

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА

з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

на тему: «Робоче місце медичного працівника призовного 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

ЗМІСТ .....	2
РОЗДІЛ 1 .....	3
ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ .....	3
1.1 Введення.....	3
1.2 Короткий огляд продукту.....	3
1.3 Ділові правила та приписи .....	4
1.4 Функціональність .....	5
1.5 Практичність.....	6
1.6 Надійність .....	6
РОЗДІЛ 2.....	7
РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ .....	7
2.1 Загальна схема прецедентів .....	7
2.2 Прецеденти для ролі медичного працівника.....	7
2.3 Діаграма бізнес-сутностей .....	9
2.4 Реляційна модель бази даних.....	9
2.5 Специфікація таблиць бази даних .....	10
РОЗДІЛ 3 .....	13
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ .....	13
3.1 Реляційно-об'єктне відображення.....	13
3.2 Специфікація HibernateUntil класу.....	17
3.3 Класи контролерів та їх специфікація .....	18
3.4 Hibernate mapping файли .....	20
РОЗДІЛ 4.....	21
ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ.....	21
4.1 Взаємодія медичного працівника і системи при прийомі призовника	
СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ .....	24
ДОДАТКИ.....	25
ДОДАТОК А.....	25

					6.050102 «Комп'ютерна інженерія»						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Нетудихата			Робоче місце медичного працівни- ка призовного відділення військо- вого комісаріату	Літ.		Арк.	Акрушів		
Перевір.		Болдак						2	33		
Реценз.						Кафедра Обчислюваль- ної техніки					
Н. Контр.											
Затверд.		Болдак									

# РОЗДІЛ 1

## ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ

### 1.1 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".

### 1.2 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.

					6.050102 «Комп'ютерна інженерія»	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

### 1.3 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 ;

					6.050102 «Комп'ютерна інженерія»	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

5. Система повертає сторінку із інформацією про призовника;
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.

## **1.4 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;

					6.050102 «Комп'ютерна інженерія»	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

- Перегляд профілю призовників;
- Внесення діагнозів до бази даних ;
- Перегляд медичних карток призовників .

## **1.5 Практичність**

### **1.5.1 Універсальність**

Сервіс може використовуватися у будь-якому військовому комісаріаті України.

## **1.6 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.

					6.050102 «Комп'ютерна інженерія»	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

## РОЗДІЛ 2

### РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

#### 2.1 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 графі у картці. Схема прецедентів представлена на рис. 2.1.

