

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ШЕВЧУК ЮРІЙ АНАТОЛІЙОВИЧ

Допускається до захисту:

Завідувач кафедри
інформаційних технологій
канд. техн. наук, доцент

_____ Т.В. Нескородева

« _____ » _____ 2021р.

РОЗРОБКА ДОДАТКУ ДЛЯ ВЕДЕННЯ СТАНУ ЗДОРОВ'Я ПАЦІЄНТА

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:

Бабаков Р.М., доцент
кафедри інформаційних технологій

Оцінка: _____ / _____ / _____

(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

(підпис)

Вінниця – 2021

ЗМІСТ

ВСТУП	2
РОЗДІЛ 1	7
ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯДІ ІСНУЮЧИХ ДОДАТКІВ ЗА ДАНОЮ ТЕМАТИКОЮ	7
1.1 Постановка задачі	7
1.2 Огляд існуючих додатків для ведення стану здоров'я пацієнтів...	7
Висновок до розділу 1	14
РОЗДІЛ 2	15
ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ	15
2.1 Java.....	15
2.2 Android Studio	18
2.3 SQLite	20
Висновок до розділу 2	22
РОЗДІЛ 3	23
РОЗРОБКА ДОДАТКУ	23
3.1 Огляд Android Studio	23
3.2 Розробка дизайну	24
3.3 Огляд SQLite.....	28
3.4 Створюємо базу даних.....	38
3.5 Створюємо об'єкти	39
3.6 Робота з базою даних	40
3.7 Відправка бази даних електронною поштою	48
ВИСНОВКИ.....	57
СПИСКИ ВИКОРИСТАНИХ ПОСИЛАНЬ	59

АННОТАЦІЯ

на бакалаврську роботу студента

4 курсу

спеціальності “Сучасні інформаційні технології
та програмування”

факультету інформаційних і прикладних технологій
Донецького національно університету ім. Василя Стуса
Шевчука Ю.А.

на тему **“Розробка додатку для ведення стану здоров’я пацієнта”**

Ця бакалаврська робота присвячена розробці додатку на Android.

Робота складається зі вступу, трьох розділів, висновку, списку використаних джерел, який складається з 27 найменувань. Робота містить 21 рисуноків.

Загальний обсяг роботи 60 сторінок.

У вступі наведено актуальність додатку та мета бакалаврської роботи.

У першому розділі поставлена задача з розробки додатку та оглядаються існуючі додатки, їх плюси та мінуси.

Другий розділ присвячений вибору та порівнянню інструментів за допомогою яких можна створити додаток.

Третій розділ містить програмну реалізацію додатку.

У висновку підбитий підсумок проведеної роботи.

Ключові слова: інформаційні технології, додаток, додаток Android.

ANNOTATION

for a student's bachelor's thesis

4 courses

specialty "Modern information technology

and programming”

Faculty of Information and Applied Technologies

Donetsk National University. Vasil` Stus

Shcherbak V.V.

on the topic "**Development of an application for managing the patient's health**"

This bachelor's thesis is devoted to the development of bot telegrams. The work consists of an introduction, three sections, a conclusion, a list of sources used, which consists of 27 items. The work contains 21 figures. The total volume of work is 60 pages.

The introduction presents the relevance of the app and the purpose of the bachelor's thesis.

The first section sets the task of developing an app and examines existing apps, their pros and cons.

The second section is devoted to the selection and comparison of tools with which you can create an app.

The third section contains the software implementation of the app.

In conclusion, summarized the work done.

Key words: information technologies, app, android app.

ВСТУП

В сучасному світі майже вся підприємницька діяльність переходить на електронні носії даних, для зберігання інформації. В них є багато переваг над способом документації, який відходить в минуле – паперовим. Серед них можна назвати оперативність в роботі з документами, так як пошук необхідного документи, його копіювання та друк займають секунди. Також це, звичайно, надійність. Цифрові документи не псуються з часом і можуть зберігатись вічно. Серед переваг можна відмітити дистанційний доступ, який можна отримувати навіть дистанційно.

Нажаль, незважаючи на стрімкий розвиток комп'ютеризації по всьому світі, у нас далеко не кожен заклад використовує переваги такого виду зберігання інформації. В тому числі до цих закладів відносяться і лікарні. В маленьких містах і сілах вся документація зберігається в паперовому виді в лікарнях. Це зменшує ефективність функціонування медичинських закладів в таких регіонах, так як процес пошуку потрібних документів сповільнюється, піднімається вірогідність втрати або пошкодження паперових носіїв. Також лікарі витрачають багато часу на заповнення історій пацієнтів та не мають можливості зробити це дистанційно. Це все дуже негативно впливає на охорону здоров'я, тому важливо максимально комп'ютерезувати заклади медицини в нашій країні. В першу чергу для цього встановлюються комп'ютери та підключається інтернет в лікарнях, завантажується необхідне програмне забезпечення. Але навіть в комп'ютерах є певні недоліки, які не пропонують максимальну гнучкість в веденні персональних даних пацієнтів та можливість зв'язатись з іншими лікарями де завгодно. Тому для оптимізації мобільності ведення даних пацієнтів запропоновано використовувати ще один пристрій – телефон. В сучасному світі неможливо уявити своє життя без телефона. Для підтримки зв'язку та зручного функціонування в суспільстві він є достатньо важливим елементом. Але телефон обмежується не лише комунікативних функціоналом. За допомогою нього можна вести таблиці, списки, записувати думки та зберігати їх на

хмарині. Тому можна сказати, що телефон є цілком підходящим варіантом для зберігання та редагування медичних історій пацієнтів, даних про них, списків ліків і т.д.

Так як для збереження інформації в базі даних використовуються майже завжди одні й ті ж інструменти, достатньо просто можна розробити додаток, який буде в якійсь мірі універсальним, так як назви полів дуже легко змінювати та додавати новий. Такий додаток має мати простий функціонал, не перенавантажений непотрібними можливостями.

В даній роботі було розроблено додаток саме для таких цілей. Він повинен бути зручним та інтуїтивним в використанні, містити невеликий аналіз даних та відправляти по електронній пошті таблиці з показниками пацієнтів. Таким чином, можна спростити ведення медичних історій лікарям, а також зробити цей процес більш мобільним, так як для цього не потрібно знаходитись перед комп'ютером, а достатньо лише запустити додаток на телефоні. Додаток буде містити невеликий аналіз даних: небезпечні для здоров'я показники будуть підсвічені червоним кольором, коли безпечні – зеленим. Це спростовує використання додатком, так як достатньо швидко можна побачити, якщо з результатами аналізів щось не так і не потрібно аналізувати їх вручну.

Додаток буде працювати майже на всіх актуальних Android девайсах, так як мінімальне необхідне API для роботи додатку 19. Це також спростить інтеграцію додатку в медичну систему, без потреби оновлювати пристрої в медичних закладах.

Мета бакалаврської роботи полягає в:

- 1). Дослідженні методів ведення медичних історій в лікарнях
- 2). Оптимізації актуальних розробок для підвищення ефективності ведення даних пацієнтів

Об'єктом досліджень є мова програмування Java, а саме – чи можна на ній розробити додаток для ведення медичних показників.

