НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА

|  |
| --- |
| з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»  на тему: «Робоче місце медичного працівника призивного відділення військового комісаріату» |

|  |  |  |
| --- | --- | --- |
|  | Студента 2 курсу групи ІО-32  напряму підготовки  6.050102 «Комп’ютерна інженерія»  Нетудихати Антона Геннадійовича  Керівник  Болдак Андрій Олександрович  (прізвище та ініціали)  Доцент кафедри ОТ  (посада, вчене звання, науковий ступінь)  Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |  |
|  |  |

|  |
| --- |
| м. Київ - 2015 рік |

# ЗМІСТ

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

2

*6.050102 «Комп’ютерна інженерія»*

Розроб.

Нетудихата

Перевір.

Болдак

Реценз.

Н. Контр.

Затверд.

Болдак

Робоче місце медичного працівника призивного відділення військового комісаріату

Літ.

Акрушів

33

*Кафедра Обчислювальної техніки*

[ЗМІСТ 2](#_Toc415604448)

[РОЗДІЛ 1 3](#_Toc415604449)

[Запити зацікавлених осіб 3](#_Toc415604450)

[1.1 Введення 3](#_Toc415604451)

[1.2 Короткий огляд продукту 3](#_Toc415604452)

[1.3 Ділові правила та приписи 4](#_Toc415604453)

[1.4 Функціональність 5](#_Toc415604454)

[1.5 Практичність 6](#_Toc415604455)

[1.6 Надійність 6](#_Toc415604456)

[РОЗДІЛ 2 7](#_Toc415604457)

[РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 7](#_Toc415604458)

[2.1 Загальна схема прецедентів 7](#_Toc415604460)

[2.2 Прецеденти для ролі медичного працівника 7](#_Toc415604461)

[2.3 Діаграма бізнес-сутностей 9](#_Toc415604462)

[2.4 Реляційна модель бази даних 9](#_Toc415604463)

[2.5 Специфікація таблиць бази даних 10](#_Toc415604464)

[РОЗДІЛ 3 13](#_Toc415604465)

[РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 13](#_Toc415604466)

[3.1 Реляційно-об’єктне відображення 13](#_Toc415604468)

[3.2 Специфікація HibernateUntil класу 17](#_Toc415604469)

[3.3 Класи контролерів та їх специфікація 18](#_Toc415604470)

[3.4 Hibernate mapping файли 20](#_Toc415604471)

[РОЗДІЛ 4 21](#_Toc415604472)

[ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ 21](#_Toc415604473)

[4.1 Взаємодія медичного працівника і системи при прийомі призовника](#_Toc415604475)

[СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 24](#_Toc415604476)

[ДОДАТКИ 25](#_Toc415604477)

[ДОДАТОК А 25](#_Toc415604478)

# РОЗДІЛ 1

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

6.050102 «Комп’ютерна інженерія»

# Запити зацікавлених осіб

## Введення

У цьому документі описуються запити зацікавлених осіб, в якості яких виступає замовник – *збройні сили України*, по відношенню до системи призовного відділення військового комісаріату "Military Control System".

### 1.1.1 Мета

Метою документа є визначення основних потреб щодо функціональності та експлуатаційної придатності, а також визначення бізнес-правил та технологічних обмежень стосовно предмета розробки.

### 1.1.2 Контекст

Перелік потреб, що перераховані в даному документі, є основою технічного завдання на розробку робочого місця медичного працівника сервісу "Military Control System".

## Короткий огляд продукту

Сервіс являє собою систему, за допомогою якої медичний спеціаліст може знайти потрібного призовника із бази, переглянути його профіль, медичну картку. Також, маючи свій унікальний пароль, внести до медичної картки призовника запис, який буде свідчити про стан здоров’я призовника. Система, за введеним паролем, автоматично ідентифікує лікаря і вводить до бази даних потрібну інформацію.

## Ділові правила та приписи

### 1.3.1 Призначення системи "Military Control System".

Функціональність веб-сервісу призначена для інформатизації процесу роботи призовного відділення військового комісаріату, а саме робочого місця медичних спеціалістів. Система допомагає зручно переглядати та редагувати інформацію про призовників.

### 1.3.2 Політика взаємовідносин з клієнтом

Клієнтами системи *«Military Control System»* є медичні працівники військового комісаріату.

### 1.3.3 Характеристика ділового процесу

Медичний працівник може переглядати список призовників, відфільтровувати цей список, за допомогою функції динамічного пошуку, що дозволяє швидко знаходити профіль призовника. Медичний працівник переглядає профіль і вносить свій діагноз, за допомогою свого пароля, також при необхідності може переглядати медичну картку.

Користувач системи має доступ до бази даних.

### 1.3.4 Сценарій занесення діагнозу призовника до бази

1. Медичний працівник знаходить призовника у списку за допомогою пошуку;

2. Система шукає серед всіх доступних призовників і відбирає тих, що відповідають параметрам пошуку;

3. Система повертає працівнику посилання на профіль шуканого призовника;

4. Працівник переходить на профіль призовника ;

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

6.050102 «Комп’ютерна інженерія»

5. Система повертає сторінку із інформацією про призовника;

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

5

6.050102 «Комп’ютерна інженерія»

6.Працівник вводить у спеціальне поле діагноз;

7. Працівник вводить пароль підтвердження операції і натискає кнопку «Надіслати»;

8. Система перевіряє чи є заданий пароль у базі даних;

9. Система заносить до бази діагноз;

10. Працівник отримує повідомлення на сторінці про успішно завершену операцію.

### 1.3.5 Сценарій перегляду медичної картки призовника

1. Медичний працівник знаходить призовника у списку за допомогою пошуку;

2. Система шукає серед всіх доступних призовників і відбирає тих, що відповідають параметрам пошуку;

3. Система повертає працівнику посилання на профіль шуканого призовника.

4. Працівник переходить на профіль призовника;

5. Система повертає сторінку із інформацією про призовника;

6. Користувач тисне кнопку «+»;

7. Система відображає медичну картку призовника.

## Функціональність

Основні потреби щодо функціональності, що пред’являються зацікавленим особам до предмету розробки, відносяться до категорій:

- Користувач системи;

- Медичний працівник.

### 1.4.1 Можливості медичного працівника

- Перегляд списку призовників;

- Динамічне фільтрування списку;

- Перегляд профілю призовників;

- Внесення діагнозів до бази даних ;

- Перегляд медичних карток призовників .

## Практичність

### 1.5.1 Універсальність

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

6

6.050102 «Комп’ютерна інженерія»

Сервіс може використовуватися у будь-якому військовому комісаріаті України.

## Надійність

### 1.6.1 Захист інформації користувача

Сервіс повинен надавати надійні засоби захисту інформації про користувачів, включаючи особисту інформацію та платіжні реквізити. Вся інформація повинна бути надійно зашифрована та максимально захищена.

### 1.6.2 Резервне копіювання

Повинно здійснюватись резервне копіювання баз даних, з можливістю їх подальшого відновлення.

# РОЗДІЛ 2

# РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ



## Загальна схема прецедентів

Загальна схема прецедентів для ролі користувача показує можливі послідовності дій користувачів системи. Основним видом діяльності користувача , а саме медичного працівника, є перегляд профілю призовника, його медичної картки і внесення змін до відповідної графи у картці. Схема прецедентів представлена на рис. 2.1.