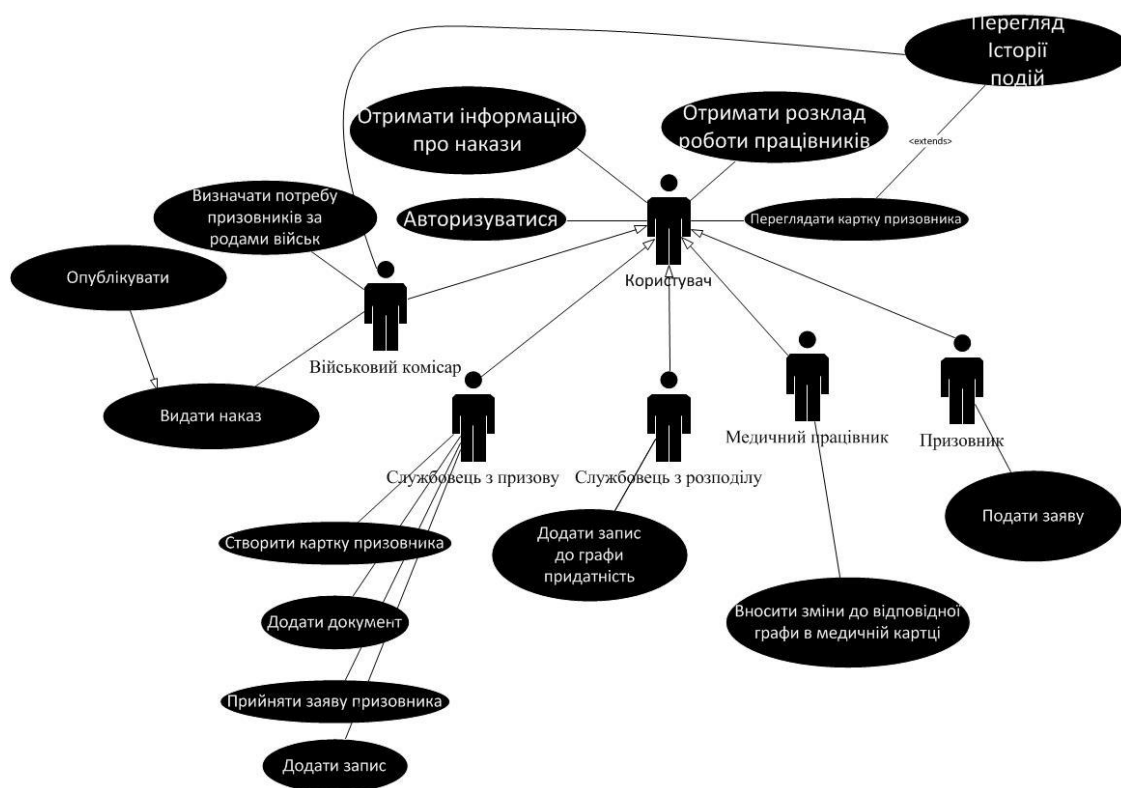


Рис. 2.1 – Загальна схема прецедентів для ролі користувача

#### 2.2 Прецеденти для ролі медичного працівника

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

## 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	
		2. Повертає сторінку із профілем призовника і полем введення діагнозу
	3. Додає запис до медичної картки і вводить власний унікальний пароль, натискає кнопку «Надіслати»	
		4. Звіряє пароль із базою даних.
		5. Ідентифікує лікаря за введеним паролем.
		6. Заносить коментар до бази даних.
Виключні ситуації	1. Призовник потребує додаткового обстеження на стаціонарі	
	2. Медичний працівник ввів невірний пароль	



