções mais viáveis para a armazenação de dados. Quanto à utilização da linguagem de programação Java esta poderá ficar à descrição do aluno, de acordo com a sua preferência por uma outra tecnologia para a realização do mesmo projecto.