**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

**Кафедра** Програмного забезпечення комп'ютерних систем
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня

*бакалавра*
(назва освітньо-кваліфікаційного рівня)

**студента** *Котка Максима Валерійовича*
(ПІБ)

**академічної групи** *122-18ск-1*
(шифр)

**спеціальності** *122 Комп'ютерні науки*
(код і назва спеціальності)

**освітньої програми** *Комп'ютерні науки*
(назва освітньої програми)

**на тему:** *Розробка додатку «Рєєстратура» для організації прийомів лікаря з використанням фреймворків Swing та Hibernate*

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| кваліфікаційної роботи | *доц. Гуліна І.Г.* | | | |
| **розділів:** | | | | |
| спеціальний | *доц. Гуліна І.Г.* | | | |
| економічний | *проф. Вагонова О.Г.* | | | |
| | | | | |
| | | | | |
| **Рецензент** | *доц. Шедловський І.А.* | | | |
| **Нормоконтролер** | *доц. Гуліна І.Г.* | | | |

**Дніпро**
**2021**

**Міністерство освіти і науки України**
**НТУ «Дніпровська політехніка»**

**ЗАТВЕРДЖЕНО:**
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

# ЗАВДАННЯ
## на кваліфікаційну роботу

*бакалавра*
(назва освітньо-кваліфікаційного рівня)

**студента** *122-18ск-1* *Котка Максима Валерійовича*
(група) (прізвище та ініціали)

**тема кваліфікаційної роботи** *Розробка додатку «Рєєстратура» для організації прийомів лікаря з використанням фреймворків Swing та Hibernate*

затверджена наказом ректора НТУ «ДП» від «07»червня 2021 р. № 317-с

| Розділ | Зміст виконання | Термін виконання |
|---|---|---|
| ***Спеціальний*** | ***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.***<br>***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми.*** | ***13.01.2021 р.*** |
| ***Економічний*** | ***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки.*** | ***27.05.2021 р.*** |

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Котко М.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *14.06.2021 р.*

# РЕФЕРАТ

Пояснювальна записка: 86 с., 38 рис., 3 дод., 25 джерел.

Об'єкт розробки: інформаційна система обліку заявок виклоків лікаря додому.

Мета кваліфікаційної роботи: підвищення ефективності роботи з обліку заявок викликів лікаря додому за рахунок розробки та використанню відповідної інформаційної системи.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування підсистеми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає в автоматизації процесса заповнення форм викликів лікарів, зборі інформації у загальну базу даних та її систематизацію.

Актуальність підсистеми маршрутизації запитів визначається великим попитом на подібні розробки.

Список ключових слів: JAVA, HIBERNATE, SWING, XAMPP, MYSQL, БАЗА ДАНИХ.

## ABSTRACT

Explanatory note: 86 pp., 38 figs., 3 apps., 25 sources.

Object of development: information system for accounting applications for doctor's calls home.

The purpose of the qualification work: to increase the efficiency of work on the registration of applications for doctor's calls home by developing and using an appropriate information system.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the subsystem, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and application load, describes the program.

The economic section determines the complexity of the developed information subsystem, calculates the cost of work to create an application and calculates the time for its creation.

The practical significance lies in the automation of the process of filling in the doctor's call forms, collecting information in a common database and its systematization.

The relevance of the query routing subsystem is determined by the high demand for such developments.



Keywords: JAVA, HIBERNATE, SWING, XAMPP, MYSQL, DATABASE.

**ЗМІСТ**

# СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

ЕОМ - електронно-обчислювальна машина;

ІС - інформаційна система;

АІС - автоматизована інформаційна система;

ПЗ - програмне забезпечення;

ЕМК - електронна медична картка;

ІТ - інформаційні технології.

# ВСТУП

Бази даних (БД) складають в даний час основу комп'ютерного забезпечення інформаційних процесів, що входять практично в усі сфери людської діяльності. Дійсно, процеси обробки інформації мають загальну природу і спираються на опис фрагментів реальності, виражене у вигляді сукупності взаємопов'язаних даних. Бази даних є ефективним засобом представлення структур даних і маніпулювання ними. Однією з найважливіших завдань підтримки сталого розвитку технологій є моніторинг основних показників їх функціонування. Як правило, для моніторингу потрібна спеціально організована постійно діюча система збору, аналізу та поширення інформації, що забезпечує контроль змін деякої просторово-обмеженої частини реального світу. Така система зазвичай називається автоматизованою інформаційною системою (АІС). АІС являє собою комплекс програмних, технічних, інформаційних, лінгвістичних, організаційно-технічних засобів і персоналу, призначений для збору, обробки (первинної), зберігання, пошуку, обробки (вторинної) та видачі даних в заданій формі (вигляді) для вирішення різнорідних професійних або призначених для користувача завдань. На відміну від інформаційно-пошукових систем - АІС характеризуються:

– багатофункціональністю (тобто здатністю вирішувати різноманітні завдання;

– незалежністю процесів збору (первинної) обробки, введення даних і їх поновлення (актуалізації) від процесів або використання прикладними програмами;

– незалежністю прикладних програм від фізичної організації баз даних. Тому створення АІС для моніторингу різних характеристик навколишнього середовища є необхідним завданням.

Метою бакалаврської кваліфікаційної роботи є розробка фрагмента автоматизованої інформаційної системи обліку заявок за викликом лікаря додому.

Великий обсяг просторових даних, їх характеристик, необхідність багатофакторного аналізу цих даних роблять необхідною і візуалізацію даних. Це і визначає актуальність обраної теми.

У кваліфікаційній роботі бакалавра вирішується завдання проектування бази даних (зміст БД) і реалізації БД в середовищі СУБД MySQL.

Завдання даної кваліфікаційної роботи та об'єкт розробки безпосередньо пов'язані з зі спеціальністю 122 «Комп’ютерні науки» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1. Загальні відомості з предметної галузі

В якості прикладу, для розробки системи може бути взята веб-орієнтована система HELSI.ME (https://helsi.me). Це сучасна, зручна та надійна електронна медична система, створена для пацієнтів, лікарів, державних та приватних медичних закладів (рис. 1.1.).

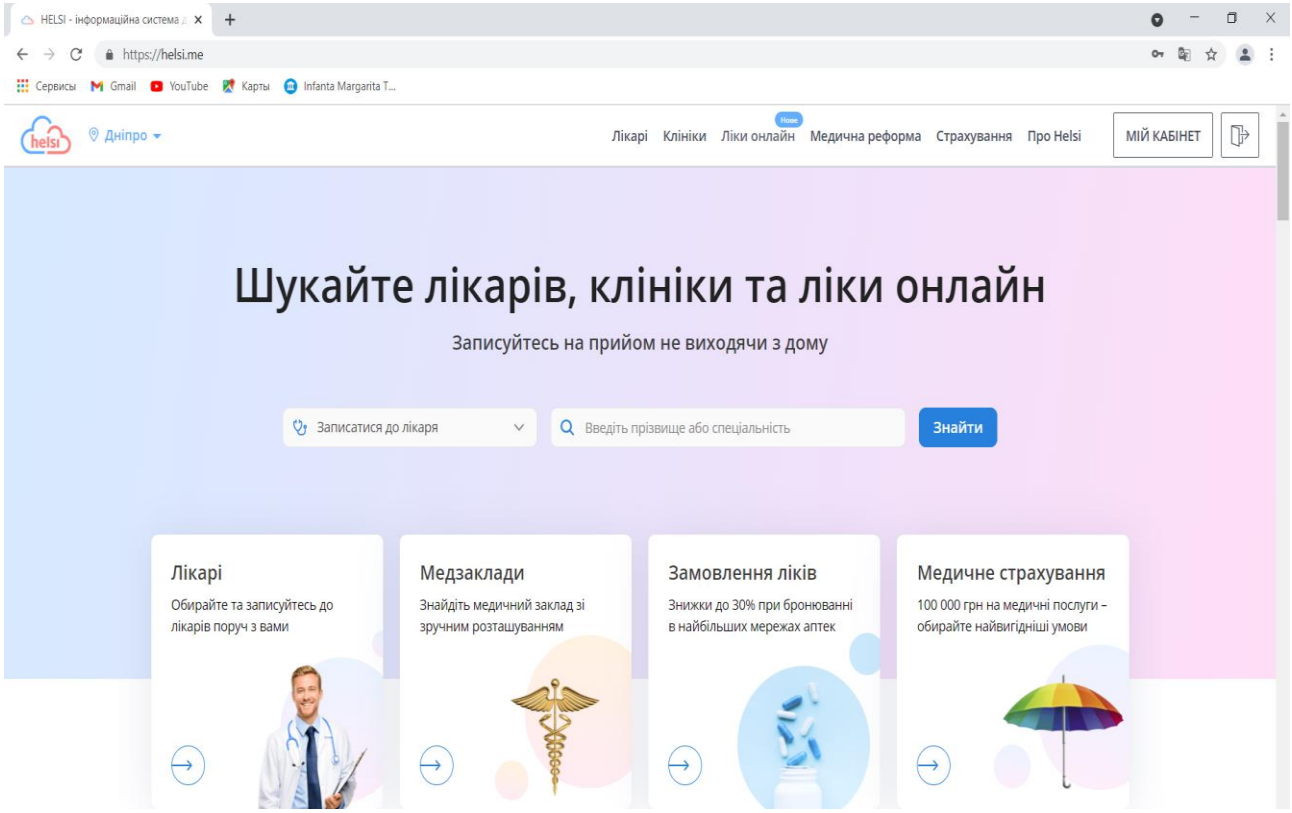



Рис. 1.1. стартова сторінка медичної системи HELSI.ME

HELSI надає для пацієнтів:

- можливість легко знайти та обрати свого лікаря;
- швидкий запис на прийом on-line себе та членів своєї родини;
- доступ до своєї електронної медичної картки (ЕМК);
- миттєві результати аналізів та діагностики в кабінеті пацієнта;

- доступ до призначень лікаря та плану лікування.

HELSI надає для лікарів:

- зручне ведення історії хвороби пацієнтів та ЕМК;

- оперативне отримання результатів діагностики та аналізів;

- легке використання клінічних протоколів;

- зрозумілий кабінет для ведення прийому пацієнтів.

HELSI надає для медичних закладів:

- комплексну автоматизацію роботи медичного закладу;

- можливість налаштування Helsi під потреби медзакладу;

- функціонал для участі в реформі (закріплення пацієнтів за лікарями, реєстрація декларацій з пацієнтами);

- моніторинг та управлінська статистика для керівників;

- формування поточної звітності та статистики.

Можливості HELSI:

- Автоматизація реєстратури та роботи лікаря.

- Управління розкладом лікаря.

- Ведення електронних медичних карток (ЕМК).

- Облік медичних препаратів та ведення оплат.

- Формування звітів та статистики.

- Підтримка формату DICOM та стандарту HL7.

- Конструктор бланків і форм.

- Інтуїтивно зрозумілий веб-інтерфейс.

- Гнучке налаштування прав доступу.

- Фіксація дій користувачів.

- Доступ пацієнтів до своєї електронної медичної картки (ЕМК).

- Забезпечення надійного шифрування та безпека даних.

- Портал з інформацією про лікарів, годинами їх прийому та адресами лікарень.

- Підтримка для лікарів та пацієнтів від контакт-центру.

Helsi надійно захищає дані пацієнтів. Вона безпечна та конфендаційна. Всі дані зберігаються у дата-центрі, який отримав сертифікат комплексної системи захисту інформації (КСЗІ) від Державної служби спеціального зв'язку та захисту інформації України.

Обробка персональної інформації громадян відбувається в правовому полі України.

Але дана система має такий недолік, як робота лише через веб інтерфейс та наявність мережі Інтернет. Також великий функціонал, який не завжди використовується користувачами на всі 100%.

Виходячи з мети роботи, інформаційна система, що розробляється, повинна надавати наступний функціонал:

• перегляд історії викликів;

• створення нових викликів;

• можливість перегляду повного списку лікарів і пацієнтів;

• пошук і сортування по записах в базі даних.

Для реалізації перерахованих вище функцій і досягнення мети кваліфікаційної роботи необхідно вирішити наступні завдання:

– ознайомитися і оволодіти мовою програмування Java;

– дослідити середовище Intellij IDEA версії 2016.2.5 і налаштувати цю IDE;

– розробити дизайн і визначити основні частини програми;

– опанувати створенням інтерфейсу користувача для програм використовуючи вбудовану бібліотеку Swing;

– спроектувати базу даних MySQL для зберігання інформації;

– налаштувати локальний віртуальний сервер для коректної взаємодії програми та бази даних;

– ознайомиться і опанувати фреймворком Hibernate;

– вирішити і створити алгоритми для обміну даними між сервером і додатком.

Структура програми повинна складатися з стартового вікна, що містить таблицю з історією викликів, і 5 додаткових розділів, які необхідні по ходу роботи програми:

«Список лікарів» - розділ з повним списком лікарів;

«Список пацієнтів» - розділ з повним списком пацієнтів;

«Додати запис» - розділ для додавання нового виклику;

«Інформація про лікаря» - розділ з повною інформацією про обраний лікаря;

«Інформація про пацієнта» - розділ з повною інформацією про обраний пацієнта.

### 1.2. Призначення розробки та галузь застосування

Інформаційна система – це сукупність функціональних елементів, фахівців та інформаційних технологій, об’єднаних інформаційними потоками в єдину організаційну структуру для реалізації стратегій підприємства.

Концепція баз даних припускає використання інтегрованих засобів зберігання інформації, що дозволяють забезпечити централізоване управління даними і обслуговування ними багатьох користувачів. При цьому БД повинна підтримуватися в комп’ютерному середовищі єдиним програмним забезпеченням, що називається системою керування базами даних (СКБД). Одне з основних призначень СКБД – підтримка програмними засобами подання, відповідного реальності.

Розробка інформаційних систем різних класів в наш час вкрай актуальна. Всі інформаційні системи пов’язані з функціями довготривалого зберігання і обробки інформації, яка є фактором, що визначає ефективність будь-якої сфери діяльності.

Основним призначенням системи, що розробляється є надання можливості ведення реєстрації викликів лікаря, зберігання та обробки даної інформаціїї. Дана система може бути використана як додаткова система рєєстрації викликів

лікаря, може бути використана як альтернативна система, що має можливість працювати стаціонарно, офф-лайн.

### 1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи є:

- освітня програма спеціальності 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;
- завдання на кваліфікаційну роботу на тему: «Розробка додатку «Рєєстратура» для організації прийомів лікаря з використанням фреймворків Swing та Hibernate».

### 1.4. Постановка завдання

Завданням до даної кваліфікаційної роботи, є створення бази даних та відповідної інформаційної системи. Інформаційна система, що розробляється в роботі, повинна забезпечувати оператору можливість реєстрації нових викликів, перегляду списків всіх лікарів і пацієнтів, а також перегляд історії викликів конкретного пацієнта.

До функцій, які повинні бути реалізовані відносяться:

• Реєстрація нових викликів лікарів додому.

• Перегляд списку лікарів.

• Перегляд списку пацієнтів.

• Можливість перегляду історії викликів конкретного пацієнта.

Основними проблемами при складанні бази даних є:

• Редагування, додавання і зберігання даних.