### 2.3 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.

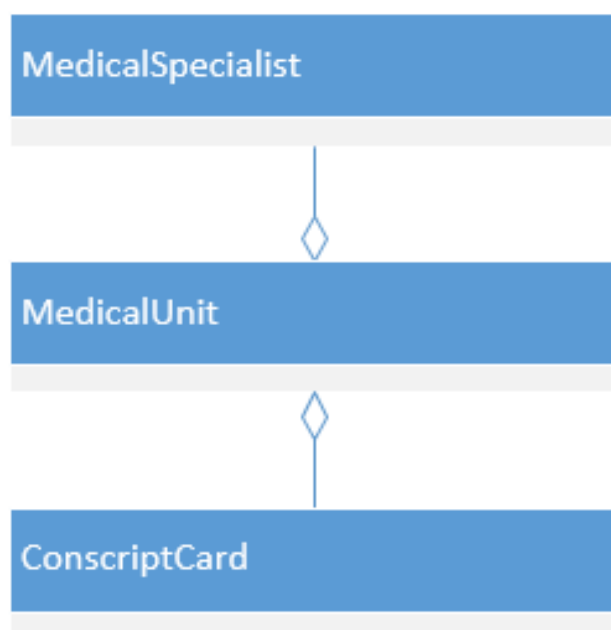


Рис 2.2 – Діаграма бізнес-сутностей

### 2.4 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.

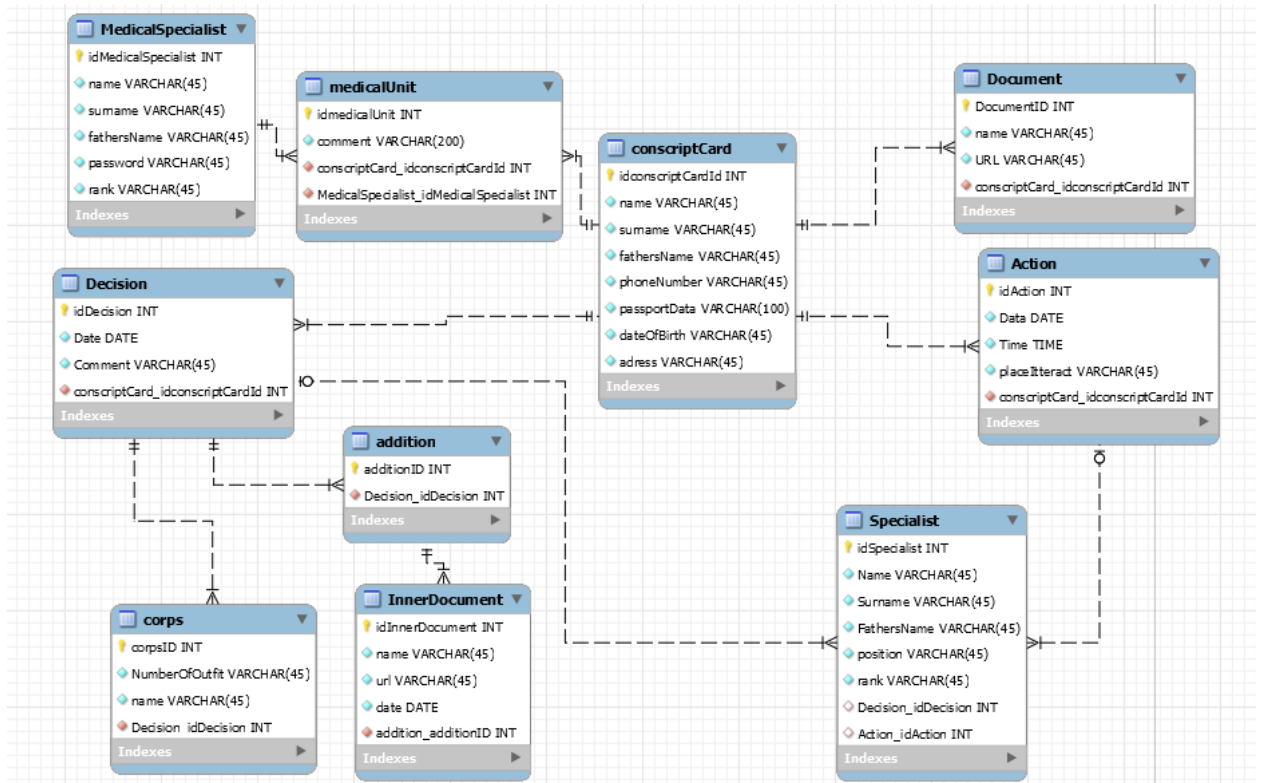


Рис. 2.3 – Реляційна модель бази даних

## 2.5 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	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
idSpecialist	INT	Так	Ні	Так	Посилання на медичного спеціаліста

