

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА № 10**

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

«Створення проектів з JPA в Eclipse»

Виконав студент групи КП-61с

Литвин Вадим Олександрович

Київ 2017

Мета роботи: Створити і налаштувати JPA-проект в Eclipse і зв'язати його з обраною БД.

Тема «ІС хлібозаводу».

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

**Хід роботи**

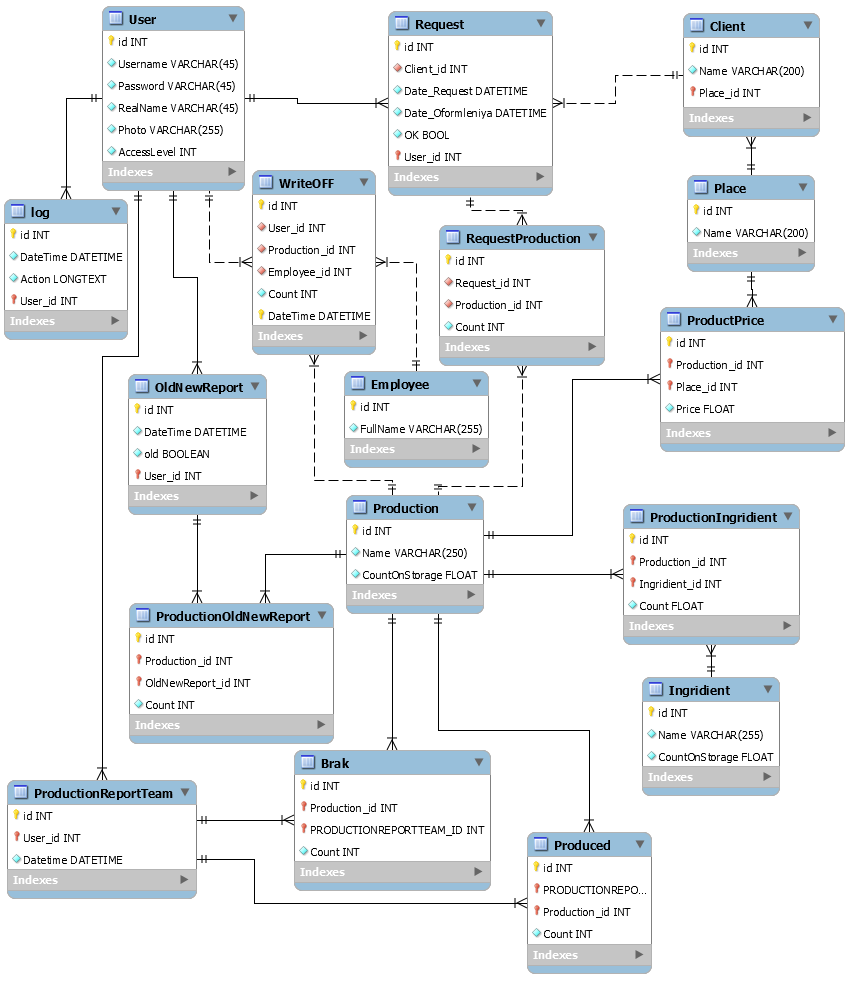


Рис. 1 Схема БД «ІС хлібозаводу»

В поточну БД було додано деякі зміни. При роботі з JPA в процесі реалізації зв’язку MANY-TO-MANY виникла проблема, що даний зв’язок реалізується на колекціях. Як типів полів і властивостей, значення яких є колекцією об'єктів, дозволено використовувати тільки наступні інтерфейси і їх параметризовані варіанти: java.util.Collection; java.util.Set; java.util.List; java.util.Map. В проекті з JDBC це було реалізовано через Dictonary, що не реалізує жодний з вище вказаних інтерфейсів. Тому зв’язки MAMY-TO-MANY реалізовано як зв’язки ONE-TO-MANY та MANY-TO-ONE. Це дещо ускладнює роботу з даними. Але на не складній БД це не дуже помітно.