• Пошук пацієнта по довільній характеристиці.

Систематизація даних дозволяє швидко провести пошук за характером параметрів, а також в зручній формі зберігати, редагувати, проводити аналіз інформації.

## 1.5. Вимоги до програми або програмного виробу

### 1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно підтримувати виконання наступних дій:

• перегляд історії викликів;

• створення нових викликів;

• можливість перегляду повного списку лікарів і пацієнтів;

• пошук і сортування по записах в базі даних.

Також система повиння мати зручний та інтуітивно зрозумілий інтерфейс.

### 1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

– семантичний та синтаксичний контроль вхідних даних;

– обробку виняткових ситуацій;

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи протягом не менше 120 годин (5 діб);

Як складовий елемент системи, повязаної із торгівельною діяльністю, підсистема маршрутизації запитів повинна мати наступні характеристики:

– час відновлення після збою - 5 хв;

– час відновлення після відмови одного з елементів підсистеми не повинно перевищувати половини операційного банківського дня;

– вірогідність виникнення не більше 2 логічних помилок на 1000 операторів за 1 рік експлуатації;

– забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови підсистеми.

### 1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми необхідно, щоб обчислювальна машина, на якій буде функціонувати веб-орієнтована підсистема, відповідала наступним вимогам:

– процесор класу Intel Xeon з тактовою частотою не менш 2.4 ГГц;

– не менше 2 GB оперативної пам'яті;

– рідкокристалічний монітор з діагоналлю не менше 17 ";

– 20 Гб вільного місця на жорсткому диску;

– доступ до мережі Internet;

– клавіатура;

– маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

### 1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати система, відповідало вимозі: робота під управлінням операційної системи сімейства Windows (7, 8, 10). Система є невимогливою до складу програмних засобів.

# РОЗДІЛ 2
# ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1. Функціональне призначення системи

Основним призначенням системи, що розробляється є надання можливості ведення реєстрації викликів лікаря, зберігання та обробки даної інформаціїї. Дана система може бути використана як додаткова система рєєстрації викликів лікаря, може бути використана як альтернативна система, що має можливість працювати стаціонарно та офф-лайн.

До функцій, які повинні бути реалізовані відносяться:

• Реєстрація нових викликів лікарів додому.

• Перегляд списку лікарів.

• Перегляд списку пацієнтів.

• Можливість перегляду історії викликів конкретного пацієнта.

Основними проблемами при складанні бази даних є:

• Редагування, додавання і зберігання даних.

• Пошук пацієнта по довільній характеристиці.

## 2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці інформаційної системи  математичні методи не використовувалися.

## 2.3. Опис використаних технологій та мов програмування

Java - сильно типізована об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (в подальшому придбаної компанією

Oracle). Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі, за допомогою віртуальної Java-машини. Дата офіційного випуску - 23 травня 1995 року.

Програми на Java транслюються в байт-код Java, який виконується віртуальною машиною Java (JVM).

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять зниження продуктивності. Ряд удосконалень кілька збільшив швидкість виконання програм на Java:

• застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді,

• широке використання переносних орієнтованого коду (native-код) в стандартних бібліотеках,

• апаратні засоби, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, підтримувана деякими процесорами фірми ARM).

За даними сайту shootout.alioth.debian.org, для семи різних завдань час виконання на Java становить в середньому в півтора-два рази більше, ніж для C / C ++, в деяких випадках Java швидше, а в окремих випадках в 7 разів повільніше. З іншого боку, для більшості з них споживання пам'яті Java-машиною було в 10-30 разів більше, ніж програмою на C / C ++. Також примітно дослідження, проведене компанією Google, згідно з яким відзначається істотно нижча

продуктивність і більше споживання пам'яті в тестових прикладах на Java в порівнянні з аналогічними програмами на C ++.

Ідеї, закладені в концепцію і різні реалізації середовища віртуальної машини Java, надихнули безліч ентузіастів на розширення переліку мов, які могли б бути використані для створення програм, що виконуються на віртуальній машині. Ці ідеї знайшли також вираз в специфікації загальномовна інфраструктури CLI, закладеної в основу платформи .NET компанією Microsoft.

Основні можливості Java:

1. автоматичне керування пам'яттю;

2. розширені можливості обробки виняткових ситуацій;

3. багатий набір засобів фільтрації введення-виведення;

4. набір стандартних колекцій: масив, список, стек і т.п. .;

5. наявність простих засобів створення мережевих додатків (у тому числі з використанням протоколу RMI);

6. наявність класів, що дозволяють виконувати HTTP-запити і обробляти відповіді;

7. вбудовані в мову засоби створення багатопоточних додатків, які потім були перенести на багато мов (наприклад, python);

8. уніфікований доступ до баз даних:

a) на рівні окремих SQL-запитів - на основі JDBC, SQLJ;

b) на рівні концепції об'єктів, що володіють здатністю до зберігання в базі даних - на основі Java Data Objects (англ.) і Java Persistence API;

9. підтримка узагальнень (починаючи з версії 1.5);

10. підтримка лямбда, замикань, вбудовані можливості функціонального програмування (з 1.8);

11. безліч варіантів реалізації багатопотокових програм.

Фреймворк Hibernate. Hibernate - бібліотека для мови програмування Java, призначена для вирішення завдань об'єктно-реляційного відображення (ORM), поширюється вільно на умовах GNU Lesser General Public License.

Метою Hibernate є звільнення розробника від значного обсягу порівняно низькоуровневого програмування при роботі в об'єктно-орієнтованих засобах в реляційної базі даних. Розробник може використовувати Hibernate як в процесі проектування системи класів і таблиць «з нуля», так і для роботи з уже існуючою базою даних.

Бібліотека не тільки вирішує завдання зв'язку класів Java з таблицями бази даних (і типів даних Java з типами даних SQL), але і також надає кошти для автоматичної генерації і оновлення набору таблиць, побудови запитів і обробки отриманих даних і може значно зменшити час розробки, яке зазвичай витрачається на ручне написання SQL- і JDBC-коду. Hibernate автоматизує генерацію SQL-запитів і звільняє розробника від ручної обробки результуючого набору даних і перетворення об'єктів, максимально полегшуючи перенесення (портирование) додатки на будь-які бази даних SQL.

Hibernate забезпечує прозору підтримку збереження даних (persistence) для «POJO» (тобто для стандартних Java-об'єктів); єдине сувора вимога для зберігається класу - наявність конструктора за замовчуванням (без параметрів). Для коректної поведінки в деяких додатках потрібно також приділити увагу методам equals () і hashCode ().

Mapping (зіставлення, проектування) Java-класів з таблицями бази даних здійснюється за допомогою конфігураційних XML-файлів або Java-анотацій. При використанні файлу XML Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання. У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості по організації відносини між класами «один-ко-многим» і «багато-до-багатьох». На додаток до управління зв'язками між об'єктами Hibernate також може управляти рефлексивними відносинами, де об'єкт має зв'язок «один-ко-многим» з іншими екземплярами свого власного типу даних.

Hibernate підтримує відображення для користувача типів значень. Це робить можливими такі сценарії:

• Перевизначення типу за замовчуванням SQL, Hibernate вибирає при відображенні стовпчика властивості.

• Проекція перераховується типу Java на поле БД, ніби вони є звичайними властивостями.

• Проекція одного властивості в кілька колонок.

Колекції об'єктів даних, як правило, зберігаються у вигляді колекцій Java-об'єктів, таких, як набір (Set) і список (List). Підтримуються узагальнені класи (Generics), введені в Java 5. Hibernate може бути налаштований на «ліниві» (відкладені) завантаження колекцій. Відкладені завантаження є варіантом за умовчанням, починаючи з Hibernate 3.

Пов'язані об'єкти можуть бути налаштовані на каскадні операції. Наприклад, батьківський клас Album (музичний альбом) може бути налаштований на каскадне збереження і / або видалення свого нащадка Track. Це може скоротити час розробки і забезпечити цілісність. Функція перевірки зміни даних (dirty checking) дозволяє уникнути непотрібного запису дій в базу даних, виконуючи SQL-оновлення тільки при зміні полів персистентних об'єктів.

Успіх бібліотеки Hibernate підштовхнув JCP до розробки специфікації JDO, що стала однією з стандартних технологій ORM на платформі JavaEE. Також Hibernate сумісна з JSR-220/317 і надає стандартні засоби JPA.

Hibernate забезпечує використання SQL-подібного мови Hibernate Query Language (HQL), який дозволяє виконувати SQL-подібні запити, записані поруч з об'єктами даних Hibernate. Запити критеріїв надаються як Об'єктно-орієнтована альтернатива до HQL.

Фреймворк Swing. Swing - бібліотека для створення графічного інтерфейсу для програм на мові Java. Swing був розроблений компанією Sun Microsystems. Він містить ряд графічних компонентів (англ. Swing widgets), таких як кнопки, поля введення, таблиці і т.д.

Swing відноситься до бібліотеки класів JFC, яка представляє собою набір бібліотек для розробки графічних оболонок. До цих бібліотек відносяться Java 2D, Accessibility-API, Drag & Drop-API і AWT.

Swing надає більш гнучкі інтерфейсні компоненти, ніж більш рання бібліотека AWT. На відміну від AWT, компоненти Swing розроблені для однаковою крос-платформної роботи, в той час як компоненти AWT повторюють інтерфейс виконуваної платформи без змін. AWT ж використовує тільки стандартні елементи ОС для відображення, тобто для кожного елементу створюється окремий об'єкт ОС (вікно), в зв'язку з чим, AWT не дозволяє створювати елементи довільної форми (можливо використовувати тільки прямокутні компоненти), елементи керування на основі AWT завжди відображаються поверх Swing-елементів (так як все Swing компоненти відображаються на поверхні контейнера).

Компоненти Swing підтримують специфічні динамічно підключаються види і поведінки (с англ. Plugable look-and-feel), завдяки якому можлива адаптація до графічного інтерфейсу платформи (тобто до компоненту можна динамічно скористатись іншим, специфічний для операційної системи, в тому числі і створений програмістом вигляд і поведінку). Таким чином, функції, які залежать Swing, можуть виглядати як рідні додатки для даної операційної системи. Основним мінусом таких «легких» (англ. Lightweight) компонентів є щодо повільна робота. Позитивна сторона - універсальність інтерфейсу створених додатків на всіх платформах.

«Lightweight» означає, що компоненти Swing відображаються самими компонентами на поверхні батьківського вікна, без використання компонентів операційної системи. На відміну від «Важких» компонентів AWT, в додатку Swing може бути тільки одне вікно, і всі інші компоненти будуються на найближчому батьку, що має власне вікно (наприклад, на JFrame). У додатку можуть поєднуватися Swing і AWT елементи, хоча це може породжувати деякі проблеми - зокрема, компоненти AWT завжди перекривають Swing елементи, а також закривають собою спливаючі меню JPopupMenu і JComboBox. Для

запобігання цьому, у цих компонентів є методи setLightWeightPopupEnabled (boolean), що дозволяють заборонити використання «легковагих» спливаючих елементів. При установці властивості в true (setLightWeightPopupEnabled (true)), AWT елементи не будуть перекривати меню.

### 2.4. Опис структури системи та алгоритмів її функціонування

Структура програми складається зі стартового вікна, що містить таблицю з історією викликів, і 5 додаткових розділів, які необхідні по ходу роботи програми:

- «Список лікарів» - розділ з повним списком лікарів;
- «Список пацієнтів» - розділ з повним списком пацієнтів;
- «Додати запис» - розділ для додавання нового виклику;
- «Інформація про лікаря» - розділ з повною інформацією про обраний лікаря;
- «Інформація про пацієнта» - розділ з повною інформацією про обраний пацієнта.

Концептуальна модель бази даних. Початковою стадією проектування системи баз даних є побудова семантичної моделі предметної галузі, яка базується на аналізі властивостей і природи об'єктів предметної галузі та інформаційних потреб майбутніх користувачів системи, що розробляється. Цю стадію прийнято називати концептуальним проектуванням системи, а її результат - концептуальною моделлю предметної галузі (об'єктом моделювання тут є предметна галузь майбутньої системи).

Такі моделі узагальнено представляють інформаційні потреби користувачів створюваної системи в частині використання збережених даних і по суті є засобом комунікації як розробників, так і користувачів на різних стадіях життєвого циклу бази даних.

Призначення концептуальних моделей визначає і деякі специфічні вимоги до засобів їх подання. Крім згаданої незалежності від середовища (обладнання) та вимоги адекватності відображення предметної галузі відзначимо наступні:

• формализованность, що забезпечує можливість автоматизованої обробки, в тому числі, наприклад, автоматичний контроль несуперечності;

• дружність, що забезпечує можливість використання наочних графічних засобів відображення і обробки їх користувачем.

До концептуальних моделей належать різні компоненти, по-різному і різними засобами відображають предметну галузь. Крім найбільш відомого опису об'єктів і зв'язків між ними (модель «сутність-зв'язок») до концептуального рівня опису предметної галузі можна віднести наступні компоненти:

• систему атрибутів і засобів опису предметної галузі. Наприклад, логічні (автоматичні) зв'язку між показниками або лінгвістичні властивості мови (синоніми, синтаксис і т.д.), яка використовується для вербального подання об'єктів;

• обмеження цілісності, що визначають допустимість значення окремих полів і взаємозв'язків як на рівні семантики вмісту БД, так і її фізичної структури (окремих файлів даних і взаємозв'язків між ними);

• опис інформаційних потреб користувачів, наприклад, у вигляді типових запитів, що відображають процедурні особливості доступу до даних.

Однією з найбільш популярних засобів формалізованого представлення предметної галузі систем, орієнтованих на обробку фактографічної інформації, є модель «сутність-зв'язок», яка покладена в основу значної кількості комерційних CASE-продуктів, що підтримують повний цикл розробки систем баз даних або окремі його стадії. При цьому багато хто з них не тільки підтримують стадію концептуального проектування предметної галузі розробляється, але і дозволяють здійснити на основі побудованої їх засобами моделі стадію логічного проектування шляхом автоматичної генерації концептуальної схеми бази даних

для обраної СУБД, наприклад, схеми бази даних для будь-якого SQL -севера або об'єктної СУБД.

Основними об'єктами (сутностями) в описі предметної галузі з точки зору бази даних є:

• пацієнт;

• лікар;

• виклик.

Атрибутами пацієнта є:

• номер пацієнта;

• ПІБ;

• дата народження;

• адреса проживання;

• Контактні телефони.

Атрибутами лікаря є:

• код лікаря;

• ПІБ;

• дата народження;

• спеціалізація;

• Контактні телефони.

Атрибутами виклику є:

• номер виклику;

• ПІБ лікаря;

• ПІБ пацієнта;

• дата виклику.

На опис предметної галузі а також описаних сутностей і їх атрибутів можна виділити наступні види зв'язків між сутностями бази даних (рис.2.1.):


Рис. 2.1. Концептуальна модель БД

Створення логічної моделі. Метою побудови логічної моделі є отримання графічного представлення логічної структури досліджуваної предметної галузі.

Логічна модель предметної галузі ілюструє суті, а також їх взаємовідносини між собою.