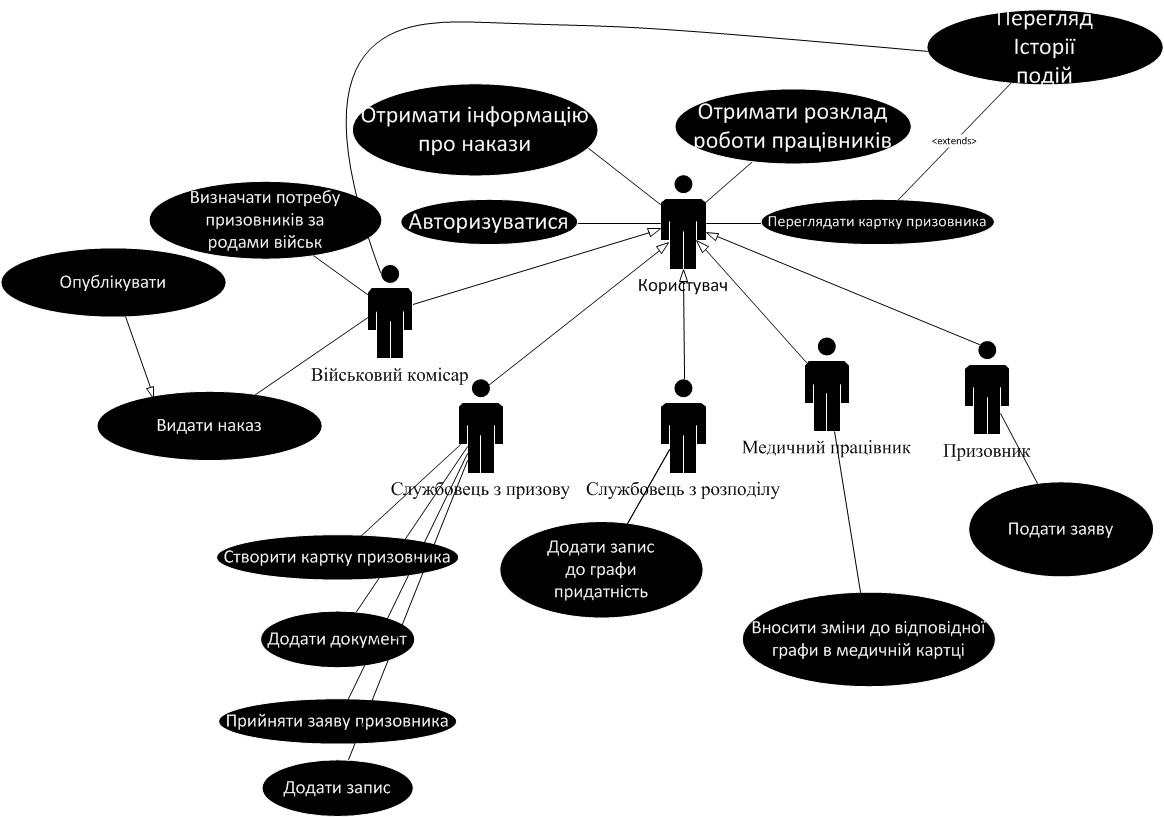


Рис. 2.1 – Загальна схема прецедентів для ролі користувача

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

7

6.050102 «Комп’ютерна інженерія»

## Прецеденти для ролі медичного працівника

Нижче описаний прецедент для ролі медичного працівника з вказаними передумовами, результатом, виключними ситуаціями та детальним описом послідовності дій.

Таблиця 2.1

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

8

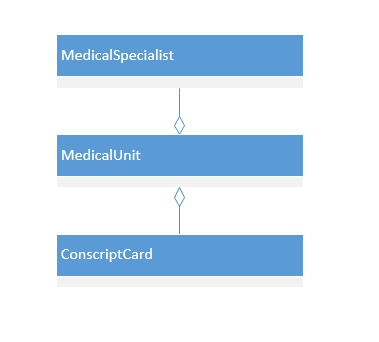
6.050102 «Комп’ютерна інженерія»

Прецедент №1. Перегляд маршрутів на карті

|  |  |  |
| --- | --- | --- |
| ID | 1 | |
| Назва | Проходження призовником медкомісії | |
| Учасники | Медичні працівники, система | |
| Передумова | Призовник зареєстрований в системі. Медичний працівник має свій унікальний пароль | |
| Результат | Висновок про придатність або непридатність призовника до служби внесений до медичної картки призовника | |
| Основний  сценарій | Дії медичного спеціаліста | Дії системи |
|  | 1. Знаходить у списку потрібного призовника та обирає його |  |
|  |  | 1. Повертає сторінку із профілем призовника і полем введення діагнозу |
|  | 1. Додає запис до медичної картки і вводить власний унікальний пароль, натискає кнопку «Надіслати» |  |
|  |  | 1. Звіряє пароль із базою даних. |
|  |  | 1. Ідентифікує лікаря за введеним паролем. |
|  |  | 1. Заносить коментар до бази даних. |
| Виключні  ситуації | 1. Призовник потребує додаткового обстеження на стаціонарі | |
|  | 1. Медичний працівник ввів невірний пароль | |

## Діаграма бізнес-сутностей

Дана діаграма створюється на етапі бізнес моделювання. Вона відображає основні сутності та взаємозв’язки між ними. В даному випадку основними сутностями є *ConscriptCard, Medicalspecialist, MedicalUnit,* які взаємодіють між собою та включають у себе допоміжні бізнес-сутності. Діаграма бізнес-сутностей проекту зображена на рис 2.2.



Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

9

6.050102 «Комп’ютерна інженерія»

Рис 2.2 – Діаграма бізнес-сутностей

## Реляційна модель бази даних

Реляційна модель бази даних (рис. 2.3) зображує структуру таблиць бази даних, взаємозв’язки між ними та поля кожної з таблиць. Наведена діаграма має багато схожого з діаграмою бізнес-сутностей. Кожній основній бізнес-сутності відповідає таблиця баз даних.

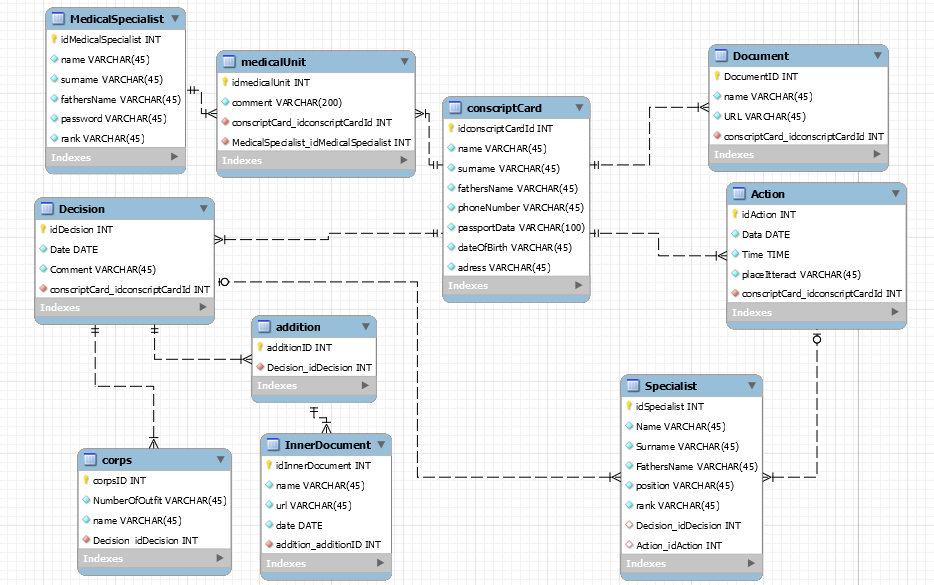


Рис. 2.3 – Реляційна модель бази даних

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

10

6.050102 «Комп’ютерна інженерія»

## Специфікація таблиць бази даних

Специфікація таблиць бази даних включає в себе інформацію про назви колонок таблиці, їхній тип, інформацію про те чи є ця колонка первинним ключем, чи поле може бути пустим, чи значення поля автоматично збільшується та коментар щодо призначення колонки. Таблиці з специфікаціями наведені нижче.