## РОЗДІЛ 3

### РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

#### 3.1 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.

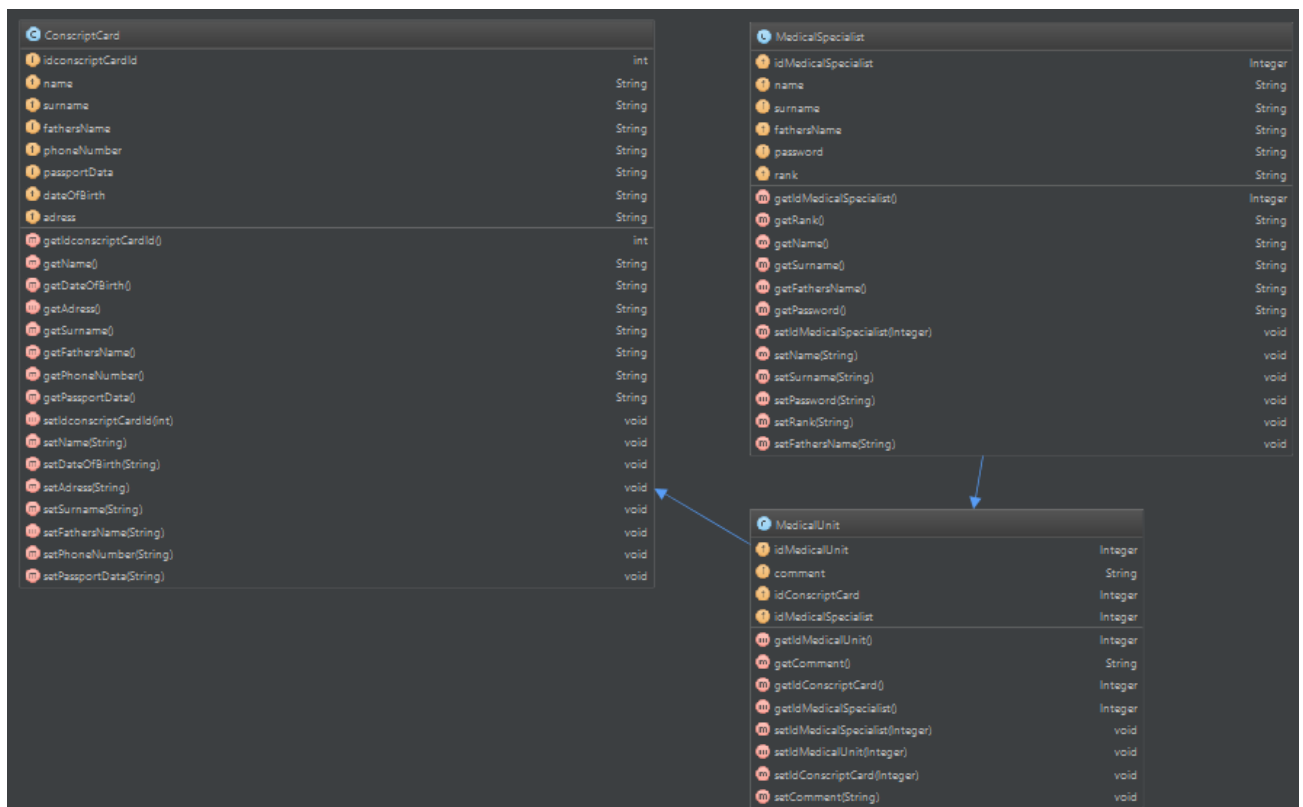


Рис 3.1 – Реляційно-об'єктне відображення

### 3.1.2 Клас «ConscriptCardDAO»

```
public class ConscriptCardDAO  
extends java.lang.Object
```

Клас DAO для роботи із сутностями типу ConscriptCard

Author:

Anton Netudykhata and Max Berezhynskyi 16.03.2015.

#### Constructor Summary

##### Constructors

##### Constructor and Description

ConscriptCardDAO()

#### Method Summary

##### All Methods

##### Instance Methods

##### Concrete Methods

##### Modifier and Type

##### Method and Description

void

addToDB(ConscriptCard card)  
Додає до бази даних об'єкт

void

deleteFromDB(java.lang.Integer id)  
Видаляє із БД по значенню id

java.util.List<ConscriptCard>

getCards()  
Повертає список карток призовників

ConscriptCard

getConscriptCardById(int id)  
Повертає картку призовника, по заданому id

### 3.1.3 Клас «MedicalSpecialistDAO»

```
public class MedicalSpecialistDAO  
extends java.lang.Object
```

Клас DAO для роботи із сутностями типу Medicalspecialist

Author:

Anton Netudykhata and Max Berezynskyi 14.03.2015.

#### Constructor Summary

##### Constructors

##### Constructor and Description

MedicalSpecialistDAO()

#### Method Summary

##### All Methods

##### Instance Methods

##### Concrete Methods

##### Modifier and Type

##### Method and Description

void

addToDB(MedicalSpecialist medspec)  
Додає до БД об'єкт класу Medicalspecialist

void

deleteFromDB(java.lang.Integer id)  
Видаляє із БД спеціаліста із заданим id

MedicalSpecialist

getMedicalSpecialistById(int id)  
Повертає об'єкт класу Medicalspecialist по заданому id

java.util.List<MedicalSpecialist>

getSpecialists()  
Повертає список об'єктів класу Medicalspecialist

### 3.1.4 Клас «MedicalUnitDAO»

```
public class MedicalUnitDAO  
extends java.lang.Object
```

Клас DAO для роботи із сутностями типу MedicalUnit

Author:

Anton Netudykhata and Max Berezhynskyi 14.03.2015.

#### Constructor Summary

##### Constructors

##### Constructor and Description

MedicalUnitDAO()
------------------

#### Method Summary

##### All Methods

##### Instance Methods

##### Concrete Methods

##### Modifier and Type

##### Method and Description

void	addToDB(MedicalUnit unit) Заносить об'єкт класу MedicalUnit до БД
void	deleteFromDB(java.lang.Integer id) Видаляє із БД об'єкт по його заданому id
MedicalUnit	getMedicalUnitById(int id) Повертає об'єкт класу, по заданому id
java.util.List<MedicalUnit>	getUnits() Повертає список об'єктів класу MedicalUnit

Змн.	Арк.	№ докум.	Підпис	Дата



## 3.2 Специфікація HibernateUtil класу

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.