Предметом досліджень є механізм розробки додатку для ведення стану здоров'я пацієнтів на базі середовища розробки Android Studio.

В ході роботи належить вивчити середовище розробки Android Studio, визначити його переваги та недоліки, отримати навички в розробці додатків для пристроїв на Android. Також закріпити навички володіння мовою Java і навчитись вбудовувати в програму базу даних SQLite.



РОЗДІЛ 1

ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯДІ ІСНУЮЧИХ ДОДАТКІВ ЗА ДАНОЮ ТЕМАТИКОЮ

Даний розділ присвячений постановці задачі, а також виборі та опису всіх необхідних інструментів та технологій для розробки додатку ведення стану здоров'я пацієнтів.

1.1 Постановка задачі

Необхідно розробити додаток для ведення показників пацієнтів, які звертаються в певний медичний заклад. Додаток повинен містити наступні ключові елементи:

- Базу даних з пацієнтами та їхніми показниками
- Можливість додавати, видаляти та редагувати існуючі записи
- Можливість відправляти таблицю по електронній пошті
- Аналіз даних
-

1.2 Огляд існуючих додатків для ведення стану здоров'я пацієнтів

Було здійснено огляд існуючих аналогів додатків для ведення стану здоров'я пацієнтів. Пошук було здійснено в додатку Play Market та було знайдено багато варіантів для ведення показників пацієнтів. В цьому розділі ми розглянемо деякі з них.

Медицинская история – один з найпопулярніших додатків для ведення показників пацієнтів в Play Market. Воно дає змогу вести єдину базу даних для всіх досліджень здоров'я пацієнтів. До додатку можуть мати доступ кілька користувачів, що дозволяє надати інформацію своєму лікарю або членам своєї сім'ї. Таким чином, за допомогою цього додатку можна швидко отримувати результати своїх аналізів і не боятись про те, що їх можна втратити або пошкодити. Тобто в лікарню не потрібно носити всю документацію, яка містить історію хвороб, а достатньо лише завантажити цей

додаток і показати вашому лікарю. Цей додаток містить в собі базу даних сканкопій документів з результатами аналізів, які раніше були зроблені в медичних закладах. Це в якійсь мірі і є його головним недоліком, так як окрім сканкопій він більше нічого не може в собі зберігати. Якщо потрібно вручну вписати результати аналізів, в даному додатку це зробити не вийде. Тому потрібно окремо зберігати результати аналізів, що є не дуже зручним. Також, деякі з користувачів скаржаться на затримку в роботі додатку. Тому, можна сказати, що цей додаток не є найкращим варіантом для ведення даних стану здоров'я пацієнтів.

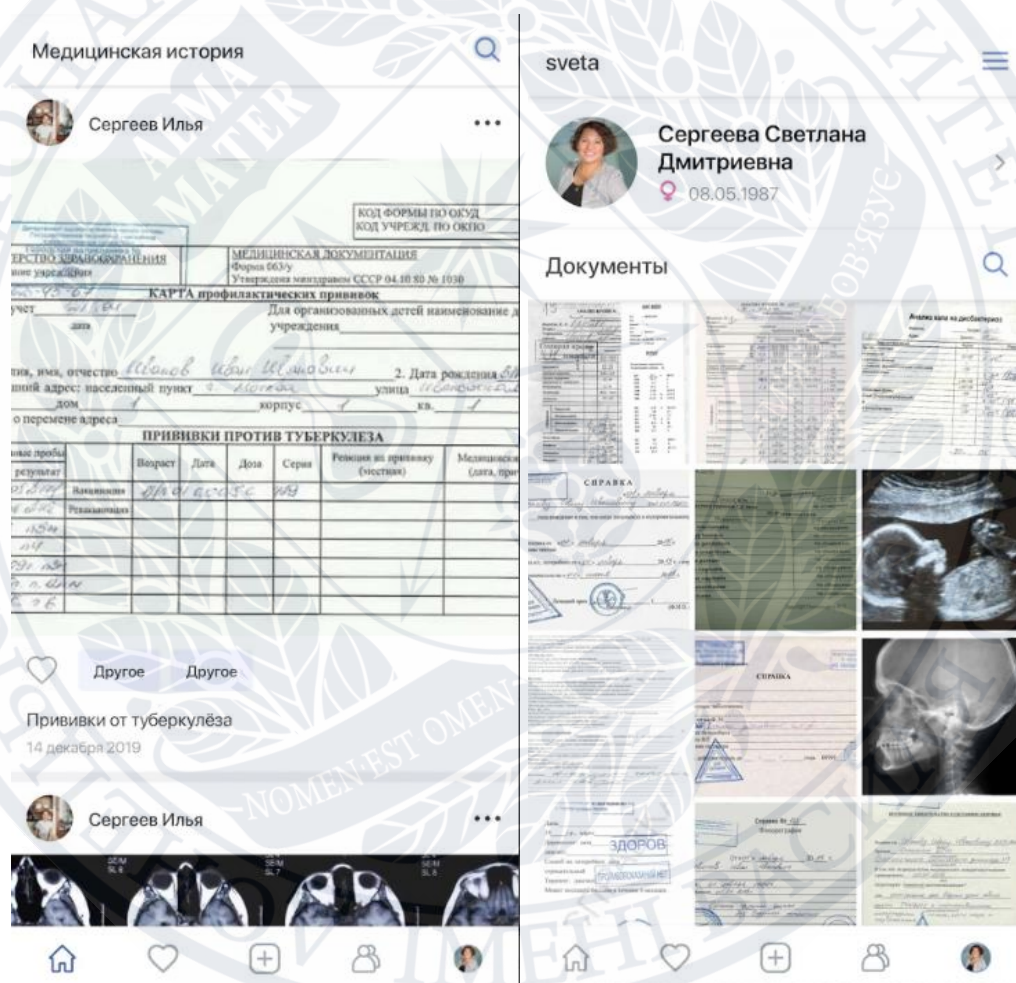


Рисунок 1.1 – Додаток «Медицинская история»