Таблиця 2.2

Таблиця «ConscriptCard»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Не пуста | Авто-інкремент | Ключ | Опис |
| ID | INT | Так | Так | Так | Ідентифікатор користувача |
| name | VARCHAR | Так | Ні | Ні | Ім’я призовника |
| surname | VARCHAR | Так | Ні | Ні | Прізвище |
| fathersName | VARCHAR | Так | Ні | Ні | По-батькові |
| phoneNomber | VARCHAR  Змн.  Арк.  № докум.  Підпис  Дата  Арк.  11  6.050102 «Комп’ютерна інженерія» | Так | Ні | Ні | Номер мобільного телефону |
| passportData | VARCHAR | Так | Ні | Ні | Паспортні данні |
| dateOfBirth | VARCHAR | Так | Ні | Ні | Дата народження |
| adress | VARCHAR | Так | Ні | Ні | Адрес призовника |

Таблиця 2.3

Таблиця «Medicalspecialist»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Не пуста | Авто-інкремент | Ключ | Опис |
| ID | INT | Так | Так | Так | Ідентифікатор міста |
| name | VARCHAR | Так | Ні | Ні | Ім’я спеціаліста |
| surname | VARCHAR | Так | Ні | Ні | Прізвище |
| fathersName | VARCHAR | Так | Ні | Ні | По-батькові |
| rank | VARCHAR | Так | Ні | Ні | Посада |
| password | VARCHAR | Так | Ні | Ні | Пароль мед. спеціаліста |

Таблиця 2.4

Таблиця «MedicalUnit»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Не пуста | Авто-інкремент | Ключ | Опис |
| ID | INT | Так | Так | Так | Ідентифікатор маршруту |
| comment | VARCHAR | Так | Ні | Ні | Коментар медичного спеціаліста |
| idConscriprCard | INT | Так | Ні | Так | Посилання на профіль призовника  Змн.  Арк.  № докум.  Підпис  Дата  Арк.  12  6.050102 «Комп’ютерна інженерія» |
| idSpecialist | INT | Так | Ні | Так | Посилання на медичного спеціаліста |

# РОЗДІЛ 3

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

13

6.050102 «Комп’ютерна інженерія»

# РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ



## Реляційно-об’єктне відображення

Для реляційно-об’єктного відображення в програмі використовується шаблон проектування DAO, який побудований з використанням бібліотеки Hibernate. Hibernate надає можливість легко встановити зв’язок з будь-якою базою даних та створити відображення між об’єктно-орієнтованою моделлю та традиційною реляційною моделлю баз даних. На рис. 3.1 зображено діаграму Entity класів. Детальна специфікація (JavaDoc) наведена нижче.

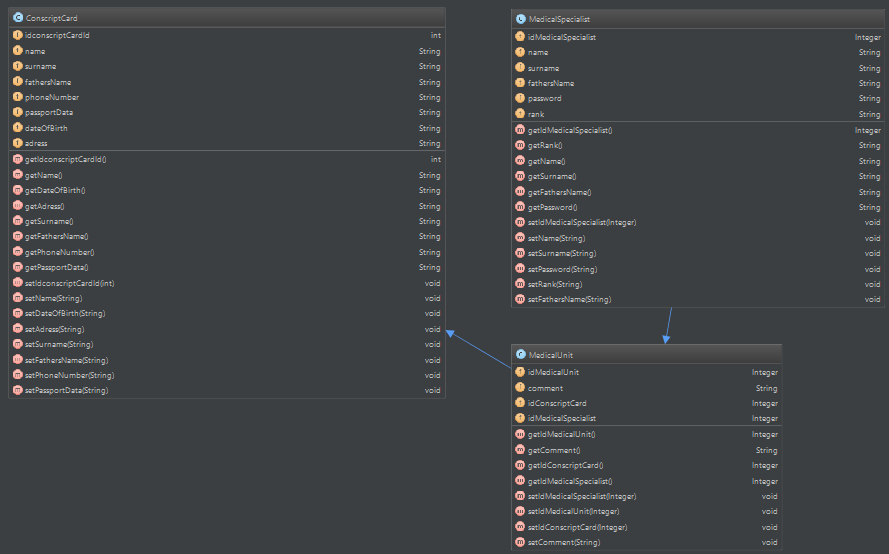


Рис 3.1 – Реляційно-об’єктне відображення

### 3.1.2 Клас «ConscriptCardDAO»

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

14

6.050102 «Комп’ютерна інженерія»

### 

### 3.1.3 Клас «MedicalSpecialistDAO»

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

15

6.050102 «Комп’ютерна інженерія»

### 

### 3.1.4 Клас «MedicalUnitDAO»



Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

16

6.05012 «Комп’ютерна інженерія»

## Специфікація HibernateUntil класу

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

17

6.050102 «Комп’ютерна інженерія»

Клас HibernateUtil слугує для того, щоб при взаємодії із файлами конфігурації створювати hibernate-сесію. Діаграму цього інтерфейса можна побачити на рис. 3.2. Детальна специфікація (JavaDoc) наведена нижче.