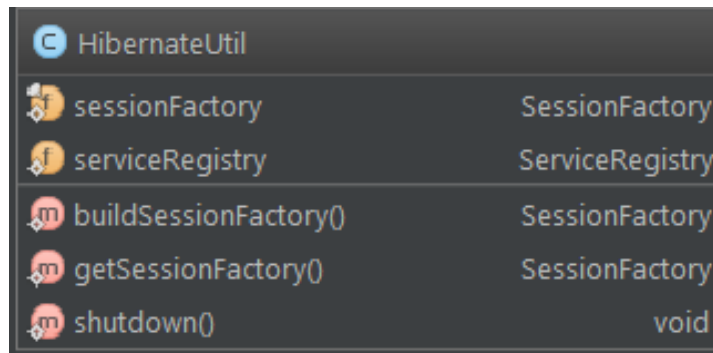


Рис 3.2. – Діаграма HibernateUnit класу

### 3.2.1 Клас «HibernateUtil»

```
public class HibernateUtil
extends java.lang.Object
```

Клас для взаємодії з конфіг файлами і створення об'єкту SessionFactory, котрий відповідає за створення hibernate-сесії

Author:

Anton Netudykhata and Max Berezhynskyi 14.03.2015.

#### Constructor Summary

##### Constructors

##### Constructor and Description

`HibernateUtil()`

#### Method Summary

##### All Methods

##### Static Methods

##### Concrete Methods

##### Modifier and Type

##### Method and Description

`static org.hibernate.SessionFactory` `getSessionFactory()`  
Повертає об'єкт SessionFactory

`static void` `shutdown()`  
Очищує кеш і закриває з'єднання із БД

### 3.3 Класи контролерів та їх специфікація

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

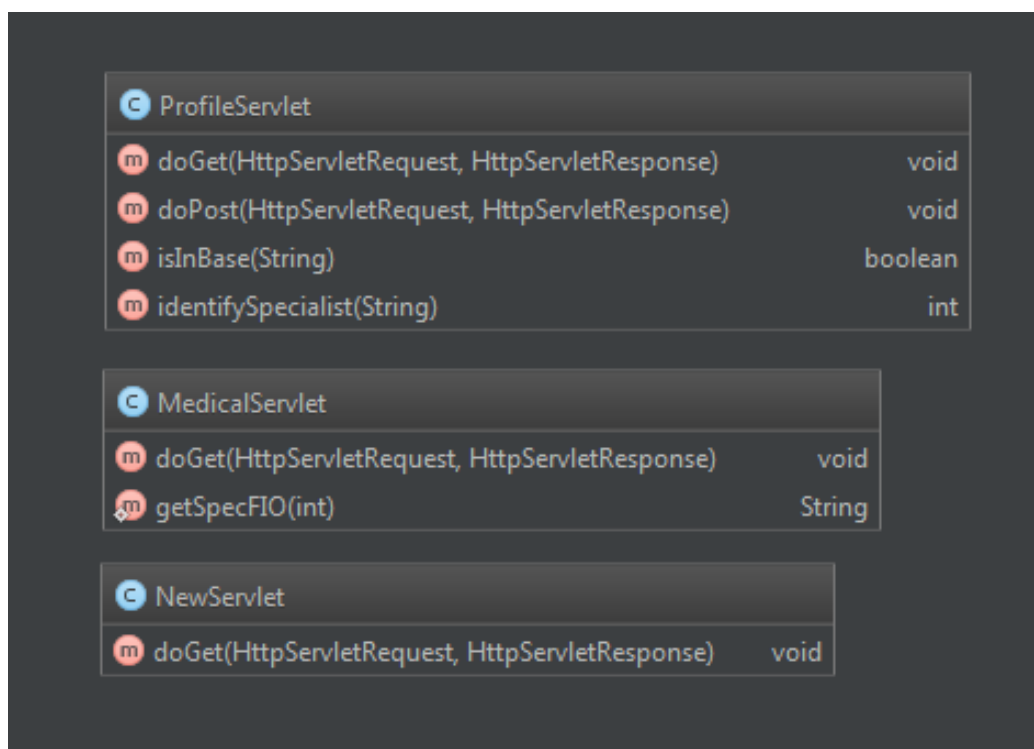


Рис 3.3 – Діаграма контролерів

## 3.3.2 Клас «MedicalServlet»

```
@WebServlet(value="/MedUnit")
public class MedicalServlet
extends javax.servlet.http.HttpServlet
```

Сервлет для взаємодії із сторінкою /medicalCard.jsp

Author:

Anton Netudykhata and Max Berezhynskyi 22.03.2015.