Суті описують об'єкти, які є предметом діяльності предметної галузі, і суб'єкти, що здійснюють діяльність в рамках предметної галузі. Властивості об'єктів і суб'єктів реального світу описуються за допомогою атрибутів.

Взаємини між сутностями ілюструються за допомогою зв'язків. Правила та обмеження взаємин описуються за допомогою властивостей зв'язків. Зазвичай зв'язку визначають якої залежності між сутностями, який вплив однієї сутності на іншу.

Таблиця 2.1.

**Атрибути та ключі сутностей логічної моделі**

| Сутність | Первинний ключ | Атрибути |
|---|---|---|
| Пацієнт | Номер рєєстрації | Номер рєєстрації<br>ПІБ<br>Дата народження<br>Адреса<br>Телефон |
| Виклик | Номер виклику | Номер виклику<br>ПІБ лікаря<br>ПІБ пацієнта<br>Дата виклику<br>Примітки |
| Лікар | Код лікаря | Код лікаря<br>ПІБ<br>Дата народження<br>Спеціалізація<br>Телефон |

Фізична модель визначає спосіб розміщення даних в середовищі зберігання і способи доступу до цих даних, які підтримуються на фізичному рівні.

Атрибут - це інформаційне відображення властивостей об'єкта. Кожен об'єкт характеризується рядом основних атрибутів і в фізичної моделі кожному атрибуту відповідає поле записи.

Фізична модель пацієнтів представлена в таблиці 2.2, викликів - в таблиці 2.3, а лікарів - в таблиці 2.4.

Таблиця 2.2.

**Таблиця пацієнтів**

| № п/п | Поле | Примітка | Тип |
|---|---|---|---|
| 1. | regNum | Номер рєєстрації | int(11) |
| 2. | FIO | ПІБ пацієнта | varchar(45) |
| 3. | dateOfBirth | Дата народження | date |
| 4. | address | Адреса проживання | varchar(65) |
| 5. | phones | Телефон | varchar(45) |

Таблиця 2.3.

**Таблиця викликів**

| № п/п | Поля | Примітка | Тип |
|---|---|---|---|
| 1. | recNum | Номер виклику | int(11) |
| 2. | docFio | ПІБ лікаря | varchar(45) |
| 3. | patFio | ПІБ пацієнта | varchar(45) |
| 4. | recDate | Дата виклику | date |
| 5. | notes | Примітка | text |

Таблиця 2.4.

**Таблиця лікарів**

| № п/п | Поля | Примітка | Тип |
|---|---|---|---|
| 1. | docCode | Код лікаря | int(11) |
| 2. | FIO | ПІБ лікаря | varchar(45) |
| 3. | dateOfBirth | Дата народження | date |
| 4. | spec | Спеціалізація | varchar(45) |
| 5. | phones | Телефон | varchar(45) |

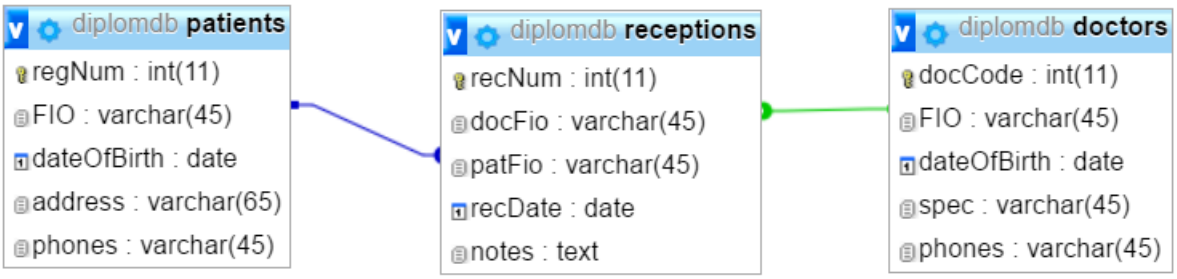



Рис. 2.2. Физична модель БД

Створення бази даних в phpMyAdmin. Для створення інформаційної системи робочого місця оператора-рєєстратора викликів, необхідно створити базу даних з наступними таблицями: ПАЦІЄНТИ, ВИКЛИКИ і ЛІКАРІ.

Створимо таблицю лікарів, в якій буде міститися інформація про лікарів лікувального закладу. Поле docCode відповідає унікальним кодом лікаря, в FIO зберігається ПІБ лікаря, в dateOfBirth - дата народження, в spec - спеціалізація, а в phones - контактні телефони для зв'язку в разі надзвичайної ситуації.

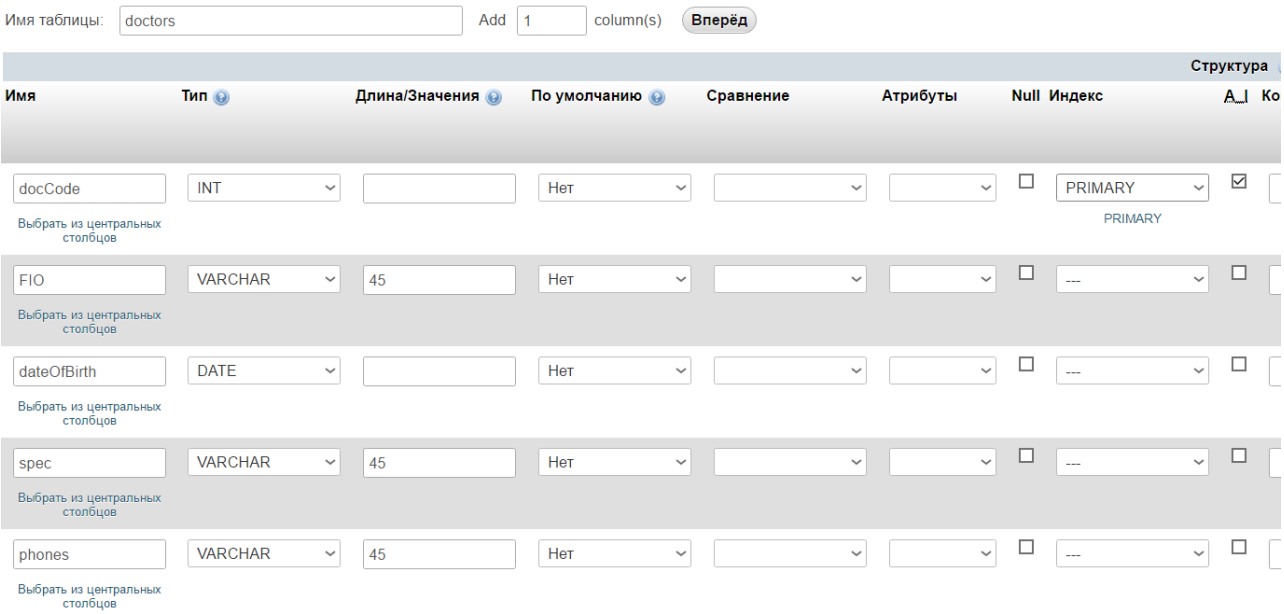
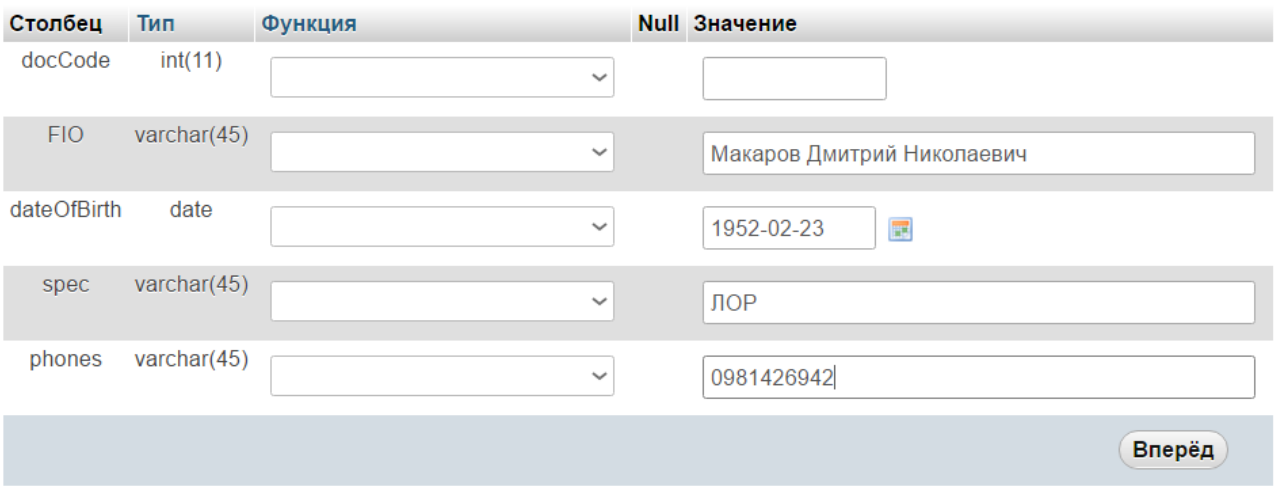



Рис. 2.3. Створення таблиці doctors

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **docCode** | int(11) | | | Нет | *Нет* |
| ☐ | 2 | **FIO** | varchar(45) | | | Нет | *Нет* |
| ☐ | 3 | **dateOfBirth** | date | | | Нет | *Нет* |
| ☐ | 4 | **spec** | varchar(45) | | | Нет | *Нет* |
| ☐ | 5 | **phones** | varchar(45) | | | Нет | *Нет* |

Рис. 2.4. Структура таблиці doctors



| Столбец | Тип | Функция | Null | Значение |
|---|---|---|---|---|
| docCode | int(11) | | | |
| FIO | varchar(45) | | | Макаров Дмитрий Николаевич |
| dateOfBirth | date | | | 1952-02-23 |
| spec | varchar(45) | | | ЛОР |
| phones | varchar(45) | | | 0981426942 |

Вперёд

Рис. 2.5. Додавання запису до таблиці doctors

Створимо таблицю пацієнтів (рис.2.6), в якій буде зберігатися інформація про пацієнтів лікувального закладу. Поле regNum відповідає унікальному реєстраційним номером, в FIO зберігається ПІБ пацієнта, в dateOfBirth - його дата народження, в address - адреса проживання, а в phones - контактні телефони.

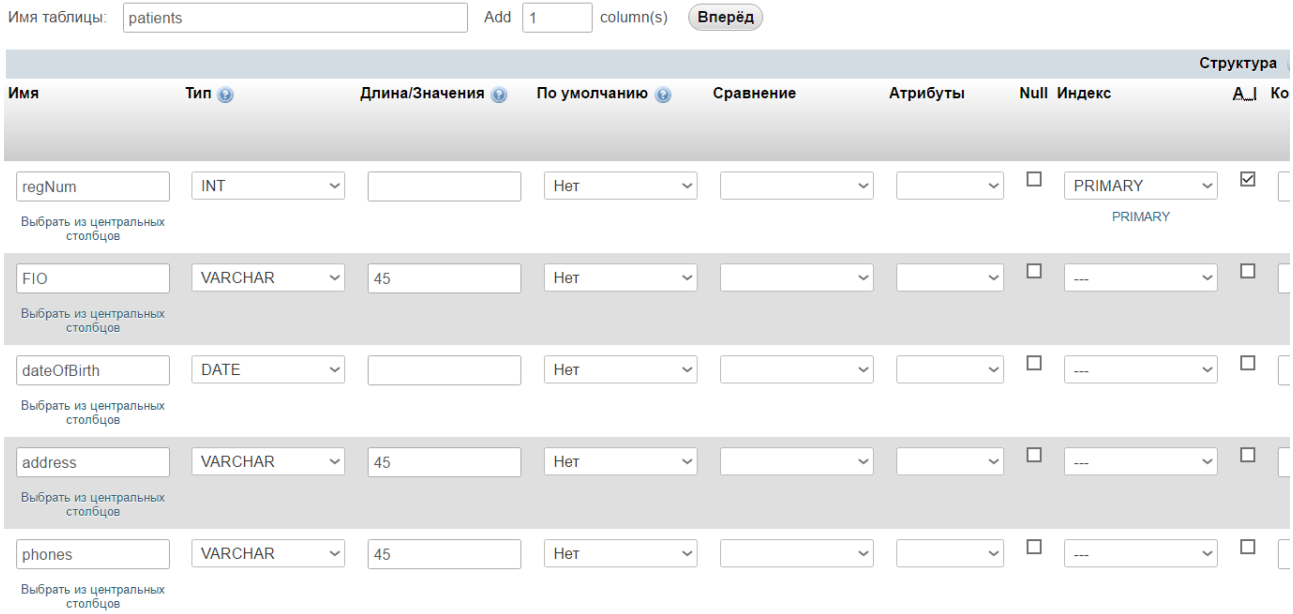



Рис. 2.6. Створення таблиці patients

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | **regNum** 🔑 | int(11) | | | Нет | *Нем* | AUTO_INCREMENT |
| ☐ | 2 | **FIO** | varchar(45) | | | Нет | *Нем* | |
| ☐ | 3 | **dateOfBirth** | date | | | Нет | *Нем* | |
| ☐ | 4 | **address** | varchar(65) | | | Нет | *Нем* | |
| ☐ | 5 | **phones** | varchar(45) | | | Нет | *Нем* | |

Рис. 2.7. Структура таблиці patients

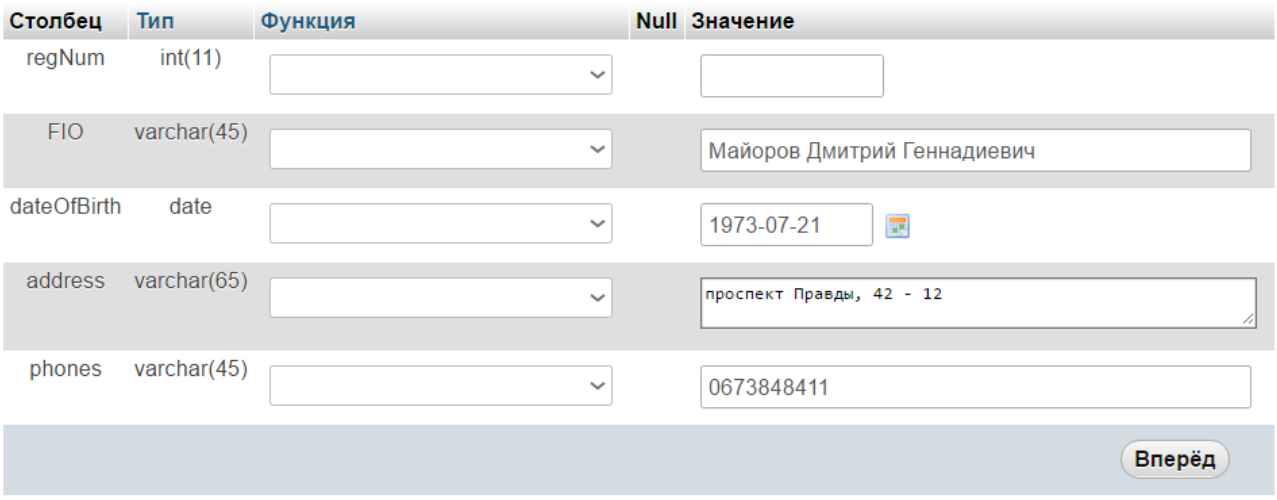



Рис. 2.8. Додавання запису до таблиці patients

Створимо таблицю викликів, що містить інформацію про виклики лікарів лікувального закладу. Поле recNum відповідає унікальним номером виклику, docFio містить ПІБ лікаря, patFio - пацієнта, в recDate - дата виклику, а в notes - примітки до виклику, такі як симптоми і рекомендації з лікування.