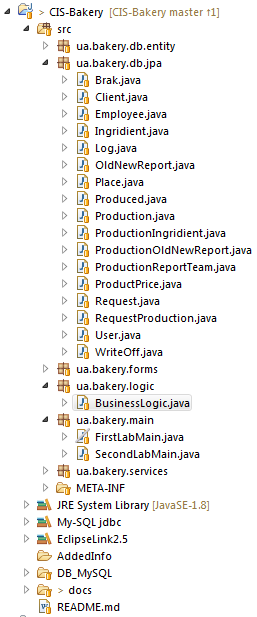


Рис. 2 Структура jdbc та JPA проекту «ІС хлібозаводу»

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. Тип проекту JPA. Роботу продовжуватимемо в проекті, який було створено в попередній лабораторній роботі.

Використовувана СУБД в даній роботі – MySQL.

На рис. 2 відображено структуру проекту даної лабораторної роботи.

Демонстрація роботи програми (Додавання нової заявки, її редагування та видалення):

\*\*\*\*\*\*\*\*\*\*\*\*СТВОРЕННЯ\*\*\*\*\*\*\*\*\*\*\*\*

[EL Info]: 2017-03-18 23:14:28.823--ServerSession(1014508942)--EclipseLink, version: Eclipse Persistence Services - 2.5.0.v20130507-3faac2b

[EL Info]: connection: 2017-03-18 23:14:29.581--ServerSession(1014508942)--file:/G:/projecs/CIS-Bakery/bin/\_CISBakeryJPA login successful

Користувач: Уманець Богдан Борисович

Клієнт : Шевченко

Замовлення:

Хліб Пшеничний (ФОРМОВИЙ) 0,5 : 82

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. : 74

Хліб Пшеничний (ФОРМОВИЙ) 0,9 : 50

Хліб Пшеничний (ФОРМОВИЙ) 0,8 : 16

Хліб арнаут КиІвський 0,6 : 41

Хліб арнаут 0,6 : 22

Хліб арнаут 0,6 різаний: 40

Хліб 0,6 НОВИЙ : 13

\*\*\*\*\*\*\*\*\*\*\*\*СТВОРЕНО\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*РЕДАГУВАННЯ\*\*\*\*\*\*\*\*\*\*\*\*

Заявка №528: Sat Mar 18 23:14:30 EET 2017: не виконано

Замовлення:

Хліб Пшеничний (ФОРМОВИЙ) 0,5 : 164

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. : 148

Хліб Пшеничний (ФОРМОВИЙ) 0,9 : 100

Хліб Пшеничний (ФОРМОВИЙ) 0,8 : 32

Хліб арнаут КиІвський 0,6 : 82

Хліб арнаут 0,6 : 44

Хліб арнаут 0,6 різаний: 80

Хліб 0,6 НОВИЙ : 26

\*\*\*\*\*\*\*\*\*\*\*\*ВІДРЕДАГОВАНО\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*ВИДАЛЕННЯ\*\*\*\*\*\*\*\*\*\*\*\*