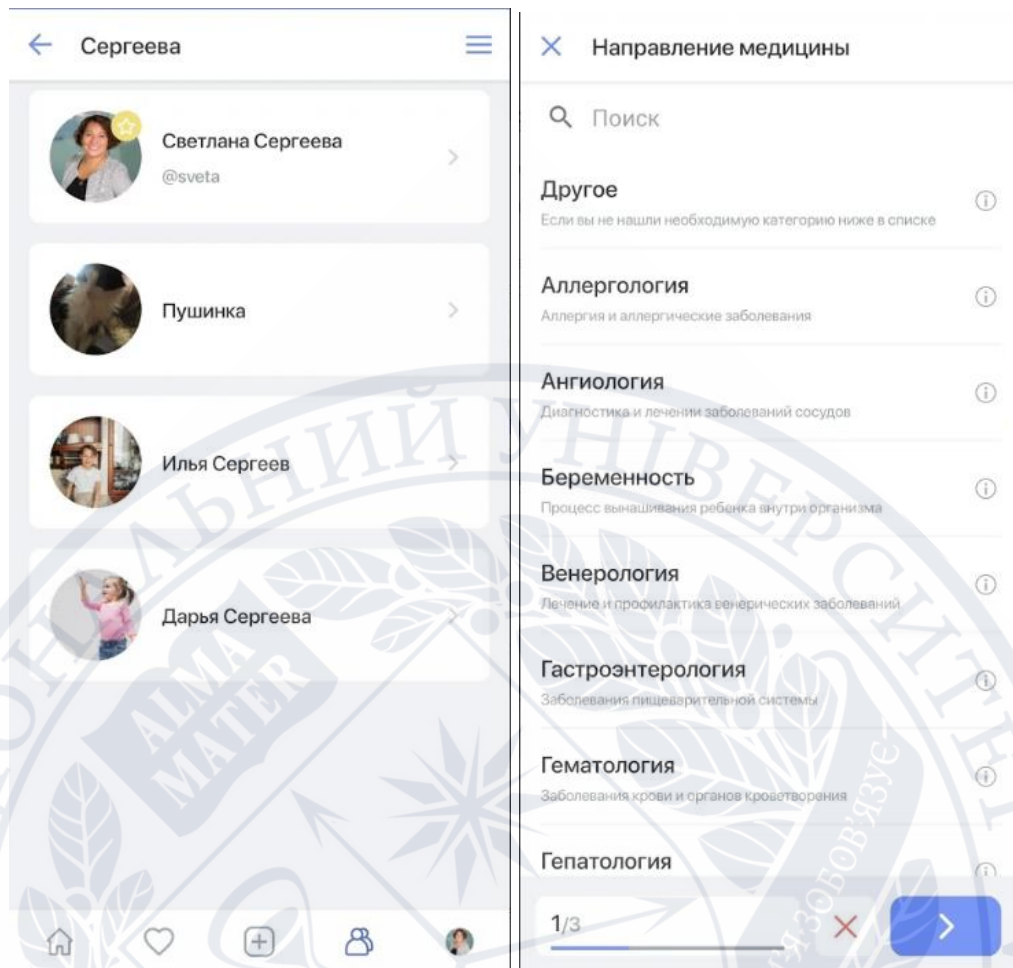


Рисунок 1.2 – Додаток «Медицинская история»

Medical Records – англomовний додаток для ведення показників стану здоров'я пацієнтів.

Переваги:

- Відповідно потреб, можна зберігати медичні дані або в пам'яті пристрою, або в захищеному хмарному сховищі, де ввімкнено синхронізацію даних.
- Підтримує кілька екранів; телефони, маленькі та великі планшети
- Працює в системі Chromebook
- Має резервне копіювання даних
- Аутентифікація імені користувача та пароля
- Можна експортувати медичні дані в аркуш Excel та PDF
- Більшість даних зберігаються за допомогою техніки автозаповнення.

- Зберігає адресу пацієнта: широту, довготу за допомогою карт
- Можливість запуску карти Google, щоб мати змогу доїхати до пацієнта або лікаря
- Звіти про історію хвороб
- Кілька методів пошуку:
 - за іменем або номером телефону
 - за датою відвідування
 - за датою призначення
- Записує відеозапис чи знімає фото для медичної діяльності
- Повноекранний повзунок зображення для перегляду звітів, записаних користувачем
- Повноекранний переглядач відео для відображення зроблених відео.
- Можливість додавання інформації про пацієнта через список контактів пристрою.
- Лікарі можуть використовувати додаток у своїх клініках як інформаційну систему клініки, систему управління клінікою, медичну книжку лікаря пацієнта, мобільний додаток управління охороною здоров'я, медичну документацію пацієнта для відстеження історії хвороб.
- Цей додаток може розглядатися як медичне управління, охорона сім'ї, програма відстеження медичних записів.

Як бачимо, додаток має багато переваг, але його основний недолік полягає в тому, що він повністю англomовний, тому не підходить для використання в українських медичних закладах.



Рисунок 1.3 – Додаток “Medical records”

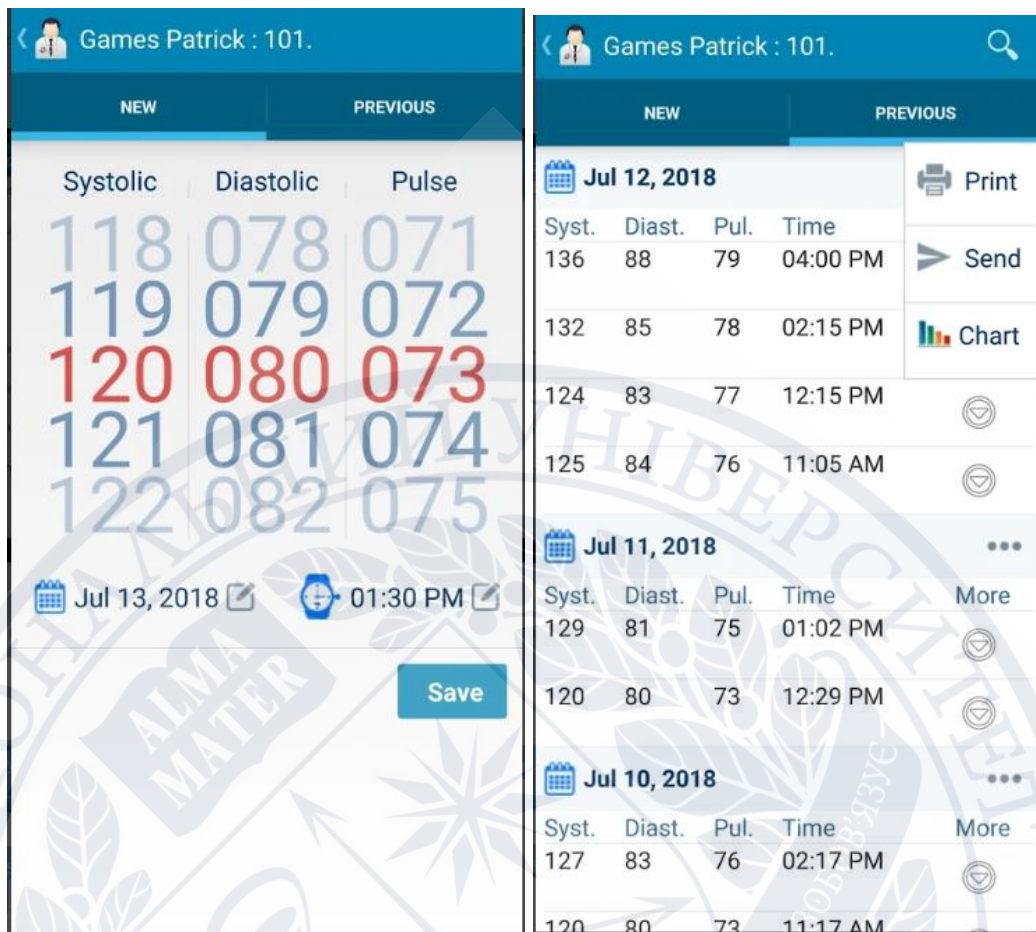


Рисунок 1.4 – Додаток “Medical records”

eHealth - це електронна система реєстрації та ведення взаємин між лікарем і пацієнтом. В майбутньому, система eHealth дасть можливість кожному, швидко отримувати свою медичну інформацію, а лікарям - правильно ставити діагноз з урахуванням цілісної картини здоров'я пацієнта.