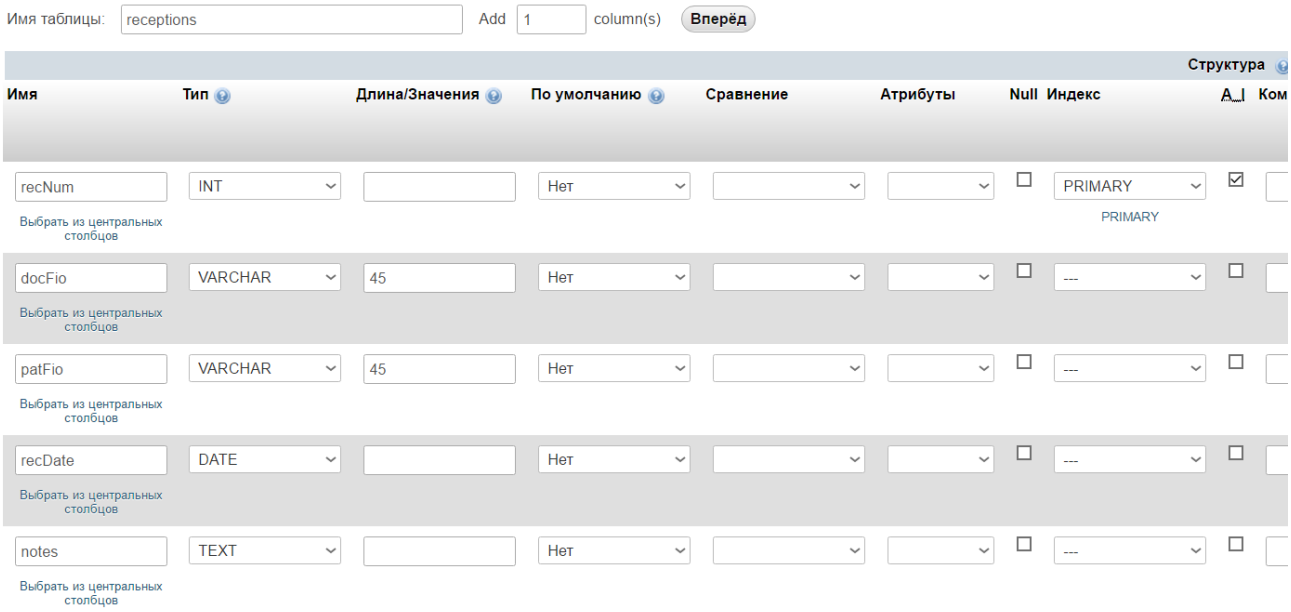



Рис. 2.9. Створення таблиці receptions

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | **recNum** | int(11) | | | Нет | *Нем* | AUTO_INCREMENT |
| ☐ | 2 | **docFio** | varchar(45) | | | Нет | *Нем* | |
| ☐ | 3 | **patFio** | varchar(45) | | | Нет | *Нем* | |
| ☐ | 4 | **recDate** | date | | | Нет | *Нем* | |
| ☐ | 5 | **notes** | text | | | Нет | *Нем* | |

Рис. 2.10. Структура таблиці receptions

| Столбец | Тип | Функция | Null | Значение |
|---|---|---|---|---|
| recNum | int(11) | | | |
| docFio | varchar(45) | | | Веткова Татьяна Никитична |
| patFio | varchar(45) | | | Димитриев Андрей Игнатович |
| recDate | date | | | 2017-06-19 |
| notes | text | | | |

Вперёд

Рис. 2.11. Додавання запису до таблиці receptions

## 2.5. Обгрунтування та організація вхідних та вихідних даних програми

В якості вхідних даних в програмі використовується інформація про клієнтів, лікарів та викликів, що заноситься до БД (див. п.2.4.).

В якості вихідних даних в програмі використовуються файли звіту роботи програми та результати запитів користувача до БД.

## 2.6. Опис роботи розробленої системи

### 2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

- ✓ процесор класу Intel ® Xeon з тактовою частотою 2.4GHz;
- ✓ шина даних - 1066MHz,
- ✓ кеш другого рівня - 2048 КБ;
- ✓ оперативна пам’ять 2 x DIMM DDR2-800 1024 Мб;
- ✓ жорсткі диски 3x 250 Гб SATA 2 16 Мб буфер, 7200 RPM;
- ✓ рідкокристалічний монітор з діагоналлю не менше 17 ";
- ✓ доступ до мережі Internet;
- ✓ клавіатура;
- ✓ маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

### 2.6.2. Використані програмні засоби

IntelliJ IDEA - інтегроване середовище розробки програмного забезпечення на багатьох мовах програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains. На рис. 2.12. зображений інтерфейс програми.

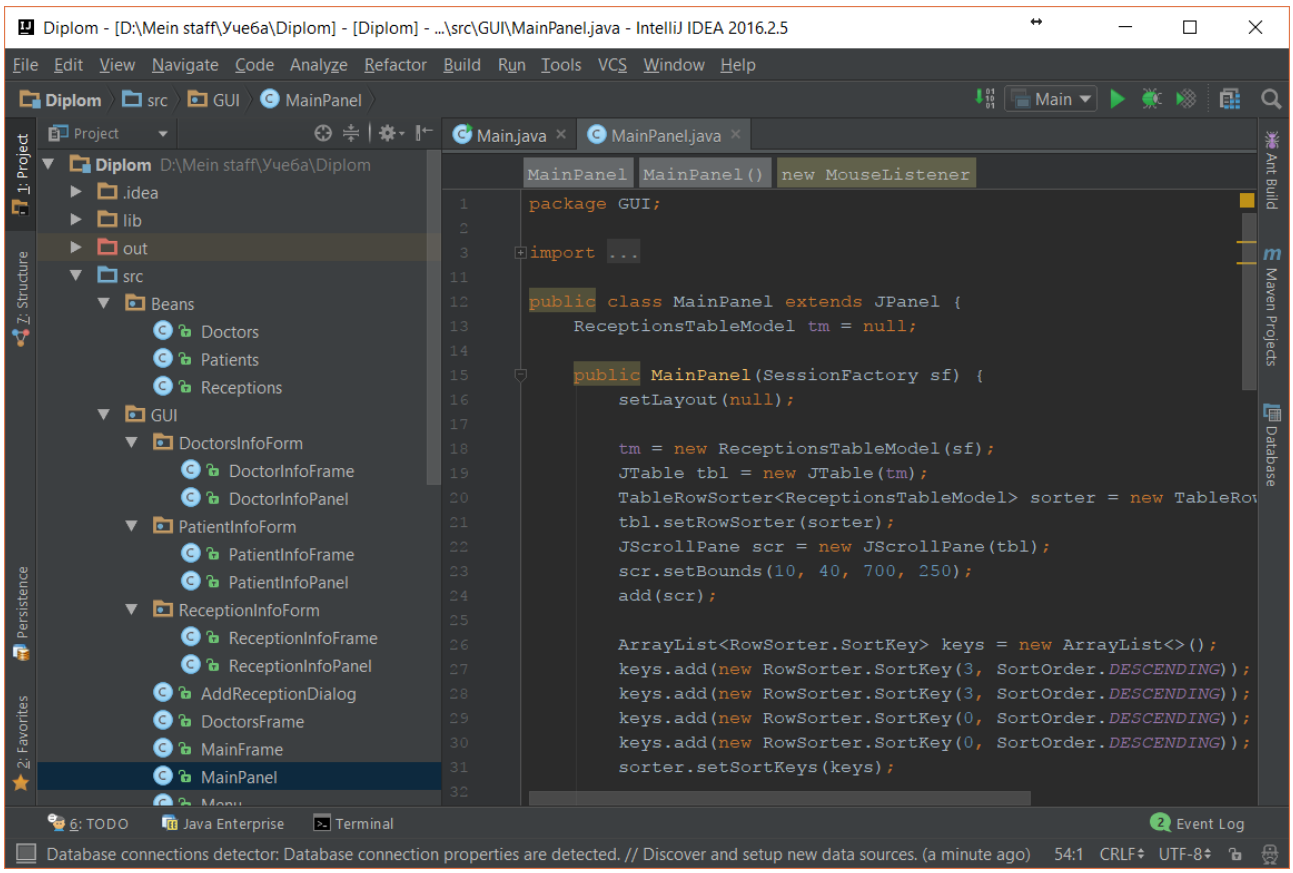



Рис. 2.12. Інтерфейс Intellij IDEA

Перша версія з'явилася в січні 2001 року і швидко стала популярною, як перша середа для Java з широким набором інтегрованих інструментів для рефакторинга, які дозволяли програмістам швидко реорганізовувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного інтерфейсу користувача. Серед інших можливостей, середа добре сумісна з багатьма популярними вільними інструментами розробників, такими як CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування на мові Ruby.

Починаючи з версії 9.0, середа доступна в двох редакціях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією,

доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В редакції Ultimate Edition реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов та фреймворків.

Мови: Java, JavaScript, CoffeeScript, HTML / XHTML / HAML, CSS / SASS / LESS, XML / XSL / XPath, YAML, ActionScript / MXML, Python, Ruby, Haxe, Groovy, Scala, SQL, PHP, Kotlin, Clojure , Ci, C ++.

Ряд мов підтримані за допомогою плагінів сторонніх розробників, зокрема, так реалізована підтримка OCaml, GLSL, Erlang, Fantom, Go, Haskell, Lua, Mathematica, Rust, Perl5.

Також, що не менш важливо, має ряд широко поширених фреймворків (наприклад, Spring), які легко настроюються і можуть бути підключені до проекту в пару клацань.

Сервер Apache і БД MySQL під керуванням XAMPP. XAMPP є повністю безкоштовним і простим в установці дистрибутивом Apache, MySQL, FileZilla, Mercury і Tomcat. Він створений з відкритим вихідним кодом, щоб бути неймовірно простим в установці і у використанні.

Мета XAMPP є створення простого в установці дистрибутива для розробників, щоб легко увійти в світ Apache. Щоб зробити його зручним для розробників, XAMPP налаштований з усіма включеними функціями. В даний час є дистрибутиви для Windows, Linux, і OS X.

Для установки XAMPP необхідно завантажити один файл формату zip, tar або exe. Компоненти програми не вимагають настройки. Програма регулярно оновлюється для включення до складу новітніх версій Apache / MySQL / PHP і Perl. Також в складі XAMPP присутні інші модулі, включаючи OpenSSL і phpMyAdmin.

Інтерфейс програми настільки простий, що її називають «складанням для ледачих» ( «lazy man's WAMP / LAMP installation»). Установка XAMPP займає менше часу, ніж установка кожного компонента окремо.

Даний web-сервер поширюється в повній, стандартної і зменшеною (відомої як XAMPP Lite) версіях. Всі додаткові модулі також доступні для скачування.

З додаткових можливостей можна відзначити, що сама компанія випускає пакети оновлень, що випускаються у вигляді zip, 7-zip, tar або exe, які дозволяють відновити всі компоненти з однієї версії збірки xampp на новішу.

Спочатку XAMPP створювався як інструмент для розробників, дозволяючи веб-дизайнерам і програмістам тестувати свою роботу, не використовуючи Інтернет. Для спрощення роботи деякі можливості і настройки безпеки відключені за замовчуванням, і в цілому XAMPP рекомендується до використання тільки в дуже дружньому оточенні. Однак XAMPP іноді використовується і у всесвітній павутині. Також програма підтримує створення і управління базами даних MySQL і SQLite.

XAMPP, як і інші подібні пакети, можна використовувати для установки власної копії Вікіпедії на комп'ютер.

Установка сайту безпосередньо на XAMPP-сервер полягає в копіюванні файлів сайту в папку htdocs.

На Рис. 1.2 зображена контрольна панель з включеними сервером Apache і базою даних MySQL.

Apache HTTP-сервер (вимовляється /ə.pæ.ʧi/, названий ім'ям групи племен північноамериканських індіанців апачів, крім того, є скороченням від англ. A patchy server; серед російських користувачів загальноприйнято спотворене апач) - вільний веб-сервер.

Apache є кросплатформним ПО, підтримує операційні системи Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Основними достоїнствами Apache вважаються надійність і гнучкість конфігурації. Він дозволяє підключати зовнішні модулі для надання даних, використовувати СУБД для аутентифікації користувачів, модифікувати повідомлення про помилки і т. Д. Підтримує IPv6.

Ядро Apache включає в себе основні функціональні можливості, такі як обробка конфігураційних файлів, протокол HTTP і система завантаження модулів. Ядро (на відміну від модулів) повністю розробляється Apache Software Foundation, без участі сторонніх програмістів.

Теоретично, ядро apache може функціонувати в чистому вигляді, без використання модулів. Однак, функціональність такого рішення вкрай обмежена.

Ядро Apache повністю написано на мові програмування C.

Система конфігурації Apache заснована на текстових конфігураційних файлах. Має три умовних рівня конфігурації:

• Конфігурація сервера (httpd.conf).

• Конфігурація віртуального хоста (httpd.conf с версії 2.2, extra / httpd-vhosts.conf).

• Конфігурація рівня директорії (.htaccess).

Має власну мову конфігураційних файлів, заснований на блоках директив. Практично всі параметри ядра можуть бути змінені через конфігураційні файли, аж до управління MPM. Велика частина модулів має власні параметри.

Частина модулів використовує в своїй роботі конфігураційні файли операційної системи (наприклад / etc / passwd і / etc / hosts).

Крім цього, параметри можуть бути задані через ключі командного рядка.

Існує безліч модулів, що додають до Apache підтримку різних мов програмування і систем розробки.

До них відносяться: PHP (mod_php), Python (mod python, mod wsgi), Ruby (apache-ruby), Perl (mod perl), ASP (apache-asp) і Tcl (rivet).

Крім того, Apache підтримує механізми CGI і FastCGI, що дозволяє виконувати програми на практично всіх мовах програмування, в тому числі C, C ++, Lua, sh, Java.

Apache має різні механізми забезпечення безпеки і розмежування доступу до даних. Основними є:

• Обмеження доступу до певних директорій або файлів.

• Механізм авторизації користувачів для доступу до директорії на основі HTTP-аутентифікації (mod_auth_basic) і digest-аутентифікації (mod_auth_digest).

• Обмеження доступу до певних директорій або всьому сервера, засноване на IP-адреси користувачів.

• Заборона доступу до певних типів файлів для всіх або частини користувачів, наприклад заборона доступу до конфігураційним файлів і файлів баз даних.

Існують модулі, що реалізують авторизацію через СУБД або PAM.

У деяких MPM-модулях є можливість запуску кожного процесу Apache використовуючи різні uid і gid з відповідними цим користувачам і групам користувачів.

Також, існує механізм suexec, який використовується для запуску скриптів і CGI-додатків з правами і ідентифікаційними даними користувача.

