

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА № 9**

**з кредитного модуля «Проектування корпоративних інформаційних систем» на тему:**

«Створення проектів з БД в Eclipse. Використання JDBC»

Виконав студент групи КП-61с

Литвин Вадим Олександрович

Київ 2017

Мета роботи: Використання технології JDBC для доступу до даних БД.

Тема «ІС хлібозаводу».

Вихідний код проекту: <https://github.com/VadyaVL/CIS-Bakery>

**Хід роботи**

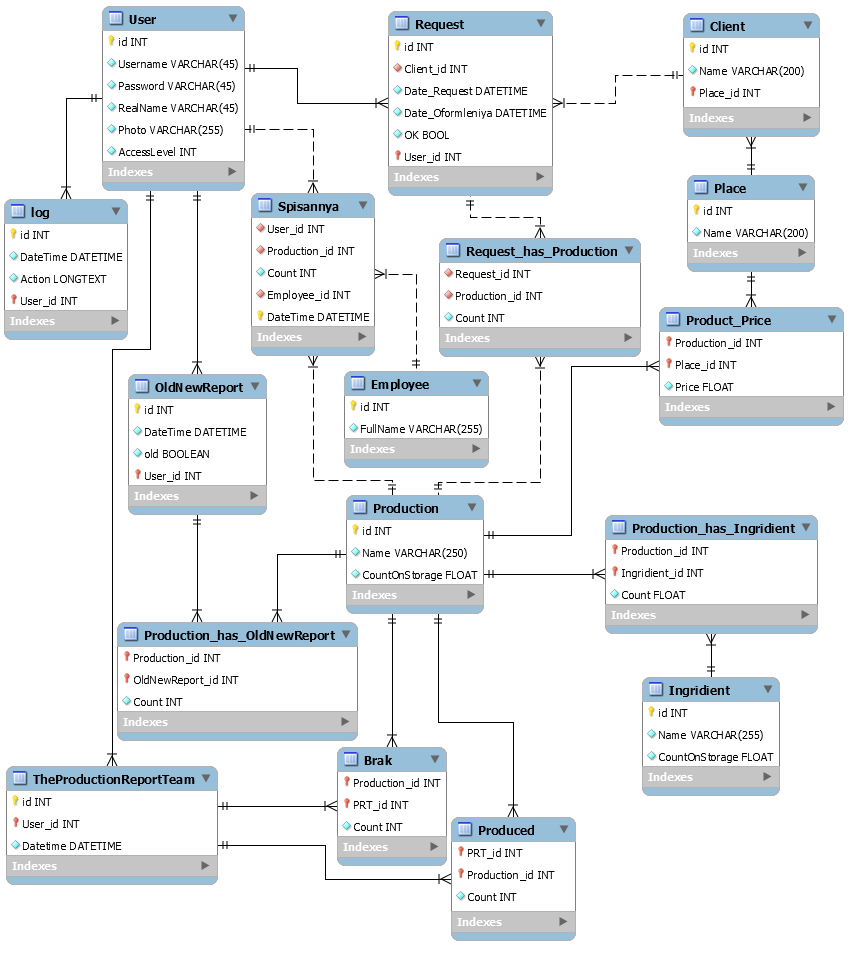


Рис. 1 Схема БД «ІС хлібозаводу»

На основі створеної в попередніх лабораторних роботах моделі та схеми бази даних, що зображена на рис. 1, опишемо класи мовою програмування java в середовищі програмування Eclipse. Для даної роботи будемо використовувати повну версію даної бази даних.

Використовувана СУБД в даній роботі – MySQL.

На рис. 2 відображено структуру проекту даної лабораторної роботи.