See Also:

Serialized Form

### Constructor Summary

#### Constructors

Constructor and Description

MedicalServlet()

### Method Summary

#### All Methods

#### Static Methods

#### Instance Methods

#### Concrete Methods

Modifier and Type

Method and Description

protected void

doGet(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse resp)  
Отримуємо список медичних карток, і видаляємо у списку все, що не стосується заданого id призовника, чия картка відкрита.

static java.lang.String

getSpecFIO(int id)  
Повертає прізвище, ім'я, по-батькові спеціаліста, із заданим id

### Method Detail

#### doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest req,
    javax.servlet.http.HttpServletResponse resp)
    throws javax.servlet.ServletException,
    java.io.IOException
```

Отримуємо список медичних карток, і видаляємо у списку все, що не стосується заданого id призовника, чия картка відкрита. Відредагований список відправляємо на сторінку /medicalCard.jsp

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

req - запит

resp - відповідь

Throws:

javax.servlet.ServletException - необхідне виключення

java.io.IOException - необхідне виключення

#### getSpecFIO

```
public static java.lang.String getSpecFIO(int id)
```

Повертає прізвище, ім'я, по-батькові спеціаліста, із заданим id

Parameters:

id - id потрібного спеціаліста

Returns:

прізвище, ім'я, по-батькові спеціаліста

### 3.4 Hibernate mapping файли

Hibernate mapping файли відповідають за взаємодію наших Entity об'єктів з Hibernate і з базою даних.

Список XML файлів наведений на рис 3.4

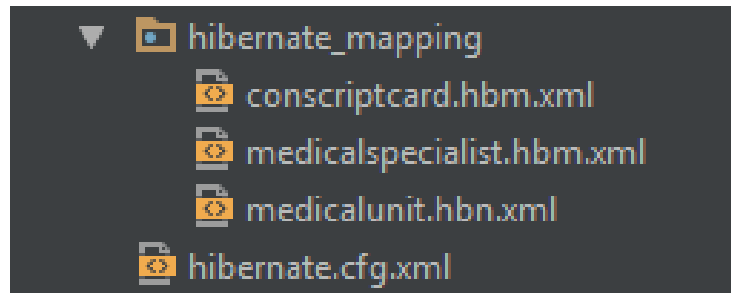


Рис 3.4 – Список XML файлів

## РОЗДІЛ 4

### ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ

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

#### 4.1 Взаємодія медичного працівника і системи при прийомі призовника

4.1.1 Головна сторінка сайту відображає медичному спеціалісту список всіх призовників(рис 4.1).

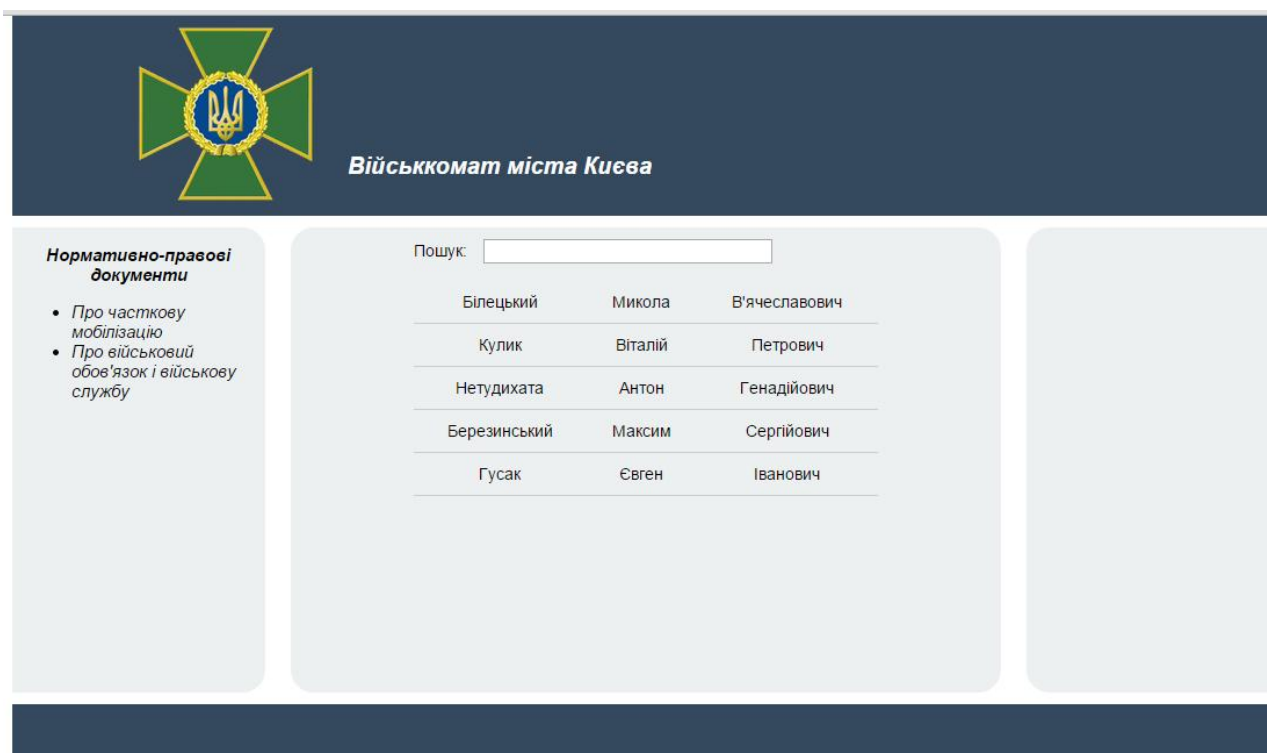


Рис 4.1 – Головна сторінка сайту

#### 4.1.2 Реалізація функції пошуку.

The screenshot shows the header of the 'Військкомат міста Києва' (Military Commissariat of Kyiv) website, featuring the Ukrainian coat of arms. Below the header, there is a sidebar on the left titled 'Нормативно-правові документи' (Normative-legal documents) with two links: 'Про часткову мобілізацію' and 'Про військовий обов'язок і військову службу'. The main content area has a search bar labeled 'Пошук:' with the text 'бе' entered. Below the search bar, the name 'Березинський Максим Сергійович' is displayed. The right side of the page is a large empty light gray box.