Для реалізації шифрування даних, що передаються між клієнтом і сервером використовується механізм SSL, реалізований через бібліотеку OpenSSL. Для підтвердження автентичності веб-сервера використовуються сертифікати X.509.

Існують зовнішні засоби забезпечення безпеки, наприклад mod_security.

За базу даних використовується MySQL з веб-інтерфейсом phpMyAdmin, потрапити в який можна включивши сервер Apache.

phpMyAdmin - веб-додаток з відкритим кодом, написаний на мові PHP і представляє собою веб-інтерфейс для адміністрування СУБД MySQL. PHPMyAdmin дозволяє через браузер і не тільки здійснювати адміністрування сервера MySQL, запускати команди SQL і переглядати вміст таблиць і баз даних. Додаток користується великою популярністю у веб-розробників, так як дозволяє управляти СУБД MySQL без безпосереднього введення SQL команд, надаючи дружній інтерфейс.

На сьогоднішній день PHPMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, враховуючи всі нововведення СУБД MySQL. Переважна більшість російських

провайдерів використовують цю програму в якості панелі управління для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

Додаток поширюється під ліцензією GNU General Public License і тому багато інших розробники інтегрують його в свої розробки, наприклад XAMPP, Denwer, AppServ, Open Server.

Проект на даний момент часу локалізований на більш ніж 62 мовах.

### 2.6.3. Виклик та завантаження програми

Спосіб виклику програми з відповідного носія даних та умови його завантаження є стандартними для запуску виконуючих файлів при роботі в ОС Windows. Додаткових чи специфічних вимог щодо запуску програми не встановлено, програма не потребує спеціального завантаження та налаштування. Для роботи потрібен ПК чи ноутбук з ОС Windows.

### 2.6.4. Опис інтерфейсу користувача

При запуску програми з’являється головне вікно.

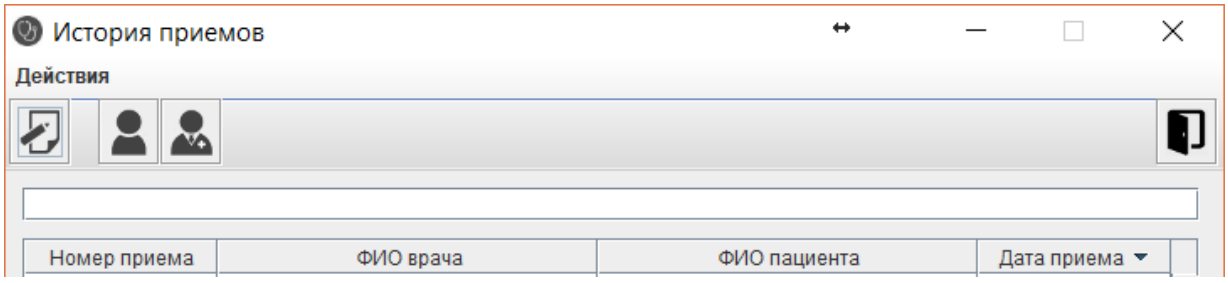



Рис. 2.13. Основне вікно програми

Інтерфейс побудований таким чином, що будь-який користувач, навіть не працював до цього з комп'ютером, може здогадатися про призначення кожного компонента. Інтерфейс має дуже зручний вид. Вікно головної форми має меню у відповідність з рис. 2.14, що дозволяє здійснювати навігацію по програмі.

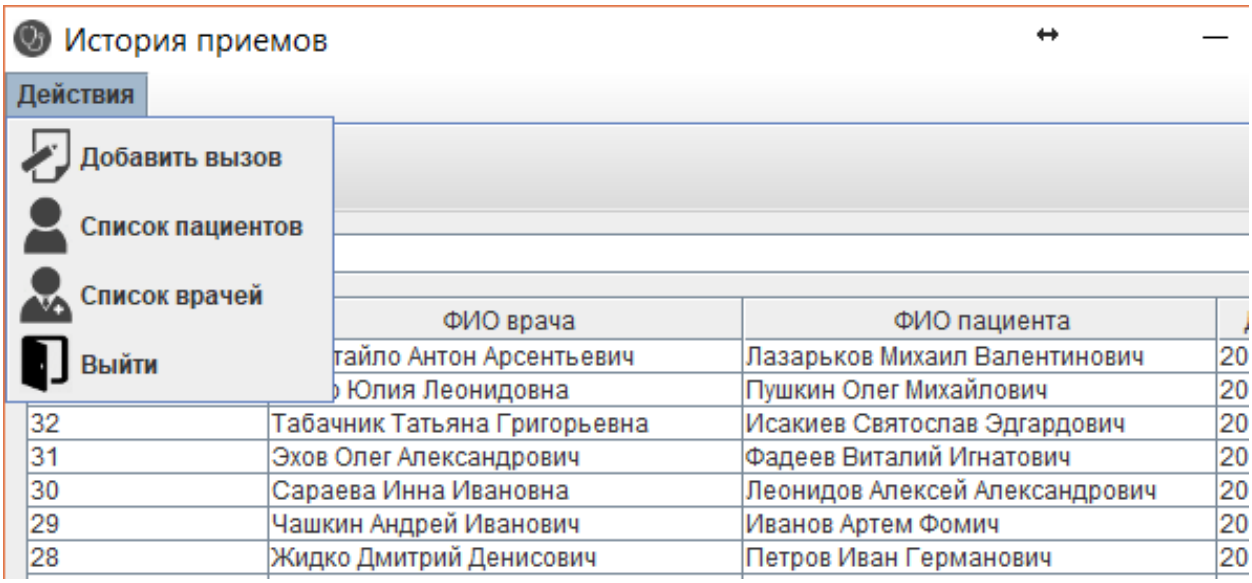



Рис.2.14. Меню основного вікна програми

Після відкриття головної форми у з'являється можливість почати роботу в програмі. Використовуючи меню або кнопки тулбара можна отримати доступ до вікон додавання викликів (рис.2.15), повного списку пацієнтів (рис.2.16) і лікарів (рис.2.17).

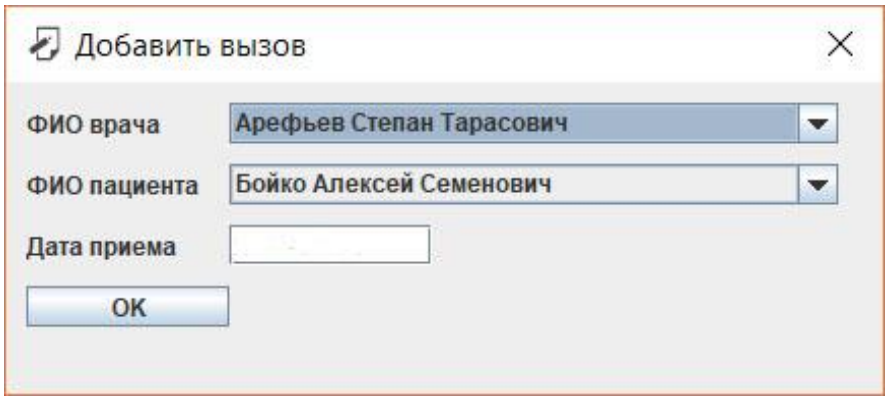



Рис. 2.15. Вікно додавання нового виклику

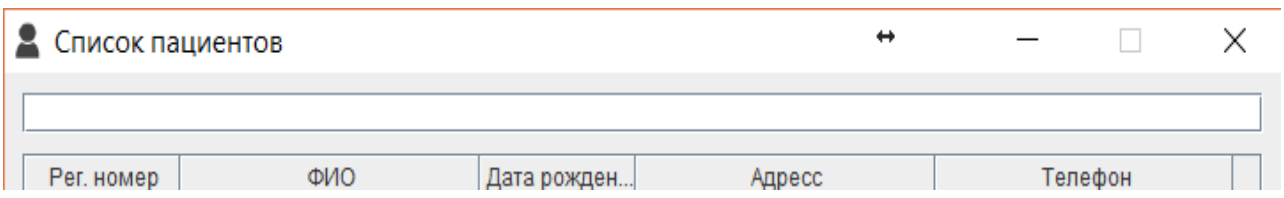
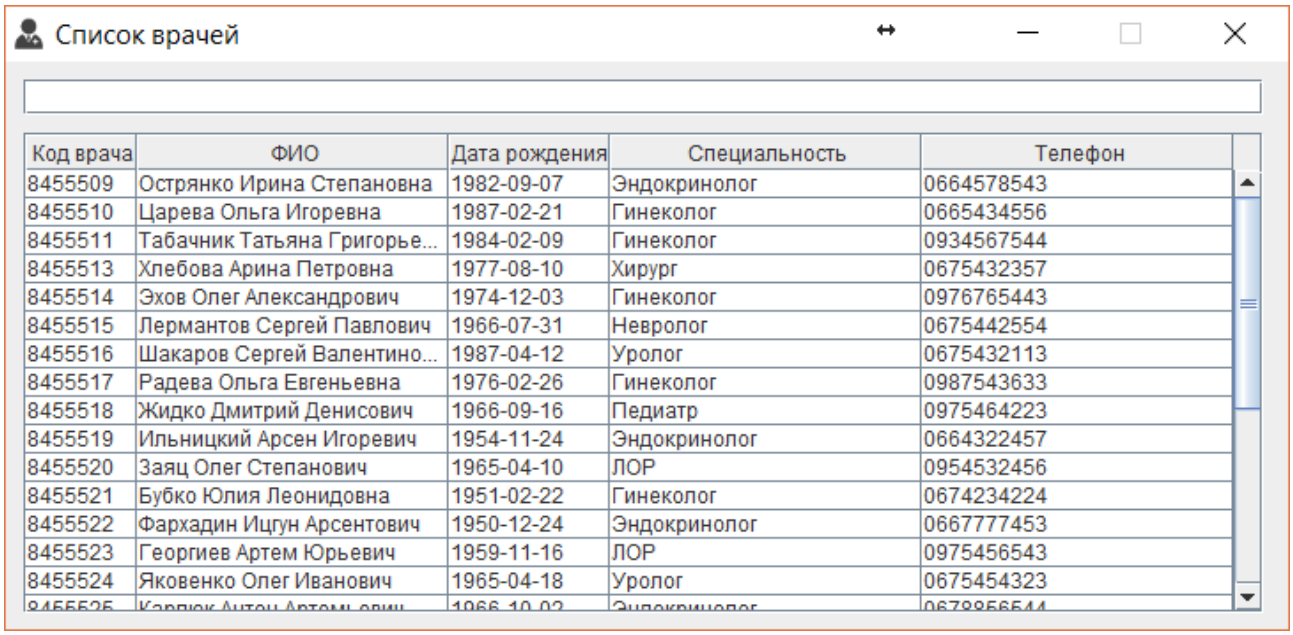


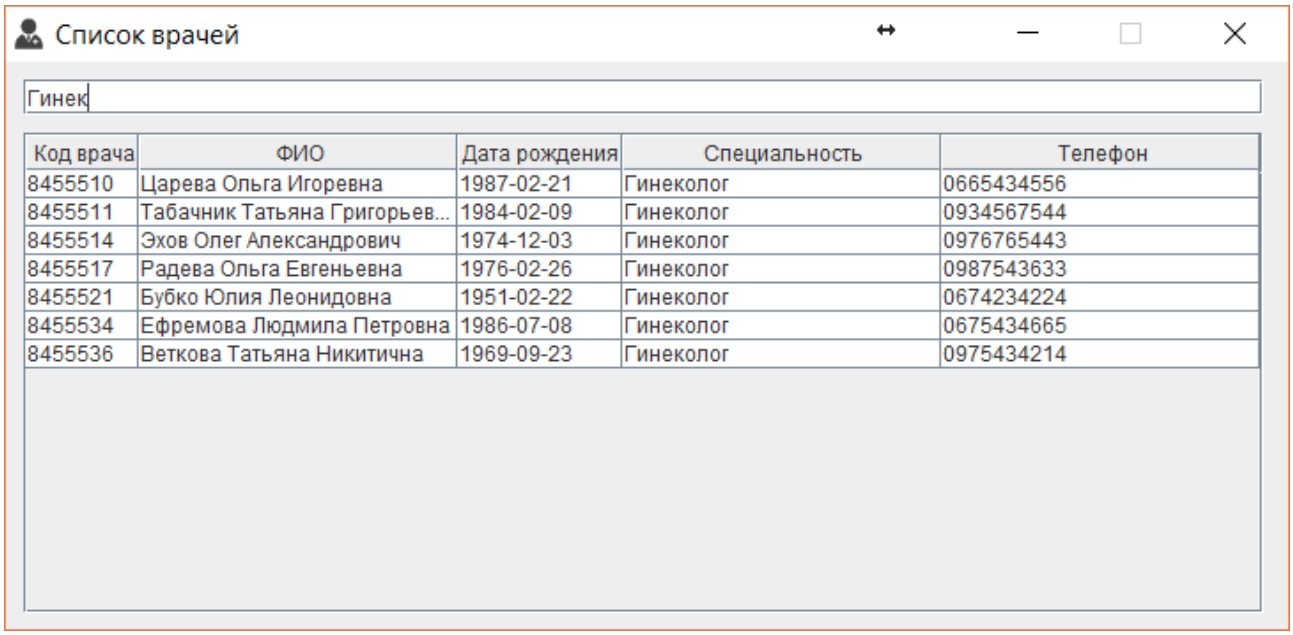

Рис. 2.16. Вікно зі списком пацієнтів




| Код врача | ФИО | Дата рождения | Специальность | Телефон |
|---|---|---|---|---|
| 8455509 | Острянко Ирина Степановна | 1982-09-07 | Эндокринолог | 0664578543 |
| 8455510 | Царева Ольга Игоревна | 1987-02-21 | Гинеколог | 0665434556 |
| 8455511 | Табачник Татьяна Григорье... | 1984-02-09 | Гинеколог | 0934567544 |
| 8455513 | Хлебова Арина Петровна | 1977-08-10 | Хирург | 0675432357 |
| 8455514 | Эхов Олег Александрович | 1974-12-03 | Гинеколог | 0976765443 |
| 8455515 | Лермантов Сергей Павлович | 1966-07-31 | Невролог | 0675442554 |
| 8455516 | Шакаров Сергей Валентино... | 1987-04-12 | Уролог | 0675432113 |
| 8455517 | Радева Ольга Евгеньевна | 1976-02-26 | Гинеколог | 0987543633 |
| 8455518 | Жидко Дмитрий Денисович | 1966-09-16 | Педиатр | 0975464223 |
| 8455519 | Ильницкий Арсен Игоревич | 1954-11-24 | Эндокринолог | 0664322457 |
| 8455520 | Заяц Олег Степанович | 1965-04-10 | ЛОР | 0954532456 |
| 8455521 | Бубко Юлия Леонидовна | 1951-02-22 | Гинеколог | 0674234224 |
| 8455522 | Фархадин Ицгун Арсентович | 1950-12-24 | Эндокринолог | 0667777453 |
| 8455523 | Георгиев Артем Юрьевич | 1959-11-16 | ЛОР | 0975456543 |
| 8455524 | Яковенко Олег Иванович | 1965-04-18 | Уролог | 0675454323 |

Рис. 2.17. Вікно зі списком лікарів

В кожному вікні може здійснюватися пошук по довільному параметру.




| Код врача | ФИО | Дата рождения | Специальность | Телефон |
|---|---|---|---|---|
| 8455510 | Царева Ольга Игоревна | 1987-02-21 | Гинеколог | 0665434556 |
| 8455511 | Табачник Татьяна Григорьев... | 1984-02-09 | Гинеколог | 0934567544 |
| 8455514 | Эхов Олег Александрович | 1974-12-03 | Гинеколог | 0976765443 |
| 8455517 | Радева Ольга Евгеньевна | 1976-02-26 | Гинеколог | 0987543633 |
| 8455521 | Бубко Юлия Леонидовна | 1951-02-22 | Гинеколог | 0674234224 |
| 8455534 | Ефремова Людмила Петровна | 1986-07-08 | Гинеколог | 0675434665 |
| 8455536 | Веткова Татьяна Никитична | 1969-09-23 | Гинеколог | 0975434214 |

Рис. 2.18. Пошук лікарів-гінекологів

На головній формі також є можливість детального перегляду інформації про виклик, лікаря або пацієнта (рис.2.19.-2.20.). Щоб викликати відповідне вікно необхідно лише натиснути на необхідну комірку.