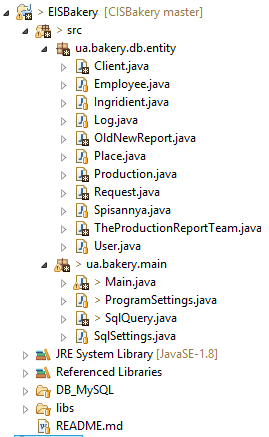


Рис. 2 Структура jdbc проекту «ІС хлібозаводу»

Демонстрація роботи програми (вибірка клієнтів, та запитів першого клієнта):

Клієнт №1: "Кузьменко"

Клієнт №2: Тутченко

Клієнт №3: Магазин Бам

Клієнт №4: Шевченко

Клієнт №5: Подорожня

Клієнт №6: Магазин Ірина

……………………………………………………………

Клієнт №102: Петюх

Клієнт №103: Саша Ніч

Клієнт №104: Магазин Продукти

Клієнт №105: калинівка вергун

Клієнт №106: вергун першотравневе

Заявка №87: 2016-08-24 20:47:39.0: виконано

Заявка №168: 2016-08-25 20:18:26.0: виконано

Заявка №188: 2016-08-26 17:36:11.0: виконано

Заявка №277: 2016-08-27 17:24:02.0: виконано

Заявка №306: 2016-08-28 10:09:42.0: не виконано

Заявка №369: 2016-08-28 18:54:04.0: виконано

Заявка №390: 2016-08-29 16:42:11.0: виконано

Заявка №517: 2016-09-01 14:25:38.0: виконано

Заявка №518: 2016-09-01 14:25:44.0: виконано

Заявка №519: 2016-09-01 15:50:20.0: виконано

Заявка №520: 2016-09-05 21:00:07.0: не виконано

Заявка №521: 2016-09-05 21:00:55.0: не виконано

Заявка №522: 2016-09-05 21:02:47.0: не виконано

Заявка №524: 2016-09-17 17:05:20.0: виконано

Заявка №525: 2016-09-17 17:08:48.0: виконано

Заявка №526: 2016-09-17 20:19:24.0: виконано

Заявка №527: 2017-03-04 20:10:00.0: не виконано

Заявка №528: 2017-03-04 20:13:28.0: не виконано

Нижче наведемо лістинг деяких файлів з проекту. Повний код лабораторної роботи можна побачити за посилання вказаним на початку даного документу.

|  |
| --- |
| Лістинг 1. Константи для налаштування з’єднання та запити |
| **public** **static** **final** String ***JDBC\_DRIVER*** = "com.mysql.jdbc.Driver";  **public** **static** **final** String ***IP*** = "127.0.0.1"; // або localhost  **public** **static** **final** String ***PORT*** = "3306";  **public** **static** **final** String ***DB\_NAME*** = "Bakery";  **public** **static** **final** String ***DB\_URL*** = "jdbc:mysql://" + ***IP*** + ":" + ***PORT*** + "/" + ***DB\_NAME***;  **public** **static** **final** String ***USERNAME*** = "root";  **public** **static** **final** String ***PASSWORD*** = "1111";  **public** **final** **static** String ***SELECT\_USER*** = "SELECT \* FROM User";  **public** **final** **static** String ***SELECT\_CLIENT*** = "SELECT \* FROM Client";  **public** **final** **static** String ***SELECT\_LOG*** = "SELECT \* FROM Log";  **public** **final** **static** String ***SELECT\_LOG\_BY\_USER*** = "SELECT \* FROM Log WHERE User\_id = ?";  **public** **final** **static** String ***SELECT\_REQUEST\_BY\_CLIENT*** = "SELECT \* FROM Request WHERE Client\_id = ?";  **public** **final** **static** String ***INSERT\_REQUEST*** = "INSERT INTO Request (Client\_id, Date\_Request, Date\_Oformleniya, OK, User\_id) VALUES (?, ?, ?, ?, ?)";  **public** **final** **static** String ***DELETE\_REQUEST*** = "DELETE FROM Request WHERE id = ?"; |