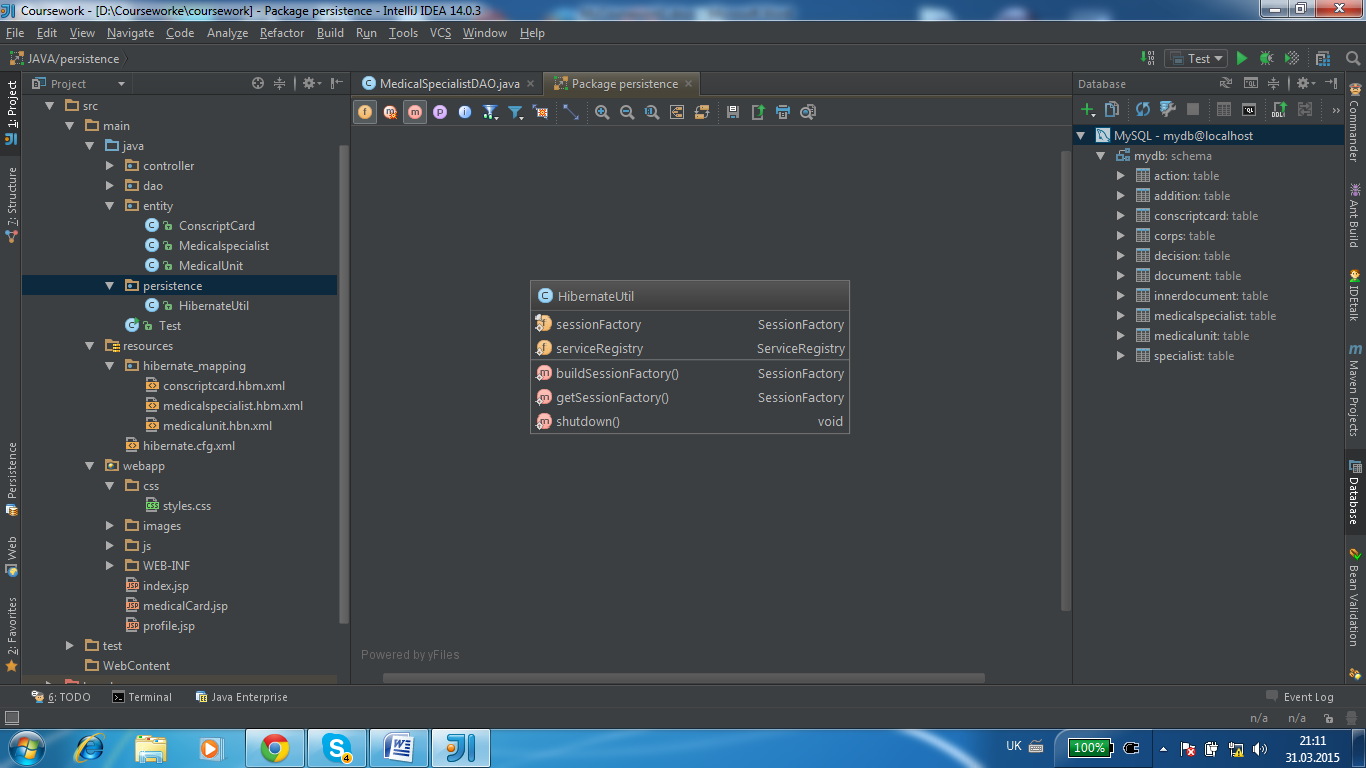


Рис 3.2. – Діаграма HibernateUnit класу

### Клас «HibernateUntil»

### 

## Класи контролерів та їх специфікація

Дані класи призначені для створення зв’язку між сервером та клієнтом. Діаграма класів контролерів представлена на рис 3.3.

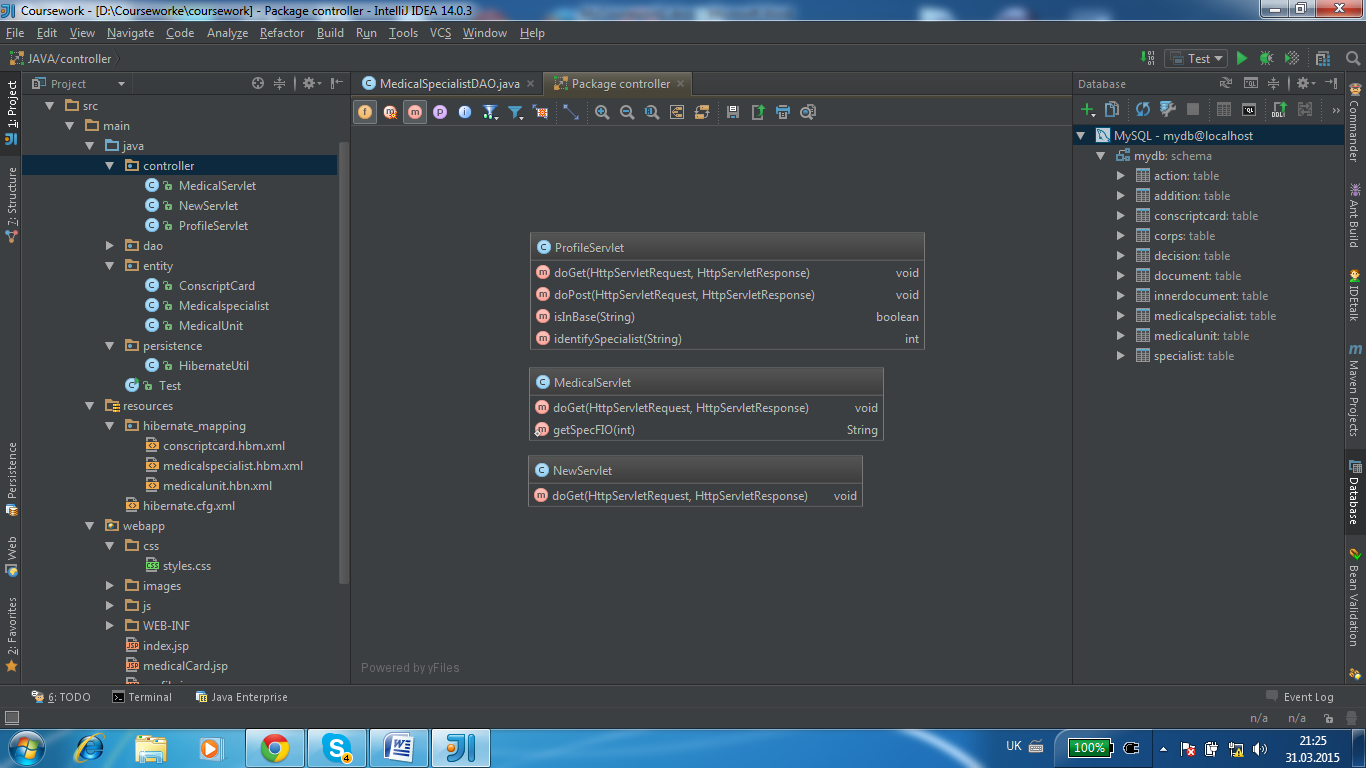


Рис 3.3 – Діаграма контролерів

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

18

6.050102 «Комп’ютерна інженерія»

### 3.3.2 Клас «MedicalServlet»

Змн.

Арк.

№ докум.