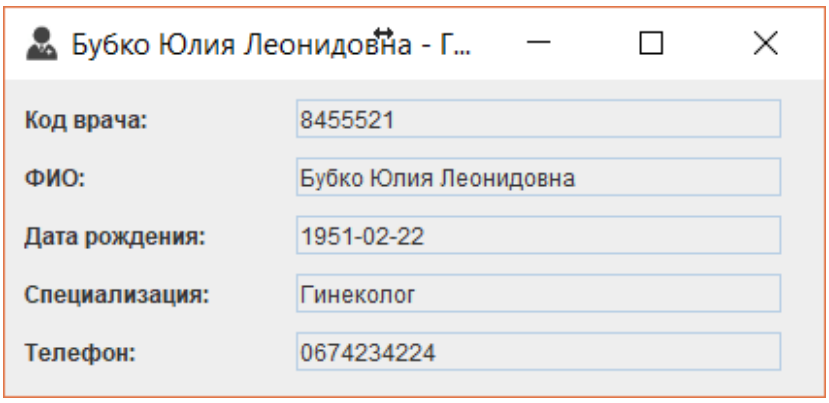



Рис. 2.19. Інформация про лікаря

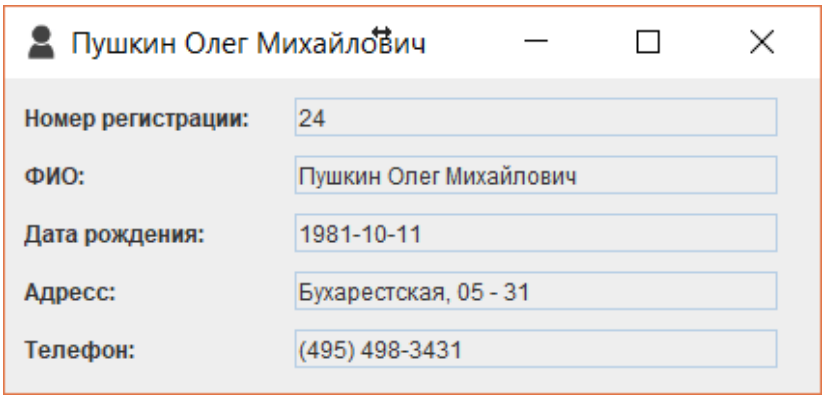



Рис. 2.20. Інформация про пацієнта

Вихід з програми здійснюється закриттям головного вікна, вибором відповідного пункту в меню або натисканням на кнопку виходу на панелі інструментів.

# РОЗДІЛ 3
# ЕКОНОМІЧНИЙ РОЗДІЛ

## 3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1250;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 60 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 13 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де $t_o$ - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

$t_u$ - витрати праці на дослідження алгоритму рішення задачі;

$t_a$ - витрати праці на розробку блок-схеми алгоритму;

$t_n$ - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де $q$ - передбачуване число операторів (1200);

$C$ - коефіцієнт складності програми (1,6);

$p$ - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1250 \cdot (1 + 0{,}05) = 2016$$

Витрати праці на вивчення опису задачі $t_u$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,}$$

де $B$ - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

$k$ - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% ($B$ = 1.2). З урахуванням коефіцієнта кваліфікації $k$ = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (2016 \cdot 1{,}2) / (75 \cdot 1{,}2) = 26{,}88 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де $Q$ – умовне число операторів програми;

$k$ – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 2016 / (20 \cdot 1{,}2) = 84 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 2016 / (25 \cdot 1{,}2) = 67{,}2 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 2016 / (5 \cdot 1{,}2) = 336 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^{k}_{отл} = 1,5 \cdot 336 = 504 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 2016 / (18 \cdot 1,2) = 93,33 \text{ людино-годин.}$$

$$t_{до} = 0,75 \cdot 93,33 = 70 \text{ людино-годин.}$$

$$t_{д} = 93,33 + 70 = 163,33 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 26,88 + 84 + 67,2 + 336 + 163,33 = 727,41 \text{ людино-годин.}$$

### 3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: $t$ - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 60 грн / год, отримуємо:

$$З_{ЗП} = 727{,}41 \cdot 60 = 43\,644{,}6 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (13 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 336 \cdot 13 = 6368 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 43644{,}6 + 6368 = 50\,012{,}6 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де $B_k$ - число виконавців (дорівнює 1);

$F_p$ - місячний фонд робочого часу (при 40 годинному робочому тижні $F_p$=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 727{,}41 / 1 \cdot 176 \approx 5 \text{ міс.}$$

**Висновок:** програмне забезпечення призначене для рєєстрації викликів лікаря до дому. Вартість даного програмного забезпечення становить 50 тис. грн. і не вимагає додаткових витрат як при розробці програми. Очікуваний час розробки становить 5 місяців. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

# ВИСНОВКИ

В результаті виконання кваліфікаційної роботи була розроблена інформаційна система «Рєєстратура» лікувального закладу. Розроблена система реалізована з використанням СУБД MySQL та написана мовою програмування Java.

У роботі над системаю була розроблена структура таблиць бази даних, яка надалі була заповнена випадковими лікарями і пацієнтами.

Для наочного уявлення отриманих даних аналізу були розроблені та реалізовані Hibernate-запити, використання яких дозволяє отримати дані з БД у вигляді об'єктів, описаних в mapping-класах.

При впровадженні даної системи в роботі лікувального закладу може бути підвищена ефективність ведення, зберігання і обробки даних про виклики лікарів додому.

Основним призначенням системи, що розробляється є надання можливості ведення реєстрації викликів лікаря, зберігання та обробки даної інформаціїї. Дана система може бути використана як додаткова система рєєстрації викликів лікаря, може бути використана як альтернативна система, що має можливість працювати стаціонарно, офф-лайн.

Також у кваліфікаційній роботі було визначено трудомісткість розробленої підсистеми 727 люд-год, проведений підрахунок вартості роботи по створенню програми 50012 грн та розраховано час на її створення 5 місяців.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Герберт Шилдт. Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. - М.: «Вильямс», 2015. - 1376 с.
2. Брюс Эккель. Философия Java; 2015. - 1168 с.
3. Барри Берд. Java 8 для чайников = Java For Dummies, 6th edition. - М .: «Диалектика», 2015. – 400 с.
4. Джеймс Гослинг, Билл Джой, Гай Стил, Гилад Брача, Алекс Бакли. Язык программирования Java SE 8. Подробное описание, 5-е издание = The Java Language Specification, Java SE 8 Edition (5th Edition) (Java Series). — М.: «Вильямс», 2015. - 672 с.
5. Патрик Нимейер, Дэниэл Леук. Программирование на Java. Исчерпывающее руководство для профессионалов; 2014. – 1216 с.
6. Джошуа Блох. Java. Эффективное программирование; 2014. – 440 с.
7. Роберт Седжвик, Кевин Уэйн. Алгоритмы на Java; 2013. – 843 с.
8. Mark Matthews, Jim Cole, Joseph D. Gradecki. MySQL and Java Developer's Guide; 2003. – 432 с.
9. Боуман, С.Эмерсон, М.Дарновски. Практическое руководство SQL; 2000. – 321 с.
10. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. SQL. Полное руководство; 2015. – 959 с.
11. Ларри Ульман. MySQL. Руководство по изучению языка; 2004. – 352 с.
12. Д. Кренке. Теория и практика построения баз данных; 2005. – 864 с.
13. Леон Аткинсон. MySQL. Библиотека профессионала; 2002. – 624 с.
14. Marc Delisle. Mastering phpMyAdmin 3.1 for Effective MySQL Management; 2009. – 352 с.
15. Andy Oppel. Databases A Beginner's Guide; 2009. – 496 с.
16. Советов Б.Я., Цехановский В.В., Чертовской В.Д. Базы данных. Теория и практика; 2014. – 464с.

17. Базы данных: основные понятия [Электрон. ресурс]. – Способ доступа: URL: http://www.webmasterwiki.ru/MySQL.

18. Документация по MySQL [Электрон. ресурс]. – Способ доступа: URL: http://www.sql.ru/docs/mysql/rus_ref/.

19. Apache: установка и настройка веб-сервера [Электрон. ресурс]. – Способ доступа: URL: http://www.internet-technologies.ru/articles/article_1747.htmlp/ .

20. Установка и настройка сервера XAMPP на Windows [Электрон. ресурс]. – Способ доступа: URL: http://makegood.ru/tools/xampp/ .

21. XAMPP [Электрон. ресурс]. – Способ доступа: URL: http://htmlbook.ru/webserver/xampp .

22. Java Hibernate [Электрон. ресурс]. – Способ доступа: URL: http://alfalavista.ru/idxfldr/2013-06-18-22-25-47/302-java-hibernate.html

23. Г. Шилдт. Swing руководство для начинающих; 2007. – 705 с.

24. Библиотека Swing [Электрон. ресурс]. – Способ доступа: URL: http://java-online.ru/libs-swing.xhtml.

25. Java Swing Tutorial [Электрон. ресурс]. – Способ доступа: URL: https://www.javatpoint.com/java-swing.

**ДОДАТОК А**

## КОД ПРОГРАМИ

### Beans.Doctors.java

```
package Beans;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import java.sql.Date;

@Entity
public class Doctors {
    private int docCode;
    private String fio;
    private Date dateOfBirth;
    private String spec;
    private String phones;

    @Id
    @Column(name = "docCode", nullable = false)
    public int getDocCode() {
        return docCode;
    }

    public void setDocCode(int docCode) {
        this.docCode = docCode;
    }

    @Basic
    @Column(name = "FIO", nullable = false, length = 45)
    public String getFio() {
        return fio;
    }

    public void setFio(String fio) {
        this.fio = fio;
    }

    @Basic
    @Column(name = "dateOfBirth", nullable = false)
    public Date getDateOfBirth() {
```

```
        return dateOfBirth;
    }

    public void setDateOfBirth(Date dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    @Basic
    @Column(name = "spec", nullable = false, length = 45)
    public String getSpec() {
        return spec;
    }

    public void setSpec(String spec) {
        this.spec = spec;
    }

    @Basic
    @Column(name = "phones", nullable = false, length = 45)
    public String getPhones() {
        return phones;
    }

    public void setPhones(String phones) {
        this.phones = phones;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Doctors doctors = (Doctors) o;

        if (docCode != doctors.docCode) return false;
        if (fio != null ? !fio.equals(doctors.fio) : doctors.fio != null) return false;
        if (dateOfBirth != null ? !dateOfBirth.equals(doctors.dateOfBirth) :
doctors.dateOfBirth != null) return false;
        if (spec != null ? !spec.equals(doctors.spec) : doctors.spec != null) return
false;
        if (phones != null ? !phones.equals(doctors.phones) : doctors.phones !=
null) return false;

        return true;
    }
```

```
    @Override
    public int hashCode() {
        int result = docCode;
        result = 31 * result + (fio != null ? fio.hashCode() : 0);
        result = 31 * result + (dateOfBirth != null ? dateOfBirth.hashCode() : 0);
        result = 31 * result + (spec != null ? spec.hashCode() : 0);
        result = 31 * result + (phones != null ? phones.hashCode() : 0);
        return result;
    }
}
```

**Beans.Patients.java**

```
package Beans;

import javax.persistence.*;
import java.sql.Date;

/**
 * Created by dimak on 18.06.2017.
 */
@Entity
public class Patients {
    private int regNum;
    private String fio;
    private Date dateOfBirth;
    private String address;
    private String phones;

    @Id
    @Column(name = "regNum", nullable = false)
    public int getRegNum() {
        return regNum;
    }

    public void setRegNum(int regNum) {
        this.regNum = regNum;
    }

    @Basic
    @Column(name = "FIO", unique = true, nullable = false, length = 45)
    public String getFio() {
        return fio;
    }
```

```
    public void setFio(String fio) {
        this.fio = fio;
    }

    @Basic
    @Column(name = "dateOfBirth", nullable = false)
    public Date getDateOfBirth() {
        return dateOfBirth;
    }

    public void setDateOfBirth(Date dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    @Basic
    @Column(name = "address", nullable = false, length = 65)
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Basic
    @Column(name = "phones", nullable = false, length = 45)
    public String getPhones() {
        return phones;
    }

    public void setPhones(String phones) {
        this.phones = phones;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Patients patients = (Patients) o;

        if (regNum != patients.regNum) return false;
        if (fio != null ? !fio.equals(patients.fio) : patients.fio != null) return false;
        if (dateOfBirth != null ? !dateOfBirth.equals(patients.dateOfBirth) :
patients.dateOfBirth != null)
```

```
            return false;
        if (address != null ? !address.equals(patients.address) : patients.address !=
null) return false;
        if (phones != null ? !phones.equals(patients.phones) : patients.phones !=
null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = regNum;
        result = 31 * result + (fio != null ? fio.hashCode() : 0);
        result = 31 * result + (dateOfBirth != null ? dateOfBirth.hashCode() : 0);
        result = 31 * result + (address != null ? address.hashCode() : 0);
        result = 31 * result + (phones != null ? phones.hashCode() : 0);
        return result;
    }
}
```

**Beans.Receptions.java**

```
package Beans;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import java.sql.Date;

@Entity
public class Receptions {
    private int recNum;
    private String docFio;
    private String patFio;
    private Date recDate;
    private String notes;

    @Id
    @Column(name = "recNum", nullable = false)
    public int getRecNum() {
        return recNum;
    }

    public void setRecNum(int recNum) {
        this.recNum = recNum;
```

```
}

@Basic
@Column(name = "docFio", nullable = false)
public String getDocFio() {
    return docFio;
}

public void setDocFio(String docFio) {
    this.docFio = docFio;
}

@Basic
@Column(name = "patFio", nullable = false)
public String getPatFio() {
    return patFio;
}

public void setPatFio(String patFio) {
    this.patFio = patFio;
}

@Basic
@Column(name = "recDate", nullable = false)
public Date getRecDate() {
    return recDate;
}

public void setRecDate(Date recDate) {
    this.recDate = recDate;
}

@Basic
@Column(name = "notes", nullable = false, length = -1)
public String getNotes() {
    return notes;
}

public void setNotes(String notes) {
    this.notes = notes;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;

        Receptions that = (Receptions) o;

        if (recNum != that.recNum) return false;
        if (recDate != null ? !recDate.equals(that.recDate) : that.recDate != null) return false;
        if (notes != null ? !notes.equals(that.notes) : that.notes != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = recNum;
        result = 31 * result + (recDate != null ? recDate.hashCode() : 0);
        result = 31 * result + (notes != null ? notes.hashCode() : 0);
        return result;
    }
}
```