Заявка №528: Sat Mar 18 23:14:30 EET 2017: не виконано

\*\*\*\*\*\*\*\*\*\*\*\*ВИДАЛЕНО\*\*\*\*\*\*\*\*\*\*\*\*

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

|  |
| --- |
| Лістинг 1. Клас Request.java |
| @Entity  @Table(name = "Request")  **public** **class** Request {    @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **int** id;  @Temporal(TemporalType.***TIMESTAMP***)  **private** Date date\_Request;  @Temporal(TemporalType.***TIMESTAMP***)  **private** Date date\_Oformleniya; // Дата оформлення, дати коректне ім'я  **private** **boolean** ok; // Чи виконано замовлення  @ManyToOne  **private** Client client;  @ManyToOne  **private** User user;  @OneToMany(mappedBy="Request", cascade = CascadeType.***ALL***) // PERSIST - при видаленні виникла помилка  **private** List<RequestProduction> requestProductionCount = **new** ArrayList<>();  @Override  **public** String toString(){  **return** "Заявка №" + id + ": " + date\_Oformleniya + ": " + (ok ? "виконано": "не виконано");  }  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** Date getDate\_Request() {  **return** date\_Request;  }  **public** **void** setDate\_Request(Date date\_Request) {  **this**.date\_Request = date\_Request;  }  **public** Date getDate\_Oformleniya() {  **return** date\_Oformleniya;  }  **public** **void** setDate\_Oformleniya(Date date\_Oformleniya) {  **this**.date\_Oformleniya = date\_Oformleniya;  }  **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;  }  **public** List<RequestProduction> getRequestProductionCount() {  **return** requestProductionCount;  }  **public** **void** setRequestProductionCount(List<RequestProduction> requestProductionCount) {  **this**.requestProductionCount = requestProductionCount;  }  } |

В наступних лістингах класів @Entity уникнемо наведення get & set методів.

|  |
| --- |
| Лістинг 2. User.java |
| @Entity  @Table(name="User")  **public** **class** User {    @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **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, рекомендується замінити на enum  @OneToMany(mappedBy="user")  **private** List<Log> logs = **new** ArrayList<>(); // Логи користувача  @OneToMany(mappedBy="user")  **private** List<Request> requests = **new** ArrayList<>();  @OneToMany(mappedBy="user")  **private** List<WriteOff> writeOffs = **new** ArrayList<>();  @OneToMany(mappedBy="user")  **private** List<OldNewReport> OldNewReports = **new** ArrayList<>();    @OneToMany(mappedBy="user")  **private** List<ProductionReportTeam> productionReportTeams = **new** ArrayList<>();    @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()\*/ **this**.realName;  }    } |

|  |
| --- |
| Лістинг 3. Production.java |
| @Entity  @Table(name = "Production")  **public** **class** Production {  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **int** id;  **private** String name;  **private** **float** countOnStorage; // Кількість наявного на складі  @OneToMany(mappedBy="Production")  **private** List<ProductPrice> productionPrice = **new** ArrayList<>();    @OneToMany(mappedBy="Production")  **private** List<ProductionIngridient> productionIngridient = **new** ArrayList<>();    @OneToMany(mappedBy="Production")  **private** List<RequestProduction> requestProductionCount = **new** ArrayList<>();  @OneToMany(mappedBy="Production")  **private** List<WriteOff> writeOffs = **new** ArrayList<>();    @OneToMany(mappedBy="Production")  **private** List<ProductionOldNewReport> productionOldNewReports = **new** ArrayList<>();    @OneToMany(mappedBy="Production")  **private** List<Brak> braks = **new** ArrayList<>();    @OneToMany(mappedBy="Production")  **private** List<Produced> produceds = **new** ArrayList<>();    } |