В лістингу 2, 3 наведемо код двох класів, що відображають сутності БД. Лістинг 4 міститиме в собі Main клас, в я кому демонструється робота з БД (вибірка, вставка, видалення сутностей)

|  |
| --- |
| Лістинг 2. Request.java |
| **package** ua.bakery.db.entity;  **import** java.util.Date;  **import** java.util.LinkedHashMap;  **import** java.util.Map;  /\*\*  \*  \* **@author** Vadym  \*  \*/  **public** **class** Request {  **private** **int** id;  **private** Date dateRequest;  **private** Date dateOformleniya; // Дата оформлення  **private** **boolean** ok; // Чи виконано замовлення  **private** Client client;  **private** User user;    // Реалізація Many-to-Many з додатковим полем кількість  **private** Map<Production, Float> productionCount;      **public** Map<Production, Float> getProductionCount() {  **return** productionCount;  }  **public** **void** setProductionCount(Map<Production, Float> productionCount) {  **this**.productionCount = productionCount;  }  **public** Request(**int** id, Date dR, Date dO, **boolean** ok){  setId(id);  setDateRequest(dR);  setDateOformleniya(dO);  setOk(ok);  }    **public** Request(Date dR, Date dO, **boolean** ok){  **this**(-1, dR, dO, ok);  }    @Override  **public** String toString(){  **return** "Заявка №" + id + ": " + dateOformleniya + ": " + (ok ? "виконано": "не виконано");  }    **public** **int** getId() {  **return** id;  }    **public** **void** setId(**int** id) {  **this**.id = id;  }    **public** Date getDateRequest() {  **return** dateRequest;  }    **public** **void** setDateRequest(Date dateRequest) {  **this**.dateRequest = dateRequest;  }    **public** Date getDateOformleniya() {  **return** dateOformleniya;  }    **public** **void** setDateOformleniya(Date dateOformleniya) {  **this**.dateOformleniya = dateOformleniya;  }    **public** **boolean** isOk() {  **return** ok;  }    **public** **void** setOk(**boolean** ok) {  **this**.ok = ok;  }    **public** Client getClient() {  **return** client;  }    **public** **void** setClient(Client client) {  **this**.client = client;  }    **public** User getUser() {  **return** user;  }    **public** **void** setUser(User user) {  **this**.user = user;  }  } |

|  |
| --- |
| Лістинг 3. User.java |
| **package** ua.bakery.db.entity;  **import** java.util.List;  /\*\*  \* Відображає таблицю БД - користувачі.  \* **@author** Vadym  \*  \*/  **public** **class** User {    **private** **int** id;  **private** String username;  **private** String password;  **private** String realName;  **private** String photo; // Посилання (шлях) на фото  **private** **int** accessLevel; // Рівень прав доступу    **private** List<Log> logs; // Логи користувача    **public** User(**int** id, String username, String password, String realName, String photo, **int** accessLevel) {  setId(id);  setUsername(username);  setPassword(password);  setRealName(realName);  setPhoto(photo);  setAccessLevel(accessLevel);  }    **public** User(String username, String password, String realName, String photo, **int** accessLevel) {  **this**(-1, username, password, realName, photo, accessLevel);  }    @Override  **public** String toString(){  **return** "[" + **this**.getClass().getName() + "] - id: " + id + ", username: " + username + ", password: " +  password + ", real name: " + realName + ", photo: " + photo + ", access level: " + accessLevel + "\n" +  "Logs:\n" + getLogsAsString();  }    **public** String getLogsAsString(){  String result = "";    **for** (Log log : logs) {  result += log + "\n";  }    **return** result;  }    // Ділянка реалізації getters and setters    **public** **int** getId() {  **return** id;  }    **public** **void** setId(**int** id) {  **this**.id = id;  }    **public** String getUsername() {  **return** username;  }    **public** **void** setUsername(String username) {  **this**.username = username;  }    **public** String getPassword() {  **return** password;  }    **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** String getRealName() {  **return** realName;  }    **public** **void** setRealName(String realName) {  **this**.realName = realName;  }    **public** String getPhoto() {  **return** photo;  }    **public** **void** setPhoto(String photo) {  **this**.photo = photo;  }    **public** **int** getAccessLevel() {  **return** accessLevel;  }    **public** **void** setAccessLevel(**int** accessLevel) {  **this**.accessLevel = accessLevel;  }    **public** List<Log> getLogs(){  **return** logs;  }    **public** **void** setLogs(List<Log> logs){  **this**.logs = logs;  }  } |