Основи створення електронної системи охорони здоров'я eHealth описані в Меморандумах між Міністерством охорони здоров'я, Проектним офісом і широкої ініціативою. Система eHealth складається з державного центрального компонента і зовнішнього приватного компонента.

- Центральний державний компонент є точкою об'єднання медичних інформаційних систем і є невидимим для кінцевих користувачів - лікарів, пацієнтів, управлінців.
- Зовнішній компонент представлений програмними рішеннями - приватними медичними інформаційними системами (MIC), які

приєдналися до системи eHealth. Саме через них кінцеві користувачі будуть співпрацювати з системою eHealth.

The screenshot shows the 'eHealth' web application interface. At the top, it says 'eHealth Тестова медична устан... Головний лікар'. Below this is a title 'Реєстрація закладу в центральному компоненті eHealth' and a warning: 'Попереджаємо про необхідність внесення в ЦБД eHealth інформації щодо закладу згідно з даними ЄДРПОУ'. The form is divided into two main sections: 'Загальна інформація про заклад' and 'Керівник'. The first section contains fields for 'Тип юр. особи' (dropdown), 'Код ЄДРПОУ' (41750202), 'Фактична адреса' (розташування, UA, КИЇВСЬКА, БОРИСПІЛЬСЬКИЙ, село, БЕЗУГЛІВКА, вулиця, Моя вулиця, 5), 'Телефони', 'E-Mail', 'Веб-сайт', 'Код отримувача', and 'Бенефіціар'. The second section, 'Керівник', contains fields for 'Прізвище', 'Ім'я', 'По батькові', 'Стать' (чоловіча), 'Дата народження' (31.12.1969), 'РНОКПП' (2556742259), 'Посада' (ФОП), and 'E-Mail'. At the bottom right are buttons 'Зберегти' and 'Відмінити'. A status bar at the bottom left shows '3.0.1.45 /32/ 06.01.2020'.

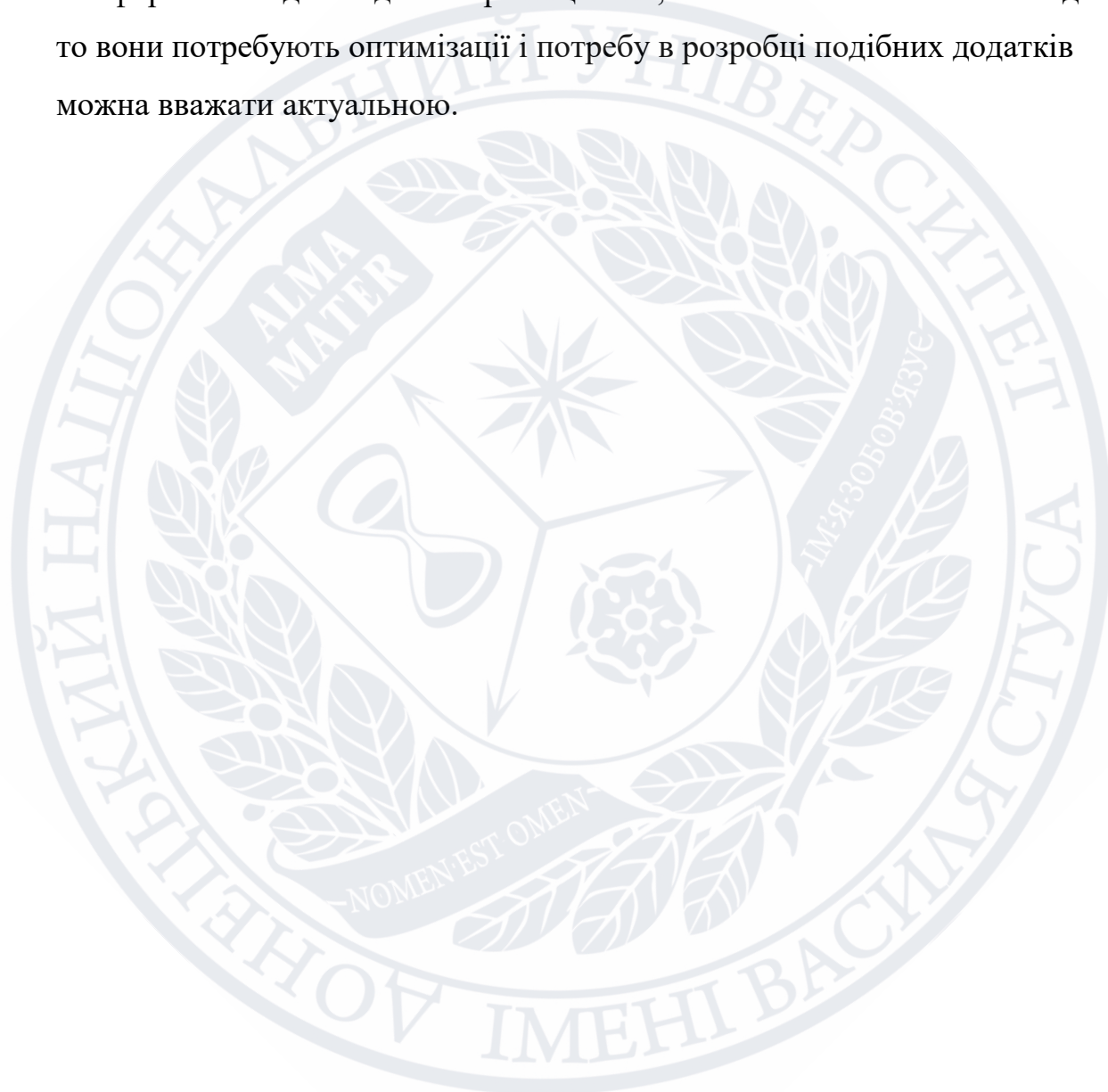
Рисунок 1.5 – Додаток “eHealth”

The screenshot shows the 'eHealth' web application interface for verification. At the top, it says 'eHealth Тестова медична устан... Головний лікар'. Below this is a title 'Реєстрація закладу в центральному компоненті eHealth' and a warning: 'Попереджаємо про необхідність внесення в ЦБД eHealth інформації щодо закладу згідно з даними ЄДРПОУ'. The form is divided into two main sections: 'Загальна інформація про заклад' and 'Керівник'. The first section contains fields for 'Тип юр. особи' (dropdown), 'Код ЄДРПОУ' (41750202), 'Фактична адреса' (розташування, UA, КИЇВСЬКА, БОРИСПІЛЬСЬКИЙ, село, БЕЗУГЛІВКА, вулиця, Моя вулиця, 5), 'Телефони', 'E-Mail', 'Веб-сайт', 'Код отримувача', and 'Бенефіціар'. The second section, 'Керівник', contains fields for 'Прізвище', 'Ім'я', 'По батькові', 'Стать' (чоловіча), 'Дата народження' (31.12.1969), 'РНОКПП' (2556742259), 'Посада' (ФОП), and 'E-Mail'. At the bottom right are buttons 'Зберегти' and 'Відмінити'. A status bar at the bottom left shows '3.0.1.45 /32/ 06.01.2020'.