Рис 4.2 –Реалізація функції пошуку

#### 4.1.3 Відображення профілю призовника

The screenshot shows the profile page of a conscript in the 'Військкомат міста Києва' website. The header is the same as in the previous screenshot. The sidebar on the left is identical. The main content area is titled 'Profile' with a plus icon. It features a placeholder image of a man's face. To the right of the image, the following personal data is listed: 'Ім'я : Максим', 'Прізвище : Березинський', 'По-батькові : Сергійович', 'Дата народження : 22.02.96', 'Місце проживання : Соборності 10', 'Паспортні дані : АП894321', and 'Номер телефону : 0995194121'. Below this information is a text area labeled 'Коментар лікаря'. At the bottom right of the profile section, there is a login form with the label 'Ваш пароль:', a password input field, and a 'Відправити' (Send) button. The right side of the page is a large empty light gray box.

Рис 4.3 –Профіль призовника

#### 4.1.4 Відображення медичної картки призовника



**Військокомат міста Києва**

**Нормативно-правові документи**

- Про часткову мобілізацію
- Про військовий обов'язок і військову службу

**Офтальмолог (Бороний Микола Сергійович) :**  
Зір поганий.

**Уролог (Іванов Петро Іванович) :**  
Здоровий

**Хірург (Лопатюк Василь Георгійович) :**  
Здоровий, як бик!

Рис 4.4 –Медична картка призовника

## СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. DAO. – Посилання: [https://ru.wikipedia.org/wiki/Data\\_Access\\_Object](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/Сервлет_(Java))

					6.050102 «Комп'ютерна інженерія»	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		



**ConscriptCardDAO.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()).buildServiceReg
istry();

            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>
    </session-factory>
</hibernate-configuration>

```

```

        <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">
            
            <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>
    </div>
</div>

```



```
        </div>
      </div>
    </div>

    <div class="layout-column">
      <div class="feature hidden">

        </div>
      </div>
    </div>
  </div>
  <div class="footer clearfix">
  </div>
</body>
</html>
```