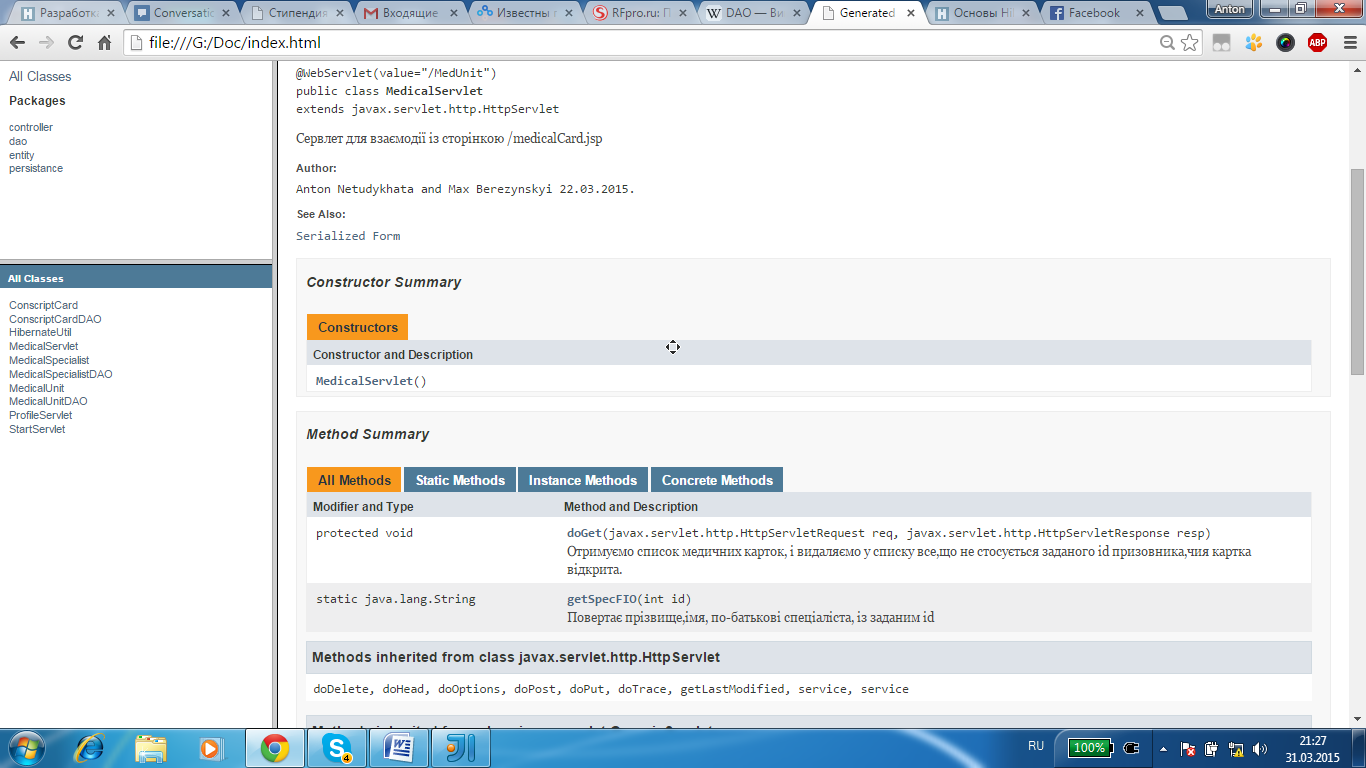
Підпис

Дата

Арк.

19

6.050102 «Комп’ютерна інженерія»



### 

## Hibernate mapping файли

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

20

6.050102 «Комп’ютерна інженерія»

Hibernate mapping файли відповідають за взаємодію наших Entity об’єктів з Hibernate і з базою данних.

Список XML файлів наведений на рис 3.4

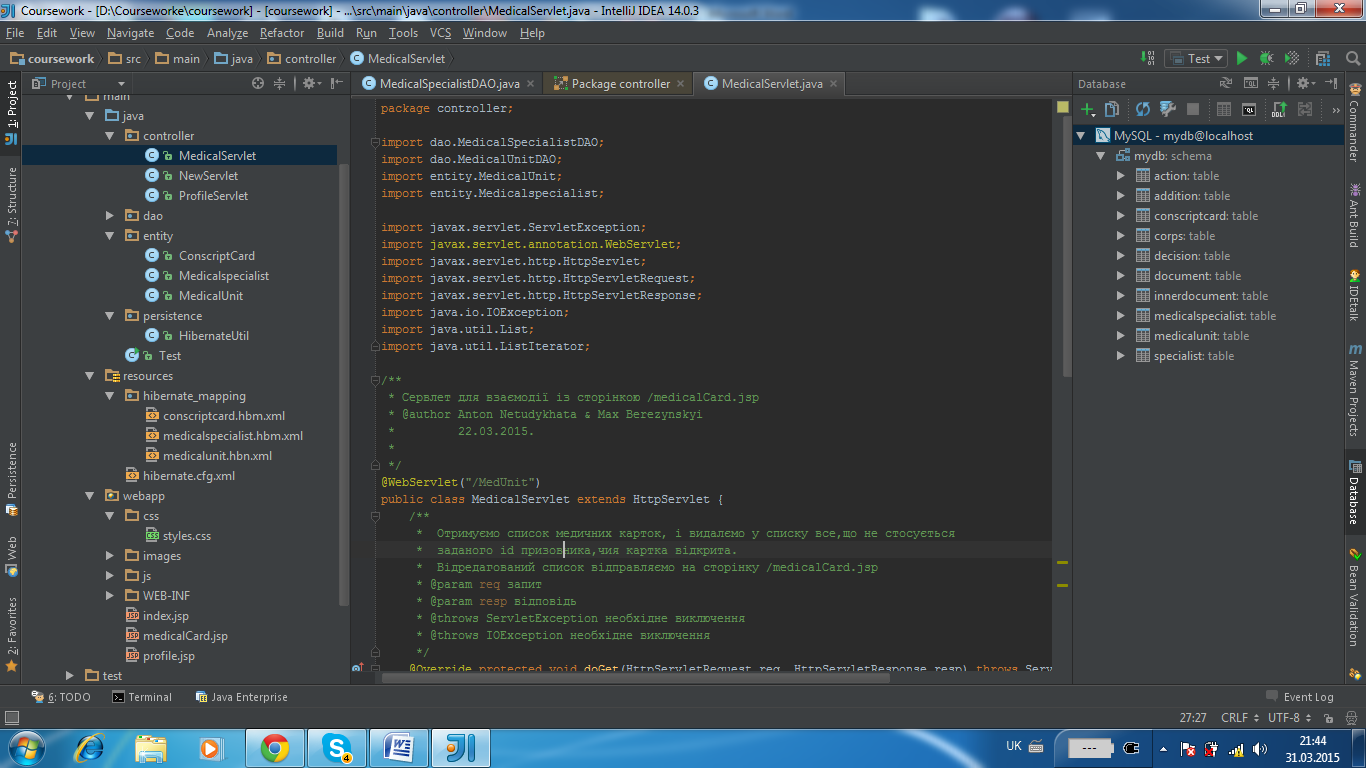


Рис 3.4 – Список XML файлів

# РОЗДІЛ 4

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

21

6.050102 «Комп’ютерна інженерія»

# ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ

Для ілюстрації роботи програми в цьому розділі наведено графічні сценарії роботи проекту.



## Взаємодія медичного працівника і системи при прийомі призовника

* + 1. Головна сторінка сайту відображає медичному спеціалісту список всіх призовників(рис 4.1).