Рисунок 1.6 – Додаток “eHealth”

Висновок до розділу 1

В даному розділі ми проаналізували додатки для відстеження стану здоров'я пацієнта які знаходяться на платформі Play Market. На даний момент, вони є достатньо багатофункціональними для підтримання комфортного ведення даних про пацієнтів, але кожен з них має свої недоліки, то вони потребують оптимізації і потребу в розробці подібних додатків можна вважати актуальною.



РОЗДІЛ 2

ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ

Даний розділ присвячений інструментам та технологіям, які використовуються для вирішення поставленої задачі. На сьогоднішній день існує дуже велика кількість інструментів для розробки додатку на Android, виходячи з їх характеристик були обрані наступні.

2.1 Java

Java - мова програмування загального призначення. Відноситься до об'єктно-орієнтованих мов програмування.

Java є об'єктно-орієнтованою мовою, відноситься до мов програмування з сильною типізацією.

Творці реалізували принцип WORA: write once, run anywhere або «пиши один раз, запускай всюди». Це означає, що написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлене середовище виконання Java (JRE, Java Runtime Environment).

Це завдання вирішується завдяки компіляції написаного на Java коду в байт-код. Цей формат виконує JVM або віртуальна машина Java. JVM - частина середовища виконання Java (JRE). Віртуальна машина не залежить від платформи.

В Java реалізований механізм управління пам'яттю, який називається складальником сміття або garbage collector. Розробник створює об'єкти, а JRE за допомогою збирача сміття очищає пам'ять, коли об'єкти перестають використовуватися.

Як зазначалося вище, синтаксис мови Java схожий на синтаксис інших С-подібних мов. Ось його деякі особливості:

- чутливість до регістру - ідентифікатори User і user в Java - різні сутності;

- для іменування методів використовується lowerCamelCase. Якщо назва методу складається з одного слова, воно повинно починатися з малої літери. Приклад: firstMethodName ();
- для іменування класів використовується UpperCamelCase. Якщо назва складається з одного слова, воно повинно починатися з великої літери. Приклад: FirstClassName.
- назва файлів програми має точно збігатися з назвою класу з урахуванням чутливості до регістру. Наприклад, якщо клас називається FirstClassName, файл повинен називатися FirstClassName.java;
- ідентифікатори завжди починаються з літери (A-Z, a-z), знака \$ або нижнього підкреслення;

Що пишуть на мові Java: сфери застосування

Вище зазначено, що Java відноситься до мов програмування загального призначення.

За даними компанії Oracle, програми на Java запускаються на 3 млрд девайсів. Це маркетингове повідомлення складно перевірити. Проте Java широко використовується і входить в число найбільш затребуваних мов, це не викликає сумніву.

Наприклад, переважна більшість великих компаній так чи інакше використовують Java. Дуже багато серверних додатків для корпорацій написані на цій мові. Наприклад, мова йде про програми для фінансових організацій, які забезпечують проведення транзакцій, фіксацію торгових операцій.

На Java написано багато веб-додатків. Популярні фреймворки, в тому числі Spring, Stuts, JSP, використовуються для створення різних додатків в інтернеті: від ecommerce-проектів до великих порталів, від освітніх платформ до урядових ресурсів.

Популярна комп'ютерна гра Minecraft написана на Java.

Мобільна розробка - ще одна область використання Java. Цією мовою пишуть програми для пристроїв, що працюють під управлінням ОС Android.

На Java створюють клієнтські програми. Простий і близький розробникам приклад: IDE NetBeans написано на «Джаві».

Також Java застосовується для роботи з Big Data, розробки програм для наукових цілей, наприклад, обробки природних мов, програмування приладів - від побутових девайсів до промислових установок.

Тобто на Java можна писати різні типи додатків: веб, мобільний і десктопний софт, ігри і так далі. Традиційно у цієї мови сильні позиції в промисловому програмуванні, в сегменті великих компаній (т.зв. ентерпрайз).

Підсумок: Java - мова програмування загального призначення. Має C-подібний синтаксис. Використовується для створення додатків в різних областях.



```
1 public class Main {
2     private static final String GREETING_TEMPLATE = "Вітаю, пане %s";
3     private static final String NAME_UNKNOWN = "Невідомий";
4
5     public static void main(String[] args) {
6         if (args.length > 0) {
7             greeting(args[0]);
8         } else {
9             greeting();
10        }
11    }
12
13    private static void greeting() {
14        greeting(NAME_UNKNOWN);
15    }
16
17    /**
18     * Виводить рядок з привітанням (зазвичай, у консоль).
19     * @param name ім'я особи, до якої звернене привітання.
20     */
21    private static void greeting(String name) {
22        System.out.println(String.format(GREETING_TEMPLATE, name));
23    }
24 }
25
```

Рисунок 2.1 – Приклад коду на мові Java

2.2 Android Studio

Android Studio - інтегроване середовище розробки виробництва Google, за допомогою якої розробникам стають доступні інструменти для створення додатків на платформі Android OS. Android Studio можна встановити на Windows, Mac і Linux. Обліковий запис розробника додатків в Google Play App Store коштує \$25. Android Studio створювалася на базі IntelliJ IDEA.

IDE можна завантажити і користуватися безкоштовно. У ній присутні макети для створення UI, з чого зазвичай починається робота над додатком. В Studio містяться інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Серед Android Studio призначена як для невеликих команд розробників мобільних додатків (навіть в кількості однієї людини), або ж великих міжнародних організацій з GIT або іншими подібними системами управління версіями. Досвідчені розробники зможуть вибрати інструменти, які більше підходять для масштабних проектів. Рішення для Android розробляються в Android Studio з використанням Java або C ++. В основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми. Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше опублікувати версію мобільного застосування в Google Play, App Store. Для цього є також підтримка інструментів LINT, ProGuard і App Signing.

За допомогою засобів оцінки продуктивності визначається стан файлу з пакетом прикладних програм. Візуалізація графіків дає можливість дізнатися, чи відповідає додаток орієнтиру Google в 16 мілісекунд. За допомогою інструменту для візуалізації пам'яті розробник дізнається, коли його застосування буде використовувати занадто багато оперативної пам'яті і коли відбудеться «прибирання сміття». Інструменти для аналізу батареї показують, яке навантаження припадає на пристрій.

Android Studio сумісна з платформою Google App Engine для швидкої інтеграції в хмарі нових API і функцій. У середовищі розробки ви знайдете різні API, такі як Google Play, Android Pay і Health. Є підтримка всіх платформ Android, починаючи з Android 1.6. Є варіанти Android, які істотно відрізняються від версії Google Android. Найпопулярніша з них - це Amazon Fire OS. В Android Studio можна створювати APK для цієї ОС. Підтримка Android Studio обмежується онлайн-форумами.