|  |
| --- |
| Лістинг 4. Main.java |
| **package** ua.bakery.main;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.ResultSet;  **import** java.sql.Statement;  **import** java.util.ArrayList;  **import** java.util.Date;  **import** java.util.List;  **import** ua.bakery.db.entity.\*;  **public** **class** Main {  **private** **static** ArrayList<User> *users*;  **private** **static** ArrayList<Client> *clients*;  **private** **static** ArrayList<Request> *requests*;    **public** **static** **void** main(String[] args) {  *demonstrateSelectUser*();  *showList*(*users*);  *demonstrateSelectClientAndRequest*();  *showList*(*clients*);  *demonstrateSelectRequestByClient*(*clients*.get(0));  *showList*(*requests*);    // Додати заявку по юзеру і клієнту  *demonstrateInsertRequest*(*users*.get(0), *clients*.get(0));    // Видалити заявку юзера  *demonstrateRemoveRequest*(*requests*.get(*requests*.size()-1));  }    // Демонстарція вибірки клієнтів і всіх замовлень певного клієнта  **public** **static** **void** demonstrateSelectClientAndRequest(){  Connection conn = **null**;  Statement stmt = **null**;  ResultSet resultSet = **null**;  *clients* = **new** ArrayList<>();    **try** {  Class.*forName*("com.mysql.jdbc.Driver");  conn = DriverManager.*getConnection*(SqlSettings.***DB\_URL***, SqlSettings.***USERNAME***, SqlSettings.***PASSWORD***);  stmt = conn.createStatement();  resultSet = stmt.executeQuery(SqlQuery.***SELECT\_CLIENT***);    **while**(resultSet.next()){  **int** id = resultSet.getInt("id");  String name = resultSet.getString("name");    Client client = **new** Client(id, name);  *clients*.add(client);  }    resultSet.close();  stmt.close();  conn.close();  }  **catch** (Exception ex){  System.***out***.println(ex.getMessage());  }  }    **public** **static** **void** demonstrateSelectRequestByClient(Client client){  Connection conn = **null**;  PreparedStatement stmt = **null**;  ResultSet resultSet = **null**;  *requests* = **new** ArrayList<>();    **try** {  Class.*forName*("com.mysql.jdbc.Driver");  conn = DriverManager.*getConnection*(SqlSettings.***DB\_URL***, SqlSettings.***USERNAME***, SqlSettings.***PASSWORD***);  stmt = conn.prepareStatement(SqlQuery.***SELECT\_REQUEST\_BY\_CLIENT***);  stmt.setInt(1, client.getId());  resultSet = stmt.executeQuery();    **while**(resultSet.next()){  **int** id = resultSet.getInt("id");  Date dateReq = resultSet.getTimestamp("Date\_Request");  Date dateOf = resultSet.getTimestamp("Date\_Oformleniya");  Boolean ok = resultSet.getBoolean("OK");  *requests*.add(**new** Request(id, dateReq, dateOf, ok));  }    //STEP 6: Clean-up environment  resultSet.close();  stmt.close();  conn.close();  }  **catch** (Exception ex){  System.***out***.println(ex.getMessage());  }  }    // Демонстрація вибірки користувачі  **public** **static** **void** demonstrateSelectUser(){  Connection conn = **null**;  Statement stmt = **null**;  ResultSet resultSet = **null**;  *users* = **new** ArrayList<>();    **try** {  //STEP 2: Register JDBC driver  Class.*forName*("com.mysql.jdbc.Driver");  //STEP 3: Open a connection  conn = DriverManager.*getConnection*(SqlSettings.***DB\_URL***, SqlSettings.***USERNAME***, SqlSettings.***PASSWORD***);  //STEP 4: Execute a query  stmt = conn.createStatement();  resultSet = stmt.executeQuery(SqlQuery.