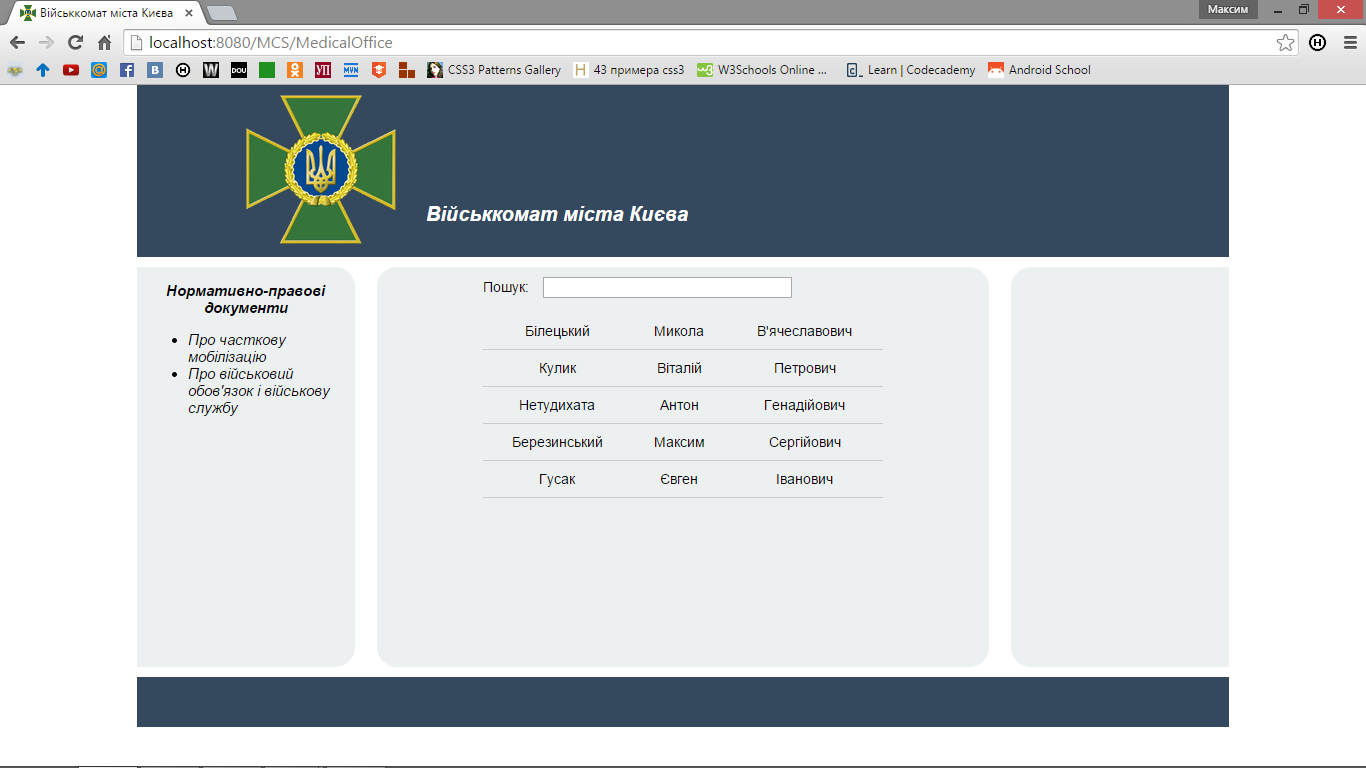
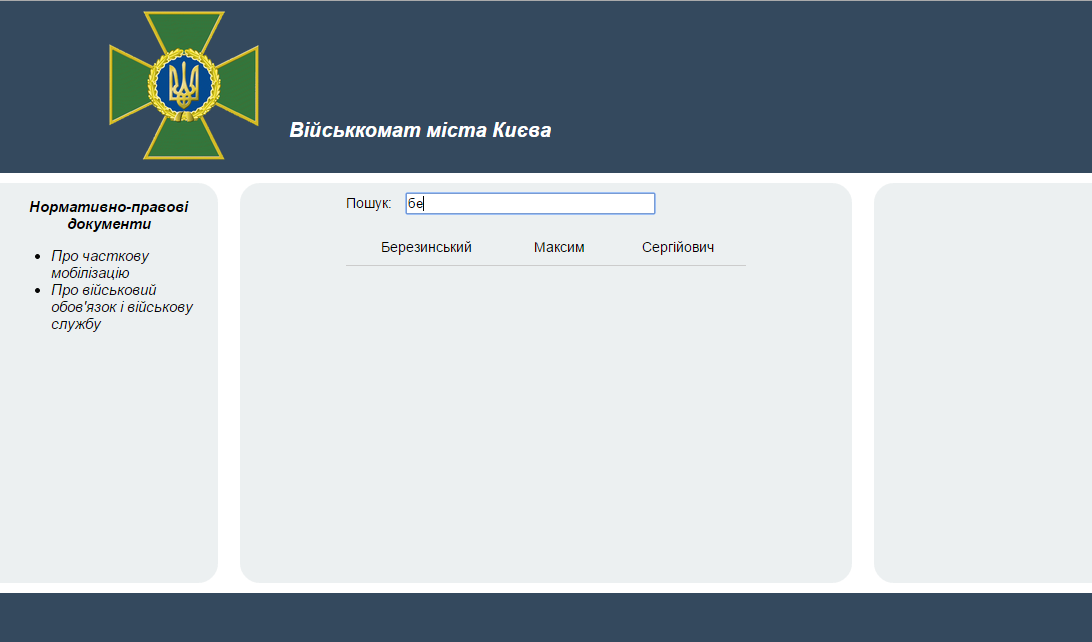


Рис 4.1 – Головна сторінка сайту

* + 1. Реалізація функції пошуку.  
         
        Рис 4.2 –Реалізація функції пошуку

Змн.

Арк.

№ докум.

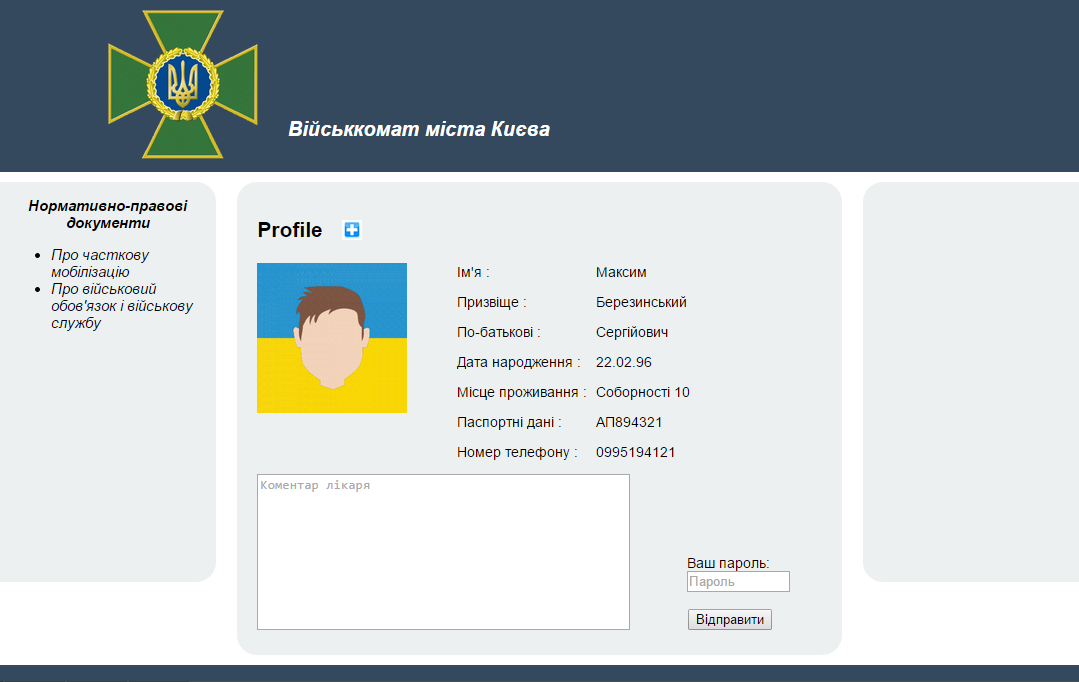
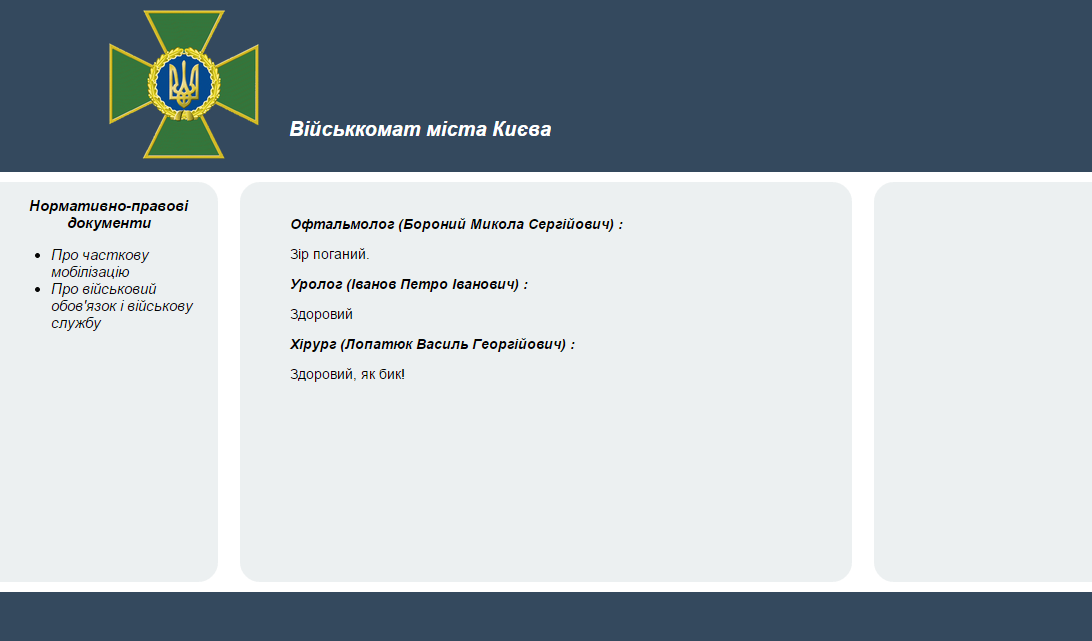
Підпис