Нові функції з'являються з кожною новою версією Android Studio. На даний момент доступні наступні функції:

- Розширений редактор макетів: WYSIWYG, здатність працювати з UI-компонентами за допомогою Drag-and-Drop, функція попереднього перегляду макета на декількох конфігураціях екрану.
- Збірка додатків, заснована на Gradle.
- Різні види збірок і генерація кількох .apk-файлів
- рефакторинг коду
- Статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій і інше.
- Вбудований ProGuard і утиліта для підписування додатків.
- Шаблони основних макетів і компонентів Android.
- Підтримка розробки додатків для Android Wear і Android TV.
- Вбудована підтримка Google Cloud Platform, яка включає в себе інтеграцію з сервісами Google Cloud Messaging і App Engine.
- Android Studio 2.1 підтримує Android Preview SDK, а це значить, що розробники зможуть почати роботу зі створення програми для нової програмної платформи.
- Нова версія Android Studio 2.1 здатна працювати з оновленим компілятором Jack, а також отримала покращену підтримку Java 8 і вдосконалену функцію Instant Run.
- Починаючи з Platform-tools 23.1.0 для Linux виключно 64-розрядна.

- В Android Studio 3.0 будуть по стандарту включені інструменти мови Kotlin, які основані на JetBrains IDE.

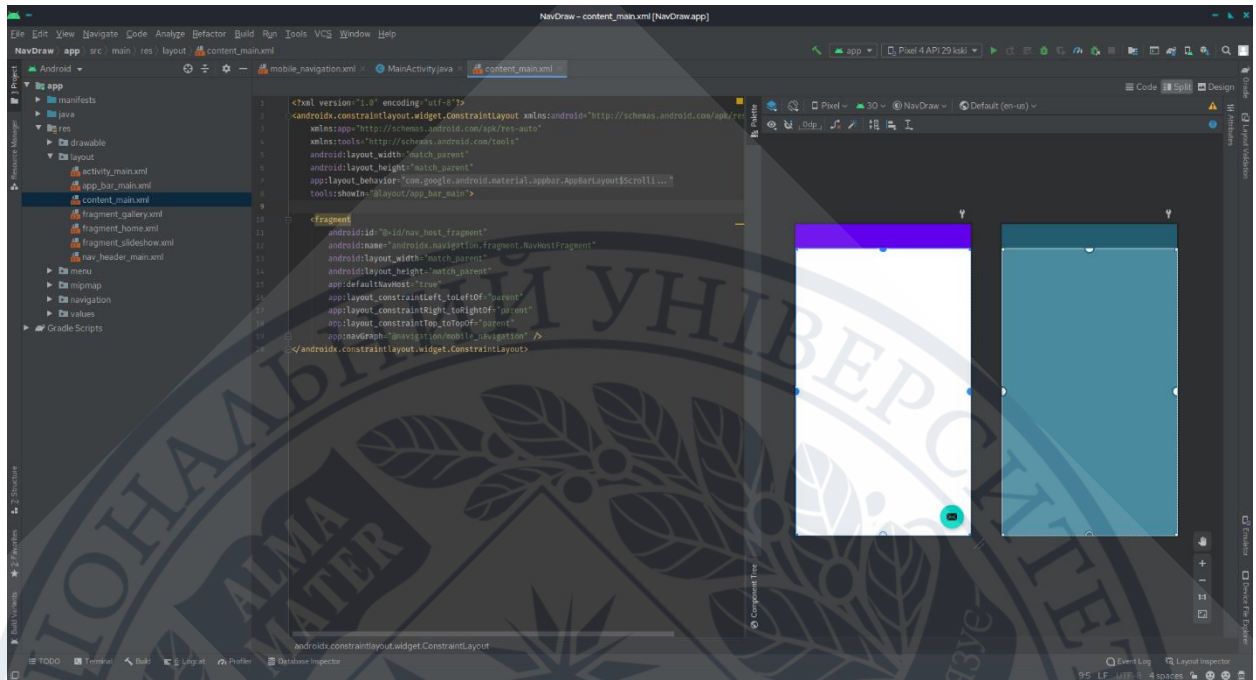


Рисунок 2.2 – Робоча область Android Studio

2.3 SQLite

SQLite - компактна вбудована реляційна база даних. Вихідний код бібліотеки переданий в суспільне надбання. Є чисто реляційної базою даних.

Слово «вбудовується» означає, що SQLite не використовує парадигму клієнт-сервер. Тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якої програма компонується і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID [1] -функції досягаються в тому числі за рахунок створення файлу журналу.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу. Можна, також, ввести таймаут операцій. Тоді підключення, зіткнувшись із зайнятістю БД, чекатиме N-секунду до того, як вилетіти з помилкою `SQLITE_BUSY`.

Також з версії 3.7.0 присутній режим WAL [2], за допомогою якого можна використовувати одну і ту ж базу кількома додатками, як на читання, так і на запис.

У комплекті поставки йде також функціональна клієнтська частина у вигляді виконуваного файлу `sqlite3`, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командним рядком, дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі движка можливо використовувати SQLite як на вбудовуваних системах, так і на виділених машинах з гігабайтними масивами даних.

Формат файлу бази даних є крос-платформних, що дозволяє без проблем використовувати одну і ту ж базу на декількох операційних системах. Також присутня можливість зберігання бази в пам'яті, без її запису на диск. Цей варіант використовується за умовчанням для консольної утиліти `sqlite3`, якщо не вказано ім'я файлу.

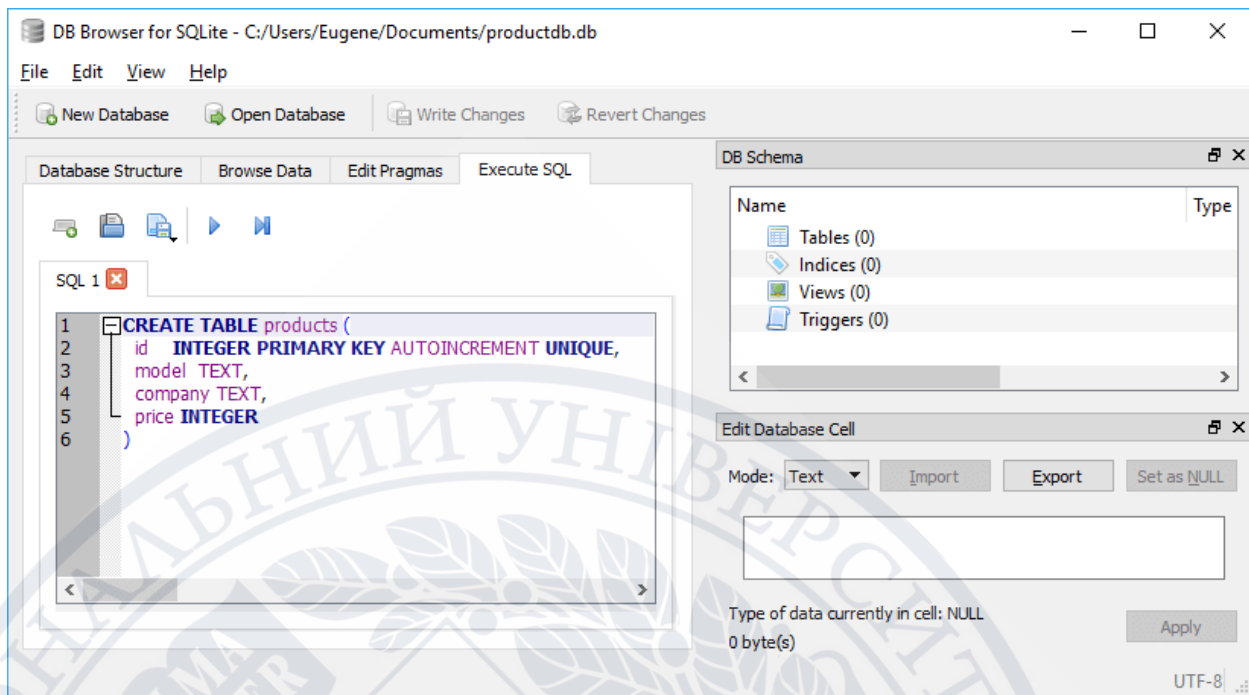


Рисунок 2.3 – DB Browser для работ з базами даних

Висновок до розділу 2

У даному розділі наведені характеристики інструментів, що були використані під час розробки додатку для ведення показників пацієнтів. У наступному розділі буде розглядатися сам процес розробки.