***SELECT\_USER***);    //STEP 5: Extract data from result set  **while**(resultSet.next()){  **int** id = resultSet.getInt("id");  String username = resultSet.getString("username");  String password = resultSet.getString("password");  String realName = resultSet.getString("realName");  String photo = resultSet.getString("photo");  **int** accessLevel = resultSet.getInt("accessLevel");    User user = **new** User(id, username, password, realName, photo, accessLevel);  user.setLogs(*selectLogByUser*(user));    *users*.add(user);  }    //STEP 6: Clean-up environment  resultSet.close();  stmt.close();  conn.close();  }  **catch** (Exception ex){  System.***out***.println(ex.getMessage());  }  }    // Демонстрація вибірки логів по користувачу  **public** **static** List<Log> selectLogByUser(User user){  Connection conn = **null**;  PreparedStatement stmt = **null**;  ResultSet resultSet = **null**;  ArrayList<Log> logs = **new** ArrayList<>();    **try** {  //STEP 2: Register JDBC driver  Class.*forName*("com.mysql.jdbc.Driver");  //STEP 3: Open a connection  conn = DriverManager.*getConnection*(SqlSettings.***DB\_URL***, SqlSettings.***USERNAME***, SqlSettings.***PASSWORD***);  //STEP 4: Execute a query  stmt = conn.prepareStatement(SqlQuery.***SELECT\_LOG\_BY\_USER***);  stmt.setInt(1, user.getId());  resultSet = stmt.executeQuery();    //STEP 5: Extract data from result set  **while**(resultSet.next()){  **int** id = resultSet.getInt("id");  Date dateTime = resultSet.getTimestamp("DateTime");  String action = resultSet.getString("Action");  logs.add(**new** Log(id, dateTime, action, user));  }    //STEP 6: Clean-up environment  resultSet.close();  stmt.close();  conn.close();  }  **catch** (Exception ex){  System.***out***.println(ex.getMessage());  }    **return** logs;  }    //  **public** **static** **void** demonstrateInsertRequest(User user, Client client){  // Сформували нову заявку  Request req = **new** Request(**new** ~~Date~~(2017, 3, 25), **new** Date(), **false**);  req.setClient(client);  req.setUser(user);  // Потім можна додати продукцію, але в демонстрації цього не робтимемо    Connection conn = **null**;  PreparedStatement pStmt = **null**;    **try** {  //STEP 2: Register JDBC driver  Class.*forName*("com.mysql.jdbc.Driver");  //STEP 3: Open a connection  conn = DriverManager.*getConnection*(SqlSettings.***DB\_URL***, SqlSettings.***USERNAME***, SqlSettings.***PASSWORD***);  //STEP 4: Execute a query  pStmt = conn.prepareStatement(SqlQuery.***INSERT\_REQUEST***);  pStmt.setInt(1, client.getId());  pStmt.setString(2, ProgramSettings.*dateFormat*.format(req.getDateRequest()));  pStmt.setString(3, ProgramSettings.*dateFormat*.format(req.getDateOformleniya()));  pStmt.setInt(4, (req.isOk() ? 1 : 0));  pStmt.setInt(5, user.getId());  pStmt.executeUpdate();    pStmt.close();  conn.close();  }  **catch** (Exception ex){  System.***out***.println(ex.getMessage());  }    }    **public** **static** **void** demonstrateRemoveRequest(Request req){    // Спочатку необхідно видалити дані many-to-many, але цього не робитимемо в демонстраційному коді    Connection conn = **null**;  PreparedStatement pStmt = **null**;    **try** {  //STEP 2: Register JDBC driver  Class.*forName*("com.mysql.jdbc.Driver");  //STEP 3: Open a connection  conn = DriverManager.*getConnection*(SqlSettings.***DB\_URL***, SqlSettings.***USERNAME***, SqlSettings.***PASSWORD***);  //STEP 4: Execute a query  pStmt = conn.prepareStatement(SqlQuery.***DELETE\_REQUEST***);  pStmt.setInt(1, req.getId());  pStmt.executeUpdate();    pStmt.close();  conn.close();  }  **catch** (Exception ex){  System.***out***.println(ex.getMessage());  }  }    // Метод відображення списку  **public** **static** <T> **void** showList(ArrayList<T> list){  **for** (T ob : list) {  System.***out***.println(ob);  }  }  } |