Дата

Арк.

22

6.050102 «Комп’ютерна інженерія»

* + 1. Відображення профілю призовника  
         
        Рис 4.3 –Профіль призовника
    2. Відображення медичної картки призовника  
         
        Рис 4.4 –Медична картка призовника

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

23

6.050102 «Комп’ютерна інженерія»

# СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

24

6.050102 «Комп’ютерна інженерія»

* + - 1. DAO. – Посилання: <https://ru.wikipedia.org/wiki/Data_Access_Object>
      2. Apache Tomcat. – Посилання: <http://tomcat.apache.org/>
      3. Hibernate. Everything data. – Посилання: <http://hibernate.org>
      4. Maven. – Посилання: <http://maven.apache.org>
      5. Java Servlet. –Посилання: [https://ru.wikipedia.org/wiki/Сервлет\_(Java)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82_(Java))

# ДОДАТКИ

# ДОДАТОК А

### СonscriptCardDAO.java

public class ConscriptCardDAO {

public void addToDB(ConscriptCard card) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

session.beginTransaction();

session.save(card);

session.getTransaction().commit();

} catch (Exception ex) {

ex.printStackTrace();

} finally {

if (session != null) {

session.close();

}

}

}

public void deleteFromDB(Integer id) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

session.beginTransaction();

ConscriptCard itemDelete = (ConscriptCard) session.get(ConscriptCard.class, id);

session.delete(itemDelete);

session.getTransaction().commit();

} catch (Exception ex) {

ex.printStackTrace();

} finally {

if (session != null) {

session.close();

}

}

}

public ConscriptCard getConscriptCardById(int id) {

SessionFactory factory = HibernateUtil.getSessionFactory();

ConscriptCard unit = null;

Session session = factory.openSession();

Transaction ta = null;

try {

ta = session.beginTransaction();

Criteria cr = session.createCriteria(ConscriptCard.class);

Criterion idCr = Restrictions.like("id", id);

cr.add(idCr);

unit = (ConscriptCard) cr.uniqueResult();

} catch (HibernateException e) {

e.printStackTrace();

} finally {

}

return unit;

}

public List<ConscriptCard> getCards() {

List<ConscriptCard> result = null;

Session session = HibernateUtil.getSessionFactory().openSession();

try {

session.beginTransaction();

Criteria criteria = session.createCriteria(ConscriptCard.class);

result = (List<ConscriptCard>) criteria.list();

session.getTransaction().commit();

} catch (Exception e) {

}finally {

if (session != null) session.close();

}

return result;

}

}

### MedicalSpecialistDAO.java

public class MedicalSpecialistDAO {  
  
 public void addToDB(Medicalspecialist medspec) {  
 Session session = HibernateUtil.*getSessionFactory*().openSession();  
 try {  
 session.beginTransaction();  
 session.save(medspec);  
 session.getTransaction().commit();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
 }  
 public void deleteFromDB(Integer id) {  
 Session session = HibernateUtil.*getSessionFactory*().openSession();  
 try {  
 session.beginTransaction();  
 Medicalspecialist itemDelete = (Medicalspecialist) session.get(Medicalspecialist.class, id);  
 session.delete(itemDelete);  
  
 session.getTransaction().commit();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
 }  
  
 public Medicalspecialist getMedicalSpecialistById(int id) {  
 SessionFactory factory = HibernateUtil.*getSessionFactory*();  
 Medicalspecialist specialist = null;  
  
 Session session = factory.openSession();  
 Transaction ta = null;  
 try {  
 ta = session.beginTransaction();  
 Criteria cr = session.createCriteria(Medicalspecialist.class);  
 Criterion idCr = Restrictions.*like*("id", id);  
 cr.add(idCr);  
 specialist = (Medicalspecialist) cr.uniqueResult();  
 } catch (HibernateException e) {  
 e.printStackTrace();  
 } finally {  
  
 }  
 return specialist;  
 }  
  
  
 public List<Medicalspecialist> getSpecialists() {  
  
 List<Medicalspecialist> result = null;  
 Session session = HibernateUtil.*getSessionFactory*().openSession();  
 try {  
 session.beginTransaction();  
 Criteria criteria = session.createCriteria(Medicalspecialist.class);  
 result = (List<Medicalspecialist>) criteria.list();  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }finally {  
 if (session != null) session.close();  
 }  
 return result;  
 }  
}

### MedicalSpecialistDAO.java

public class MedicalUnitDAO {  
  
 public void addToDB(MedicalUnit unit) {  
 Session session = HibernateUtil.*getSessionFactory*().openSession();  
 try {  
 session.beginTransaction();  
 session.save(unit);  
 session.getTransaction().commit();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
 }  
  
 public void deleteFromDB(Integer id) {  
 Session session = HibernateUtil.*getSessionFactory*().openSession();  
 try {  
 session.beginTransaction();  
 MedicalUnit itemDelete = (MedicalUnit) session.get(MedicalUnit.class, id);  
 session.delete(itemDelete);  
  
 session.getTransaction().commit();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
 }  
  
 public MedicalUnit getMedicalUnitById(int id) {  
 SessionFactory factory = HibernateUtil.*getSessionFactory*();  
 MedicalUnit unit = null;  
  
 Session session = factory.openSession();  
 Transaction ta = null;  
 try {  
 ta = session.beginTransaction();  
 Criteria cr = session.createCriteria(Medicalspecialist.class);  
 Criterion idCr = Restrictions.*like*("id", id);  
 cr.add(idCr);  
 unit = (MedicalUnit) cr.uniqueResult();  
 } catch (HibernateException e) {  
 e.printStackTrace();  
 } finally {  
 }  
 return unit;  
 }  
 public List<MedicalUnit> getUnits() {  
  