РОЗДІЛ 3

РОЗРОБКА ДОДАТКУ

В даному розділі наведено опис розробки додатку, проведено аналіз вибору засобів та інструментів та зразок розроблення додатку за їх допомогою.

3.1 Огляд Android Studio

Інтерфейс Android Studio схожий на більшість високорівневих IDE. Але розробка для Android досить сильно відрізняється від звичного програмування, коли ми набираємо програму в одному файлі, а потім повністю виконуємо. А тут є безліч файлів ресурсів, які повинні бути згруповані між собою.

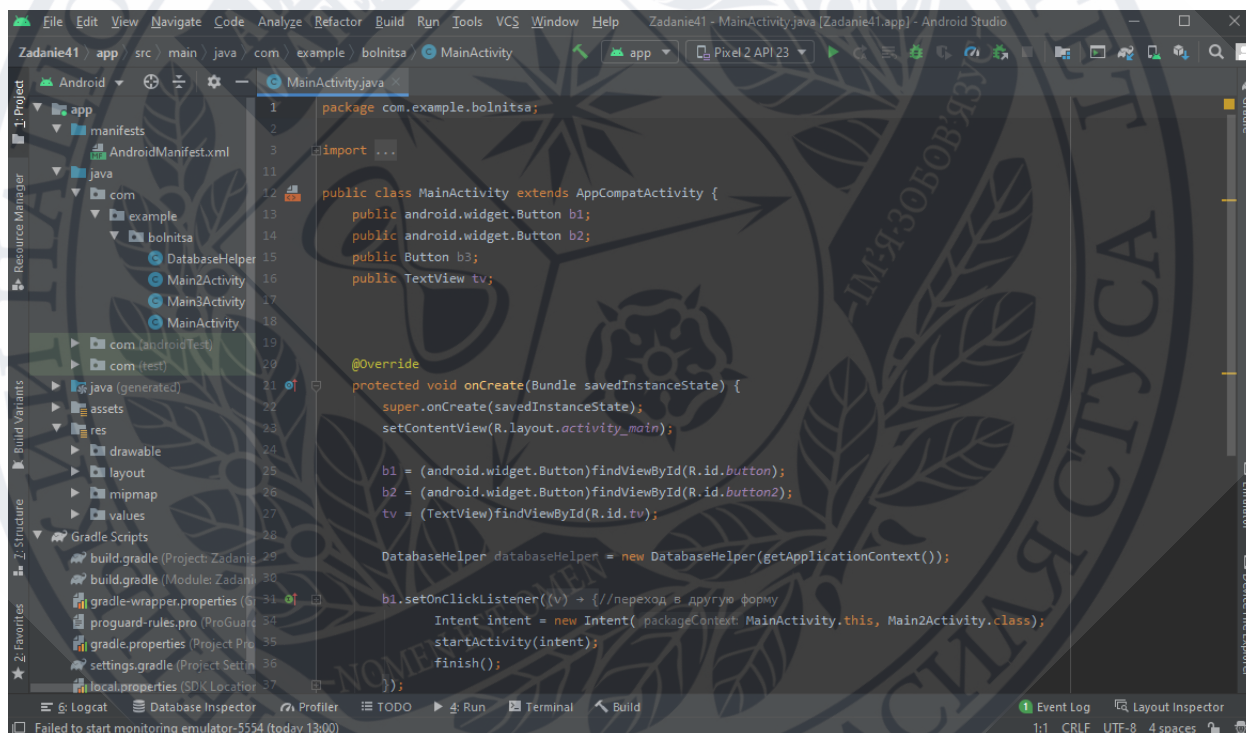


Рисунок 3.1 – Android Studio

Програмування в Android Studio виконується в файлах Java, який має таке ж ім'я, як і у Activity. Однак, зовнішній вигляд програми знаходиться в іншому файлі - це файл xml в якому на мові розмітки описані всі елементи програми. Таким чином, якщо ми хочемо створити кнопку, то нам доведеться описати її в xml файлі, а щоб прив'язати для неї дії - використаємо файл Java.

Ось цей рядок коду завантажує макет з XML файлу:

```
setContentView (. R.layout activity_main);
```

Це означає, що ми могли б використовувати один макет для декількох Activity, а також одна Activity може мати кілька файлів XML з описом відображення. Так чи інакше, можна отримати доступ до всіх файлів проекту в лівій частині вікна, а вкладки над робочою областю дозволяють перемикатися між тими файлами, які зараз відкриті.

Крім того, є ще папка res, в якій знаходяться різні ресурси, такі як зображення. Звернемо увагу, що назви всіх файлів повинні бути в нижньому регістрі.

Ще є папка Values, в якій містяться XML файли зі значеннями різних змінних.

Основна інформація про програму міститься в файлі AndroidManifest.xml, тут описані повноваження, назва програми, мініатюра, і інше.

Можна створити будь-які файли, класи і Activity в будь-який момент, щоб розширити функціональність програми. Просто клацнемо правою кнопкою миші по потрібного каталогу, а потім виберіть "Create".

3.2 Розробка дизайну

Спочатку відкриваємо activity_main.xml файл для того, щоб налаштувати дизайн нашої головної сторінки, яка буде відкриватись після безпосереднього запуску додатку. Я це зробив за допомогою двох об'єктів Button, які відповідають за дві таблиці: Пацієнти та Прийом.