**GUI.DoctorsInfoForm.DoctorsInfoFrame.java**

```
package GUI.DoctorsInfoForm;

import Beans.Doctors;

import javax.swing.*;

public class DoctorInfoFrame extends JFrame {

    public DoctorInfoFrame(Doctors doc){
        setTitle(doc.getFio() + " - " + doc.getSpec());
        setBounds(700, 400, 440, 210);
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        setIconImage(new ImageIcon(
                DoctorInfoFrame.class.getResource("/resources/images/doc-icon.png")).getImage());

        add(new DoctorInfoPanel(doc));

        setVisible(true);
    }
}
```

**GUI.DoctorsInfoForm.DoctorsInfoPanel.java**

```
package GUI.DoctorsInfoForm;

import Beans.Doctors;

import javax.swing.*;

public class DoctorInfoPanel extends JPanel {

    public DoctorInfoPanel(Doctors doc) {
        setLayout(null);

        JLabel regnumLbl = new JLabel("Код врача: ");
        regnumLbl.setBounds(10, 10, 200, 20);
        add(regnumLbl);
        JTextField regnumText = new JTextField(Integer.toString(doc.getDocCode()));
        regnumText.setEditable(false);
        regnumText.setBounds(150, 10, 250, 20);
        add(regnumText);

        JLabel fioLbl = new JLabel("ФИО: ");
        fioLbl.setBounds(10, 40, 200, 20);
        add(fioLbl);
        JTextField fioText = new JTextField(doc.getFio());
        fioText.setEditable(false);
        fioText.setBounds(150, 40, 250, 20);
        add(fioText);

        JLabel birthLbl = new JLabel("Дата рождения: ");
        birthLbl.setBounds(10, 70, 200, 20);
        add(birthLbl);
        JTextField birthText = new JTextField(doc.getDateOfBirth().toString());
        birthText.setEditable(false);
        birthText.setBounds(150, 70, 250, 20);
        add(birthText);

        JLabel addressLbl = new JLabel("Специализация: ");
        addressLbl.setBounds(10, 100, 200, 20);
        add(addressLbl);
        JTextField addressText = new JTextField(doc.getSpec());
        addressText.setEditable(false);
        addressText.setBounds(150, 100, 250, 20);
        add(addressText);

        JLabel phonesLbl = new JLabel("Телефон: ");
```

```
            phonesLbl.setBounds(10, 130, 200, 20);
            add(phonesLbl);
            JTextField phonesText = new JTextField(doc.getPhones());
            phonesText.setEditable(false);
            phonesText.setBounds(150, 130, 250, 20);
            add(phonesText);
        }
    }
```

**GUI.PatientsInfoForm.PatientsInfoFrame.java**

```
    package GUI.PatientInfoForm;

    import Beans.Patients;

    import javax.swing.*;

    public class PatientInfoFrame extends JFrame {

        public PatientInfoFrame(Patients pat){
            setTitle(pat.getFio());
            setBounds(700, 400, 440, 210);
            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
            setIconImage(new ImageIcon(
                    PatientInfoFrame.class.getResource("/resources/images/pat-
icon.png")).getImage());

            add(new PatientInfoPanel(pat));

            setVisible(true);
        }
    }
```

**GUI.PatientsInfoForm.PatientsInfoPanel.java**

```
    package GUI.PatientInfoForm;

    import Beans.Patients;

    import javax.swing.*;

    public class PatientInfoPanel extends JPanel {

        public PatientInfoPanel(Patients pat) {
            setLayout(null);

            JLabel regnumLbl = new JLabel("Номер регистрации: ");
            regnumLbl.setBounds(10, 10, 200, 20);
```

```
            add(regnumLbl);
            JTextField regnumText = new
JTextField(Integer.toString(pat.getRegNum()));
            regnumText.setEditable(false);
            regnumText.setBounds(150, 10, 250, 20);
            add(regnumText);

            JLabel fioLbl = new JLabel("ФИО: ");
            fioLbl.setBounds(10, 40, 200, 20);
            add(fioLbl);
            JTextField fioText = new JTextField(pat.getFio());
            fioText.setEditable(false);
            fioText.setBounds(150, 40, 250, 20);
            add(fioText);

            JLabel birthLbl = new JLabel("Дата рождения: ");
            birthLbl.setBounds(10, 70, 200, 20);
            add(birthLbl);
            JTextField birthText = new JTextField(pat.getDateOfBirth().toString());
            birthText.setEditable(false);
            birthText.setBounds(150, 70, 250, 20);
            add(birthText);

            JLabel addressLbl = new JLabel("Адресс: ");
            addressLbl.setBounds(10, 100, 200, 20);
            add(addressLbl);
            JTextField addressText = new JTextField(pat.getAddress());
            addressText.setEditable(false);
            addressText.setBounds(150, 100, 250, 20);
            add(addressText);

            JLabel phonesLbl = new JLabel("Телефон: ");
            phonesLbl.setBounds(10, 130, 200, 20);
            add(phonesLbl);
            JTextField phonesText = new JTextField(pat.getPhones());
            phonesText.setEditable(false);
            phonesText.setBounds(150, 130, 250, 20);
            add(phonesText);
        }
    }
```

**GUI.ReceptionsInfoForm.ReceptionsInfoFrame.java**

```
package GUI.ReceptionInfoForm;

import Beans.Receptions;
```

```
import javax.swing.*;

public class ReceptionInfoFrame extends JFrame {

    public ReceptionInfoFrame(Receptions rec){
        setTitle("Вызов №" + rec.getRecNum());
        setBounds(700, 400, 440, 390);
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        setIconImage(new ImageIcon(
                ReceptionInfoFrame.class.getResource("/resources/images/add-
icon.png")).getImage());

        add(new ReceptionInfoPanel(rec));

        setVisible(true);
    }
}
```

**GUI.ReceptionsInfoForm.ReceptionsInfoPanel.java**

```
package GUI.ReceptionInfoForm;

import Beans.Receptions;

import javax.swing.*;

public class ReceptionInfoPanel extends JPanel {

    public ReceptionInfoPanel(Receptions rec) {
        setLayout(null);

        JLabel recNumLbl = new JLabel("Номер вызова: ");
        recNumLbl.setBounds(10, 10, 200, 20);
        add(recNumLbl);
        JTextField recNumText = new
JTextField(Integer.toString(rec.getRecNum()));
        recNumText.setEditable(false);
        recNumText.setBounds(150, 10, 250, 20);
        add(recNumText);

        JLabel docFioLbl = new JLabel("ФИО врача: ");
        docFioLbl.setBounds(10, 40, 200, 20);
        add(docFioLbl);
        JTextField docFioText = new JTextField(rec.getDocFio());
        docFioText.setEditable(false);
```

```
        docFioText.setBounds(150, 40, 250, 20);
        add(docFioText);

        JLabel patFioLbl = new JLabel("ФИО пациента: ");
        patFioLbl.setBounds(10, 70, 200, 20);
        add(patFioLbl);
        JTextField patFioText = new JTextField(rec.getPatFio());
        patFioText.setEditable(false);
        patFioText.setBounds(150, 70, 250, 20);
        add(patFioText);

        JLabel dateLbl = new JLabel("Дата вызова: ");
        dateLbl.setBounds(10, 100, 200, 20);
        add(dateLbl);
        JTextField dateText = new JTextField(rec.getRecDate().toString());
        dateText.setEditable(false);
        dateText.setBounds(150, 100, 250, 20);
        add(dateText);

        JLabel notesLbl = new JLabel("Примечания: ");
        notesLbl.setBounds(10, 130, 200, 20);
        add(notesLbl);
        JTextArea notesText = new JTextArea(rec.getNotes());
        notesText.setEditable(false);
        notesText.setLineWrap(true);
        notesText.setWrapStyleWord(true);
        JScrollPane scr = new JScrollPane(notesText);
        scr.setBounds(150, 130, 250, 200);
        add(scr);
    }
}
```

**GUI.AddReceptionDialog.java**

```
package GUI;

import Beans.Doctors;
import Beans.Patients;
import Beans.Receptions;
import Logic.ReceptionsTableModel;
import org.hibernate.Session;

import javax.swing.*;
import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
```

```
import java.util.Calendar;

public class AddReceptionDialog extends JDialog {

    public AddReceptionDialog(Session session, ReceptionsTableModel tm) {
        setIconImage(new ImageIcon(
                AddReceptionDialog.class.getResource("/resources/images/add-
icon.png")).getImage());

        ArrayList<Doctors> docs = new ArrayList<>(
                session.createQuery("FROM Doctors ORDER BY FIO").list());
        ArrayList<Patients> pats = new ArrayList<>(
                session.createQuery("FROM Patients ORDER BY FIO").list());

        setLayout(null);
        setTitle("Добавить вызов");
        setBounds(675, 375, 450, 200);

        JLabel docFIO = new JLabel("ФИО врача");
        docFIO.setBounds(10, 10, 100, 20);
        add(docFIO);
        JComboBox docBox = new JComboBox();
        docBox.setBounds(110, 10, 300, 20);
        for(Doctors doc : docs) {
            docBox.addItem(doc.getFio());
        }
        add(docBox);

        JLabel patFIO = new JLabel("ФИО пациента");
        patFIO.setBounds(10, 40, 100, 20);
        add(patFIO);
        JComboBox patBox = new JComboBox();
        patBox.setBounds(110, 40, 300, 20);
        for(Patients pat : pats) {
            patBox.addItem(pat.getFio());
        }
        add(patBox);

        JLabel recDate = new JLabel("Дата приема");
        recDate.setBounds(10, 70, 100, 20);
        add(recDate);
        JTextField dateText = new JTextField(
                new Date(Calendar.getInstance().getTime().getTime()).toString());
        dateText.setBounds(110, 70, 100, 20);
        add(dateText);
```

```
            JButton okBtn = new JButton("OK");
            okBtn.setBounds(10, 100, 100, 20);
            add(okBtn);
            okBtn.addActionListener(e -> {
               Date date = null;
               Receptions rec = null;
               try {
                  SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-
dd");
                  java.util.Date utilDate = format.parse(dateText.getText());
                  java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime());

                  rec = new Receptions();
                  rec.setDocFio(docBox.getSelectedItem().toString());
                  rec.setPatFio(patBox.getSelectedItem().toString());
                  rec.setRecDate(sqlDate);
                  rec.setNotes("");

                  session.beginTransaction();
                  session.save(rec);
                  session.getTransaction().commit();
                  session.close();

                  tm.read();
                  tm.fireTableDataChanged();

                  this.dispose();
               } catch (Exception e1) {
                  JOptionPane.showMessageDialog(
                        null, e1.getMessage(), "Ошибка при вставке",
JOptionPane.OK_OPTION);
               }
            });
         }
      }
```

**GUI.DoctorsFrame.java**

```
      package GUI;

      import Logic.DoctorsTableModel;
      import org.hibernate.SessionFactory;

      import javax.swing.*;
      import javax.swing.table.TableRowSorter;
```

```
public class DoctorsFrame extends JFrame {

    public DoctorsFrame(SessionFactory sf) {
        setTitle("Список врачей");
        setBounds(625, 325, 730, 355);
        setIconImage(new ImageIcon(
                DoctorsFrame.class.getResource("/resources/images/doc-
icon.png")).getImage());
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        JPanel panel = new JPanel();
        panel.setLayout(null);
        DoctorsTableModel tm = new DoctorsTableModel(sf);
        JTable tbl = new JTable(tm);
        tbl.getColumnModel().getColumn(0).setPreferredWidth(100);
        tbl.getColumnModel().getColumn(1).setPreferredWidth(300);
        tbl.getColumnModel().getColumn(2).setPreferredWidth(150);
        tbl.getColumnModel().getColumn(3).setPreferredWidth(300);
        tbl.getColumnModel().getColumn(4).setPreferredWidth(300);
        JScrollPane scr = new JScrollPane(tbl);
        scr.setBounds(10, 40, 700, 271);
        panel.add(scr);

        TableRowSorter<DoctorsTableModel>          sorter          =          new
TableRowSorter<>(tm);
        tbl.setRowSorter(sorter);

        JTextField filter = new JTextField();
        filter.setBounds(10, 10, 700, 20);
        add(filter);
        filter.addCaretListener(e -> {
            try {
                sorter.setRowFilter(RowFilter.regexFilter("(?i)" + filter.getText()));
                sorter.setSortKeys(null);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        add(panel);
        setResizable(false);
        setVisible(true);
    }
}
```

**GUI.MainFrame.java**

```
package GUI;

import Logic.HibernateSessionFactory;
import org.hibernate.SessionFactory;

import javax.swing.*;
import java.awt.*;

public class MainFrame extends JFrame {

   public MainFrame() {
      setTitle("История приемов");
      setBounds(600, 300, 730, 400);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
      setIconImage(new ImageIcon(
            MainFrame.class.getResource("/resources/images/main-
icon.png")).getImage());
      setLayout(new BorderLayout());

      SessionFactory sf = HibernateSessionFactory.getSessionFactory();

      MainPanel mp = new MainPanel(sf);
      add(mp, BorderLayout.CENTER);
      setJMenuBar(new Menu(sf, mp.tm));

      add(new ToolBar(sf, mp.tm), BorderLayout.NORTH);

      setResizable(false);
      setVisible(true);
   }
}
```

**GUI.MainPanel.java**

```
package GUI;

import Logic.ReceptionsTableModel;
import org.hibernate.SessionFactory;

import javax.swing.*;
import javax.swing.table.TableRowSorter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
```

```
public class MainPanel extends JPanel {
    ReceptionsTableModel tm = null;

    public MainPanel(SessionFactory sf) {
        setLayout(null);

        tm = new ReceptionsTableModel(sf);
        JTable tbl = new JTable(tm);
        TableRowSorter<ReceptionsTableModel> sorter = new
TableRowSorter<>(tm);
        tbl.setRowSorter(sorter);
        JScrollPane scr = new JScrollPane(tbl);
        scr.setBounds(10, 40, 700, 250);
        add(scr);

        ArrayList<RowSorter.SortKey> keys = new ArrayList<>();
        keys.add(new RowSorter.SortKey(3, SortOrder.DESCENDING));
        keys.add(new RowSorter.SortKey(3, SortOrder.DESCENDING));
        keys.add(new RowSorter.SortKey(0, SortOrder.DESCENDING));
        keys.add(new RowSorter.SortKey(0, SortOrder.DESCENDING));
        sorter.setSortKeys(keys);

        sorter.setSortable(0, false);
        sorter.setSortable(1, false);
        sorter.setSortable(2, false);
        sorter.setSortable(3, false);

        tbl.getColumnModel().getColumn(0).setPreferredWidth(150);
        tbl.getColumnModel().getColumn(1).setPreferredWidth(300);
        tbl.getColumnModel().getColumn(2).setPreferredWidth(300);
        tbl.getColumnModel().getColumn(3).setPreferredWidth(150);
        tbl.addMouseListener(new MouseListener() {
            @Override
            public void mouseClicked(MouseEvent e) {
                int row = tbl.convertRowIndexToModel(tbl.getSelectedRow());
                int col = tbl.getSelectedColumn();
                switch (col){
                    case 0: tm.showRec(row); break;
                    case 1: tm.showDoc(row); break;
                    case 2: tm.showPat(row); break;
                    default: break;
                }
            }

            @Override
```

```
            public void mousePressed(MouseEvent e) {

            }
            @Override
            public void mouseReleased(MouseEvent e) {

            }
            @Override
            public void mouseEntered(MouseEvent e) {

            }
            @Override
            public void mouseExited(MouseEvent e) {

            }
        });

        JTextField filter = new JTextField();
        filter.setBounds(10, 10, 700, 20);
        add(filter);
        filter.addCaretListener(e -> {
            try {
                sorter.setRowFilter(RowFilter.regexFilter("(?i)" + filter.getText()));
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });
    }
}
```