|  |
| --- |
| Лістинг 4. BusinessLogic.java |
| **package** ua.bakery.logic;  **import** java.util.ArrayList;  **import** java.util.Date;  **import** java.util.List;  **import** java.util.Random;  **import** javax.persistence.EntityManager;  **import** javax.persistence.EntityManagerFactory;  **import** javax.persistence.Persistence;  **import** javax.persistence.Query;  **import** ua.bakery.db.jpa.\*;  **public** **final** **class** BusinessLogic {    **private** BusinessLogic() { }    **private** **static** **final** String ***PERSISTENCE\_UNIT\_NAME*** = "CISBakeryJPA";  **private** **static** EntityManagerFactory *factory*;    **public** **static** **int** AddNewRequest(){  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*СТВОРЕННЯ\*\*\*\*\*\*\*\*\*\*\*\*");  // Для генераціїї випадкових кількостей продукції  Random random = **new** Random();  *factory* = Persistence.*createEntityManagerFactory*(***PERSISTENCE\_UNIT\_NAME***);  EntityManager em = *factory*.createEntityManager();    // Отримаємо існуючого користувача і клієнта  User user = (User) em.find(User.**class**, 2);  Client client = (Client) em.find(Client.**class**, 4);  System.***out***.println("Користувач: " + user.toString());  System.***out***.println("Клієнт : " + client.toString());    // Отримаємо всі продукти  Query q = em.createQuery("select p from Production p");  @SuppressWarnings("unchecked")  List<Production> productions = q.getResultList();  // Створимо замовлення  Request request = **new** Request();  request.setDate\_Request(**new** Date()); // Встановити дату виконання  request.setDate\_Oformleniya(**new** Date()); // Встановити дату формування  request.setClient(client);  request.setUser(user);  request.setOk(**false**);    // Сформуємо список необхідних продуктів    // Штучно обмежимо максимальну кількість продукції в замовленні - ЛИШЕ ДЛЯ ЗРУЧНОЇ ДЕМОНСТРАЦІЇ  **int** i = 0, MAX = 7;    System.***out***.println("Замовлення: ");  List<RequestProduction> requestProductions = **new** ArrayList<>();  **for**(Production production : productions){    **int** count = random.nextInt(100);    **if**(count != 0){  RequestProduction rp = **new** RequestProduction();  rp.setProduction(production);  rp.setRequest(request);  rp.setCount(count);  requestProductions.add(rp);  System.***out***.println(production.getName() + ": " + count);    **if**(++i>MAX) **break**;  }  }    request.setRequestProductionCount(requestProductions);    // Збережемо до БД  em.getTransaction().begin();  em.persist(request);  em.getTransaction().commit();  em.close();  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*СТВОРЕНО\*\*\*\*\*\*\*\*\*\*\*\*");    **return** request.getId();  }    // Помножити кількість продукції на два  **public** **static** **void** EditRequest(**int** id){  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*РЕДАГУВАННЯ\*\*\*\*\*\*\*\*\*\*\*\*");  *factory* = Persistence.*createEntityManagerFactory*(***PERSISTENCE\_UNIT\_NAME***);  EntityManager em = *factory*.createEntityManager();    // Отримаємо замовлення  Request request = (Request) em.find(Request.**class**, id);    // Переглянемо замовлення  System.***out***.println(request);    // Змінимо замовлення  System.***out***.println("Замовлення: ");  em.getTransaction().begin();  **for**(RequestProduction requestProduction : request.getRequestProductionCount()){  requestProduction.setCount(requestProduction.getCount() \* 2);  System.***out***.println(requestProduction.getProduction().getName() + ": " + requestProduction.getCount());  }  em.getTransaction().commit();  em.close();  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*ВІДРЕДАГОВАНО\*\*\*\*\*\*\*\*\*\*\*\*");  }    **public** **static** **void** DeleteRequest(**int** id){  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*ВИДАЛЕННЯ\*\*\*\*\*\*\*\*\*\*\*\*");  *factory* = Persistence.*createEntityManagerFactory*(***PERSISTENCE\_UNIT\_NAME***);  EntityManager em = *factory*.createEntityManager();    // Отримаємо замовлення  Request request = (Request) em.find(Request.**class**, id);    // Переглянемо замовлення  System.***out***.println(request);    // Видалемо замовлення  em.getTransaction().begin();  em.remove(request);  em.getTransaction().commit();  em.close();  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*ВИДАЛЕНО\*\*\*\*\*\*\*\*\*\*\*\*");  }    } |

|  |
| --- |
| Лістинг 5. Main.java |
| **package** ua.bakery.main;  **import** ua.bakery.logic.BusinessLogic;  **public** **class** SecondLabMain {    **public** **static** **void** main(String[] args) {  **int** requestId = BusinessLogic.*AddNewRequest*();  BusinessLogic.*EditRequest*(requestId);  BusinessLogic.*DeleteRequest*(requestId);  }    } |