 List<MedicalUnit> result = null;  
 Session session = HibernateUtil.*getSessionFactory*().openSession();  
 try {  
 session.beginTransaction();  
 Criteria criteria = session.createCriteria(MedicalUnit.class);  
 result = (List<MedicalUnit>) criteria.list();  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }finally {  
 if (session != null) session.close();  
 }  
 return result;  
 }  
}

### HibernateUtil.java

public class HibernateUtil {  
 private static final SessionFactory *sessionFactory* = *buildSessionFactory*();  
 private static ServiceRegistry *serviceRegistry*;  
  
 private static SessionFactory buildSessionFactory() {  
 try {  
 // Создает сессию с hibernate\_mapping.cfg.xml  
 Configuration configuration = new Configuration();  
 configuration.configure();  
 *serviceRegistry* = new ServiceRegistryBuilder().applySettings(configuration.getProperties()).buildServiceRegistry();  
  
 return configuration.buildSessionFactory(*serviceRegistry*);  
 }  
 catch (Throwable ex) {  
 System.*err*.println("Initial SessionFactory creation failed." + ex);  
 throw new ExceptionInInitializerError(ex);  
 }  
 }  
  
 public static SessionFactory getSessionFactory() {  
 return *sessionFactory*;  
 }  
  
 public static void shutdown() {  
 *getSessionFactory*().close();  
 }  
  
}

### HibernateUtil.java

@WebServlet("/MedUnit")  
public class MedicalServlet extends HttpServlet {  
 @Override protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
  
 String id = req.getParameter("id");  
 int idInt = Integer.*parseInt*(id);  
  
 MedicalUnitDAO dao = new MedicalUnitDAO();  
 List<MedicalUnit> list = dao.getUnits();  
 for (ListIterator< MedicalUnit> i = list.listIterator(); i.hasNext(); ) {  
 MedicalUnit el = i.next();  
 if(el.getIdConscriptCard() != idInt) {  
 i.remove();  
 }  
 }  
  
 req.setAttribute("medUnits",list);  
 req.getRequestDispatcher("/medicalCard.jsp").forward(req, resp);  
  
 }  
  
 public static String getSpecFIO(int id) {  
 MedicalSpecialistDAO dao = new MedicalSpecialistDAO();  
 Medicalspecialist spec = dao.getMedicalSpecialistById(id);  
 String fio = spec.getName()+" "+spec.getSurname()+" "+spec.getFathersName();  
 return fio;  
 }  
}

### conscriptcard.hbm.xml

<?xml version="1.0"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
  
<hibernate-mapping>  
 <class name="entity.ConscriptCard" table="conscriptcard" catalog="mydb">  
 <id name="idconscriptCardId" type="java.lang.Integer">  
 <column name="idconscriptCardId" />  
 <generator class="identity" />  
 </id>  
 <property name="name" type="string">  
 <column name="name" length="45" not-null="true" />  
 </property>  
 <property name="surname" type="string">  
 <column name="surname" length="45" not-null="true" />  
 </property>  
 <property name="fathersName" type="string">  
 <column name="fathersName" length="45" not-null="true" />  
 </property>  
 <property name="phoneNumber" type="string">  
 <column name="phoneNumber" length="45" not-null="true" />  
 </property>  
 <property name="passportData" type="string">  
 <column name="passportData" length="45" not-null="true" />  
 </property>  
  
 </class>  
</hibernate-mapping>

### medicalspecialist.hbm.xml

<?xml version="1.0"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
  
<hibernate-mapping>  
 <class name="entity.Medicalspecialist" table="medicalspecialist" catalog="mydb">  
 <id name="idMedicalSpecialist" type="java.lang.Integer">  
 <column name="idMedicalSpecialist" />  
 <generator class="identity" />  
 </id>  
 <property name="name" type="string">  
 <column name="name" length="45" not-null="true" />  
 </property>  
 <property name="surname" type="string">  
 <column name="surname" length="45" not-null="true" />  
 </property>  
 <property name="fathersName" type="string">  
 <column name="fathersName" length="45" not-null="true" />  
 </property>  
 <property name="password" type="string">  
 <column name="password" length="45" not-null="true" />  
 </property>  
  
 </class>  
</hibernate-mapping>

### medicalunit.hbm.xml

<?xml version="1.0"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
  
<hibernate-mapping>  
 <class name="entity.MedicalUnit" table="medicalunit" catalog="mydb">  
 <id name="idMedicalUnit" type="java.lang.Integer">  
 <column name="idMedicalUnit" />  
 <generator class="identity" />  
 </id>  
 <property name="comment" type="string">  
 <column name="comment" length="45" not-null="true" />  
 </property>  
 <property name="idConscriptCard" type="java.lang.Integer">  
 <column name="conscriptCard\_idConscriptCardId" not-null="true" />  
 </property>  
 <property name="idMedicalSpecialist" type="java.lang.Integer">  
 <column name="MedicalSpecialist\_idMedicalSpecialist" not-null="true" />  
 </property>  
 </class>  
</hibernate-mapping>

### hibernate.cfg.xml

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.bytecode.use\_reflection\_optimizer">false</property>  
 <property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>  
 <property name="hibernate.connection.password">1996</property>  
 <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mydb</property>  
 <property name="hibernate.connection.username">root</property>  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  
 <property name="show\_sql">true</property>  
  
 <mapping resource="hibernate\_mapping/medicalspecialist.hbm.xml"></mapping>  
 <mapping resource="hibernate\_mapping/conscriptcard.hbm.xml"></mapping>  
 <mapping resource="hibernate\_mapping/medicalunit.hbn.xml"></mapping>  
 </session-factory>  
</hibernate-configuration>

**MedicalCard.jsp**

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **page** language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
*<%--  
 Created by IntelliJ IDEA.  
 User: Max Berezynskyi  
 Date: 22.03.2015  
 Time: 15:48  
 To change this template use File | Settings | File Templates.  
--%>*<html>  
<head>  
 <title>Військкомат міста Києва</title>  
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
 <link rel="stylesheet" href="http://localhost:8080/MCS/css/styles.css">  
 <link rel="shortcut icon" href="http://localhost:8080/MCS/images/favicon.ico">  
</head>  
<body>  
  
<div class="header clearfix">  
 <div class="layout-positioner">  
 <div class="layout-column">  
 <img src="http://localhost:8080/MCS/images/gerb.png" class="header-img" onclick="window.location.href='/MCS/MedicalOffice'">  
 <div class="headerText">  
 <p>Військкомат міста Києва</p>  
 </div>  
 </div>  
 </div>  
</div>  
<div class="features clearfix">  
 <div class="layout-positioner">  
 <div class="layout-column">  
 <div class="feature hidden">  
  
 </div>  
 </div>  
 <div class="layout-column">  
 <div class="feature hidden">  
 <div class="med-profile">  
 <c:forEach var="unit" items="**${**medUnits**}**">  
 <div class="filling-unit">  
 <b><i><p><c:out value="**${**MedicalServlet.getSpecFIO(unit.idMedicalSpecialist)**}**"/> :</p></i></b>  
 <p>**${**unit.comment**}**</p>  
 </div>  
 </c:forEach>  
 </div>  
 </div>  
 </div>  
  
 <div class="layout-column">  
 <div class="feature hidden">  
  
 </div>  
 </div>  
 </div>  
</div>  
<div class="footer clearfix">  
</div>  
</body>  
</html>