**GUI.Menu.java**

```
package GUI;

import Logic.Actions;
import Logic.ReceptionsTableModel;
import org.hibernate.SessionFactory;

import javax.swing.*;

public class Menu extends JMenuBar {
    public Menu(SessionFactory sf, ReceptionsTableModel tm) {
        Actions actions = new Actions(sf, tm);

        JMenu main = new JMenu("Действия");
        main.add(new JMenuItem(actions.addAction));
```

```
            main.add(new JMenuItem(actions.patAction));
            main.add(new JMenuItem(actions.docAction));
            main.add(new JMenuItem(actions.exitAction));

            add(main);
        }
    }
```

**GUI.PatientsFrame.java**

```
    package GUI;

    import Logic.PatientsTableModel;
    import org.hibernate.SessionFactory;

    import javax.swing.*;
    import javax.swing.table.TableRowSorter;

    public class PatientsFrame extends JFrame {

        public PatientsFrame(SessionFactory sf) {
            setTitle("Список пациентов");
            setBounds(650, 350, 730, 355);
            setIconImage(new ImageIcon(
                    PatientsFrame.class.getResource("/resources/images/pat-
icon.png")).getImage());
            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

            JPanel panel = new JPanel();
            panel.setLayout(null);
            PatientsTableModel tm = new PatientsTableModel(sf);
            JTable tbl = new JTable(tm);
            tbl.getColumnModel().getColumn(0).setPreferredWidth(150);
            tbl.getColumnModel().getColumn(1).setPreferredWidth(300);
            tbl.getColumnModel().getColumn(2).setPreferredWidth(150);
            tbl.getColumnModel().getColumn(3).setPreferredWidth(300);
            tbl.getColumnModel().getColumn(4).setPreferredWidth(300);
            JScrollPane scr = new JScrollPane(tbl);
            scr.setBounds(10, 40, 700, 271);
            panel.add(scr);

            TableRowSorter<PatientsTableModel> sorter = new
TableRowSorter<>(tm);
            tbl.setRowSorter(sorter);

            JTextField filter = new JTextField();
```

```
        filter.setBounds(10, 10, 700, 20);
        add(filter);
        filter.addCaretListener(e -> {
            try {
                sorter.setRowFilter(RowFilter.regexFilter("(?i)" + filter.getText()));
                sorter.setSortKeys(null);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        add(panel);
        setResizable(false);
        setVisible(true);
    }
}
```

**GUI.ToolBar.java**

```
package GUI;

import Logic.Actions;
import Logic.ReceptionsTableModel;
import org.hibernate.SessionFactory;

import javax.swing.*;
import java.awt.*;

public class ToolBar extends JToolBar {

    public ToolBar(SessionFactory sf, ReceptionsTableModel tm) {
        setFloatable(false);

        Actions actions = new Actions(sf, tm);
        add(actions.addAction);
        add(new JSeparator(){
            @Override
            public Dimension getMaximumSize(){
                return new Dimension(1000, 5000);
            }
        });
        add(actions.patAction);
        add(actions.docAction);
        add(new JSeparator());
        add(actions.exitAction);
    }
```

```
}
```

**Logic.Actions.java**

```
package Logic;

import GUI.AddReceptionDialog;
import GUI.DoctorsFrame;
import GUI.PatientsFrame;
import GUI.ToolBar;
import org.hibernate.SessionFactory;

import javax.swing.*;
import java.awt.event.ActionEvent;

public class Actions {

    public Action addAction = null;
    public Action patAction = null;
    public Action docAction = null;
    public Action exitAction = null;

    public Actions(SessionFactory sf, ReceptionsTableModel tm) {
        addAction = new AbstractAction("Добавить вызов",
                new    ImageIcon(ToolBar.class.getResource("/resources/images/add-
icon.png"))) {
            @Override
            public void actionPerformed(ActionEvent e) {
                new AddReceptionDialog(sf.openSession(), tm).setVisible(true);
            }
        };

        patAction = new AbstractAction("Список пациентов",
                new     ImageIcon(ToolBar.class.getResource("/resources/images/pat-
icon.png"))) {
            @Override
            public void actionPerformed(ActionEvent e) {
                new PatientsFrame(sf);
            }
        };

        docAction = new AbstractAction("Список врачей",
                new    ImageIcon(ToolBar.class.getResource("/resources/images/doc-
icon.png"))) {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
                new DoctorsFrame(sf);
            }
        };

        exitAction = new AbstractAction("Выйти",
                new    ImageIcon(ToolBar.class.getResource("/resources/images/exit-
icon.png"))) {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        };
    }
}
```

**Logic.DoctorsTableModel.java**

```
package Logic;

import Beans.Doctors;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import javax.swing.table.AbstractTableModel;
import java.util.ArrayList;

public class DoctorsTableModel extends AbstractTableModel {
    Session session;
    ArrayList<Doctors> docs;

    public DoctorsTableModel(SessionFactory sf) {
        this.session = sf.openSession();
        try {
            docs = new ArrayList<>(session.createQuery("FROM Doctors").list());
        } catch (HibernateException e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
    }

    @Override
    public int getRowCount() {
        return docs.size();
    }
```

```
        @Override
        public String getColumnName(int index)
        {
            String[] names = {"Код врача", "ФИО", "Дата рождения",
"Специальность", "Телефон"};
            return names[index];
        }

        @Override
        public int getColumnCount() {
            return 5;
        }

        @Override
        public Object getValueAt(int rowIndex, int columnIndex) {
            Doctors doc = docs.get(rowIndex);
            switch (columnIndex) {
                case 0: return doc.getDocCode();
                case 1: return doc.getFio();
                case 2: return doc.getDateOfBirth();
                case 3: return doc.getSpec();
                case 4: return doc.getPhones();
                default: return null;
            }
        }
    }
```

**Logic.HibernateSessionFactory.java**

```
    package Logic;

    import org.hibernate.SessionFactory;
    import org.hibernate.boot.MetadataSources;
    import org.hibernate.boot.registry.StandardServiceRegistry;
    import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

    public class HibernateSessionFactory {

        private static SessionFactory sessionFactory = buildSessionFactory();

        protected static SessionFactory buildSessionFactory() {
            // A SessionFactory is set up once for an application!
            final StandardServiceRegistry registry = new
StandardServiceRegistryBuilder()
                    .configure() // configures settings from hibernate.cfg.xml
```

```
                .build();
        try {
            sessionFactory = new MetadataSources( registry ).buildMetadata().buildSessionFactory();
        }
        catch (Exception e) {
            // The registry would be destroyed by the SessionFactory, but we had trouble building the SessionFactory
            // so destroy it manually.
            StandardServiceRegistryBuilder.destroy( registry );

            throw new ExceptionInInitializerError("Initial SessionFactory failed" + e);
        }
        return sessionFactory;
    }


    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        // Close caches and connection pools
        getSessionFactory().close();
    }

}
```

**Logic.PatientsTableModel.java**

```
package Logic;

import Beans.Patients;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import javax.swing.table.AbstractTableModel;
import java.util.ArrayList;

public class PatientsTableModel extends AbstractTableModel {
    Session session;
    ArrayList<Patients> pats;

    public PatientsTableModel(SessionFactory sf) {
```

```
            this.session = sf.openSession();
            try {
                pats = new ArrayList<>(session.createQuery("FROM Patients").list());
            } catch (HibernateException e) {
                e.printStackTrace();
            } finally {
                session.close();
            }
        }

        @Override
        public int getRowCount() {
            return pats.size();
        }

        @Override
        public String getColumnName(int index)
        {
            String[] names = {"Рег. номер", "ФИО", "Дата рождения", "Адресс",
"Телефон"};
            return names[index];
        }

        @Override
        public int getColumnCount() {
            return 5;
        }

        @Override
        public Object getValueAt(int rowIndex, int columnIndex) {
            Patients pat = pats.get(rowIndex);
            switch (columnIndex) {
                case 0: return pat.getRegNum();
                case 1: return pat.getFio();
                case 2: return pat.getDateOfBirth();
                case 3: return pat.getAddress();
                case 4: return pat.getPhones();
                default: return null;
            }
        }


    }
```

**Logic.ReceptionsTableModel.java**

```
package Logic;

import Beans.Doctors;
import Beans.Patients;
import Beans.Receptions;
import GUI.DoctorsInfoForm.DoctorInfoFrame;
import GUI.PatientInfoForm.PatientInfoFrame;
import GUI.ReceptionInfoForm.ReceptionInfoFrame;
import org.hibernate.*;

import javax.swing.table.AbstractTableModel;
import java.util.ArrayList;

public class ReceptionsTableModel extends AbstractTableModel {
    Session session;
    ArrayList<Receptions> recs;

    public ReceptionsTableModel(SessionFactory sf) {
        this.session = sf.openSession();
        read();
    }

    @Override
    public int getRowCount() {
        return recs.size();
    }

    @Override
    public String getColumnName(int index)
    {
        String[] names = {"Номер приема", "ФИО врача", "ФИО пациента",
"Дата приема"};
        return names[index];
    }

    @Override
    public int getColumnCount() {
        return 4;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Receptions rec = recs.get(rowIndex);
        switch (columnIndex) {
            case 0: return rec.getRecNum();
```

```
                case 1: return rec.getDocFio();
                case 2: return rec.getPatFio();
                case 3: return rec.getRecDate();
                default: return null;
            }
        }

        public void read(){
            try {
                recs = (ArrayList<Receptions>) session.createSQLQuery("SELECT *
FROM Receptions").addEntity(Receptions.class).list();
            } catch (HibernateException e) {
                e.printStackTrace();
            }
        }

        public void showPat(int row){
            Receptions rec = recs.get(row);
            Patients pat = (Patients) session.createQuery("FROM Patients WHERE
FIO = :FIO")
                    .setParameter("FIO", rec.getPatFio()).uniqueResult();
            new PatientInfoFrame(pat);
        }

        public void showDoc(int row){
            Receptions rec = recs.get(row);
            Doctors doc = (Doctors) session.createQuery("FROM Doctors WHERE
FIO = :FIO")
                    .setParameter("FIO", rec.getDocFio()).uniqueResult();
            new DoctorInfoFrame(doc);
        }

        public void showRec(int row){
            new ReceptionInfoFrame(recs.get(row));
        }
    }
```

**Main.java**

```
    import GUI.MainFrame;

    public class Main {

        public static void main(String[] args) {
            new MainFrame();
        }
```

```
}
```

```
-- phpMyAdmin SQL Dump
-- version 4.5.1
-- http://www.phpmyadmin.net
--
-- Хост: 127.0.0.1
-- Версия сервера: 10.1.19-MariaDB
-- Версия PHP: 7.0.13

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- База данных: `diplomdb`
--

-- --------------------------------------------------------

--
-- Структура таблицы `doctors`
--
```

```
CREATE TABLE `doctors` (
  `docCode` int(11) NOT NULL,
  `FIO` varchar(45) NOT NULL,
  `dateOfBirth` date NOT NULL,
  `spec` varchar(45) NOT NULL,
  `phones` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

--
-- --------------------------------------------------------

--
-- Структура таблицы `patients`
--

CREATE TABLE `patients` (
  `regNum` int(11) NOT NULL,
  `FIO` varchar(45) NOT NULL,
  `dateOfBirth` date NOT NULL,
  `address` varchar(65) NOT NULL,
  `phones` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

--
-- --------------------------------------------------------

--
-- Структура таблицы `receptions`
--
```

```
CREATE TABLE `receptions` (
  `recNum` int(11) NOT NULL,
  `docFio` varchar(45) NOT NULL,
  `patFio` varchar(45) NOT NULL,
  `recDate` date NOT NULL,
  `notes` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

--
-- Индексы сохранённых таблиц
---
-- Индексы таблицы `doctors`
--
ALTER TABLE `doctors`
  ADD PRIMARY KEY (`docCode`),
  ADD KEY `FIO` (`FIO`);

--
-- Индексы таблицы `patients`
--
ALTER TABLE `patients`
  ADD PRIMARY KEY (`regNum`),
  ADD KEY `FIO` (`FIO`);

--
-- Индексы таблицы `receptions`
--
ALTER TABLE `receptions`
  ADD PRIMARY KEY (`recNum`),
  ADD KEY `regNum` (`patFio`),
```

```
  ADD KEY `patFio` (`patFio`),
  ADD KEY `docFio` (`docFio`);


--
-- AUTO_INCREMENT для сохранённых таблиц
--


--
-- AUTO_INCREMENT для таблицы `patients`
--
ALTER TABLE `patients`
  MODIFY `regNum` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=43;
--
-- AUTO_INCREMENT для таблицы `receptions`
--
ALTER TABLE `receptions`
  MODIFY `recNum` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=35;
--
-- Ограничения внешнего ключа сохраненных таблиц
--


--
-- Ограничения внешнего ключа таблицы `receptions`
--
ALTER TABLE `receptions`
  ADD CONSTRAINT `receptions_ibfk_1` FOREIGN KEY (`docFio`) REFERENCES `doctors` (`FIO`),
```

ADD CONSTRAINT `receptions_ibfk_2` FOREIGN KEY (`patFio`) REFERENCES `patients` (`FIO`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

**ДОДАТОК Б**

**ВІДГУК**
**керівника економічного розділу**
на кваліфікаційну роботу бакалавра на тему:
«Розробка додатку «Рєєстратура» для організації прийомів лікаря з використанням фреймворків Swing та Hibernate»
**студента групи 122-18ск-1 Котко Максима Валерійовича**

**ДОДАТОК В**

# ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

| **Ім’я файлу** | **Опис** |
|---|---|
| Пояснювальні документи | |
| Диплом Котко.doc | Пояснювальна записка до кваліфікаційної роботи. Документ Word. |
| Диплом Котко.pdf | Пояснювальна записка до кваліфікаційної роботи в форматі PDF. |
| Програма | |
| diplom Котко.zip | Архів. Містить коди програми і откомпільовану програму. |
| Презентація | |
| Презентація Котко.ppt | Презентація кваліфікаційної роботи. |