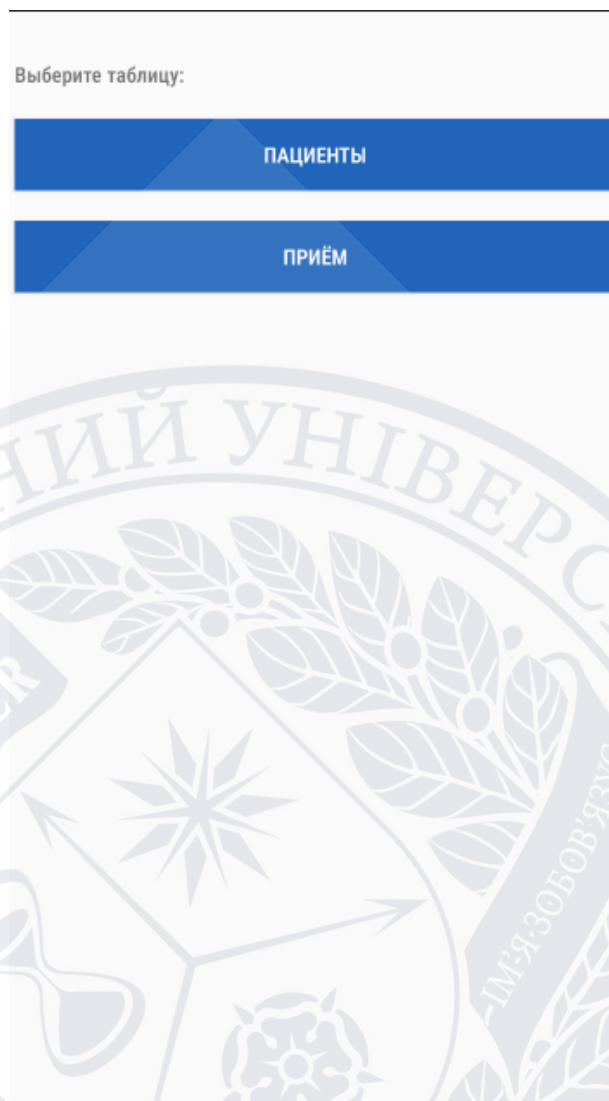


Рисунок 3.2 – Головна форма додатку

В першій таблиці буде зберігатись база даних з усіма пацієнтами, які відвідали клініку, та були записані в відповідну їм медкарту. Для цієї сторінки був створений новий Activity з назвою Main2Activity, а також XML файл з назвою activity_main2.xml. Ця сторінка буде в собі відображати таблицю з персональними даними пацієнтів, які відвідали клініку. В таблиці будуть створені наступні поля: ПІБ, номер телефона, стать, дата народження та паспортні дані. Додамо на сторінку об'єкти EditText з відповідними назвами:

The image shows a screenshot of a mobile application interface for a patient form. At the top, there is a blue navigation bar with five icons: a back arrow, a forward arrow, a document icon, a plus sign, and a trash can. Below the navigation bar, the form consists of several input fields with labels in Russian: 'ФИО' (Full Name), 'Телефон' (Phone), 'Пол' (Gender), 'Дата рождения' (Date of Birth), and 'Паспортные данные' (Passport Data). The form is overlaid on a large, faint watermark of the National University of Medicine and Pharmacy of Ukraine.

Рисунок 3.3 – Форма «Пацієнти»







В другій таблиці будуть зберігатись результати обстежень пацієнтів після прийому. Для цього створюємо новий Activity з назвою Main3Activity та відповідний XML файл activity_main3.xml. В ньому, як і на першій сторінці, буде відображатись таблиця. Таблиця буде містити наступні поля: №, ПІБ пацієнта (вибирається з випадючого списку), дата прийому, ціна, температура, тиск верхня межа, тиск нижня межа, зріст, вага. Для цих полів створимо відповідні EditText:

№
Пациент ФИО
Дата приема
Цена
Температура
Давление верхняя граница
Давление нижняя граница
Рост
Вес

Рисунок 3.4 – Форма «Приём»

Також, для того щоб можна було переключатись між рядками наших таблиць, потрібно створити навігацію. Для цього я вирішив її додати в верхній частині екрану. Так як об'єкт Button не дозволяє додати картинки на нього, а відображається лише в виді кнопки з текстом, для навігації між рядками я додав об'єкти ImageButton. Але спочатку потрібно в директорію проекту додати малюнки (кліпарти) для того щоб пізніше можна було їх додати на відповідні кнопки. Для цього в каталозі drawable я створюю декілька Image Asset's які пізніше буду додавати до кнопок. Потім я під'єдную відповідні кліпарти до кнопок, та розташовую їх в верхній частині екрану.

В підсумку ми отримали 6 кнопок:

-  - для повернення на попередній рядок
-  - для переходу на наступний рядок
-  - для збереження відредагованого рядку
-  - для додавання нового рядку
-  - для видалення рядку
-  - для відправки таблиці по електронній пошті CSV-файлом

3.3 Огляд SQLite

Мобільних додатків, в яких не потрібно зберігати хоча б деяку кількість постійних даних, мало і дуже мало. Використання баз даних є важливим аспектом більшості додатків, починаючи від додатків, які майже повністю керуються даними, і закінчуючи тими, яким просто потрібно зберігати невеликі обсяги даних, таких як переважна оцінка гри.

Важливість зберігання постійних даних стає ще очевиднішою, якщо взяти до уваги тимчасовий життєвий цикл типового додатка Android. З урахуванням постійного ризику того, що система виконання Android припинить роботу компонента програми, щоб звільнити ресурси, комплексна стратегія зберігання даних, щоб уникнути втрати даних, є ключовим фактором у розробці та реалізації будь-якої стратегії розробки додатків.

У цьому підрозділі подано огляд системи управління базами даних SQLite, що постачається в комплекті з операційною системою Android, а також схему класів SDK для Android, яка надається для полегшення постійного зберігання бази даних на основі SQLite у додатку Android. Перш ніж вникати в особливості SQLite в контексті розробки Android, буде викладено короткий огляд баз даних та SQL.

1) Розуміння таблиць баз даних

Таблиці баз даних забезпечують найосновніший рівень структури даних у базі даних. Кожна база даних може містити кілька таблиць, і кожна

таблиця призначена для зберігання інформації певного типу. Наприклад, база даних може містити таблицю клієнтів, яка містить ім'я, адресу та номер телефону для всіх клієнтів певного бізнесу. Ця ж база даних може також включати таблицю товарів, яка використовується для зберігання описів товарів із відповідними кодами товарів для товарів, що продаються підприємством.

Кожній таблиці в базі даних присвоюється ім'я, яке повинно бути унікальним у межах цієї конкретної бази даних. Ім'я таблиці, присвоєне таблиці в одній базі даних, може використовуватися повторно лише в контексті іншої бази даних.

2) Представимо схему баз даних

Схема бази даних визначає характеристики даних, що зберігаються в таблиці бази даних. Наприклад, схема таблиці для таблиці бази даних клієнтів може визначати, що ім'я клієнта - це рядок довжиною не більше 20 символів, а номер телефону клієнта - це числове поле даних певного формату.

Схеми також використовуються для визначення структури цілих баз даних та взаємозв'язку між різними таблицями, що містяться в кожній базі даних.

3) Стовпці та типи даних

Кожен стовпець представляє поле даних у відповідній таблиці. Наприклад, поля імені, адреси та телефонних даних таблиці - це стовпці.

Кожен стовпець, у свою чергу, визначається таким, що містить певний тип даних, який диктує тип даних, який може містити стовпець. Отже, стовпець, призначений для зберігання чисел, буде визначений як числовий тип даних.

4) Рядки бази даних

Кожен новий запис, який зберігається в таблиці, зберігається підряд. Кожен рядок, у свою чергу, складається зі стовпців даних, пов'язаних із збереженим записом.

Розглянемо аналогію з електронними таблицями. Кожен запис у таблиці клієнтів еквівалентний рядку в електронній таблиці, і кожен стовпець містить дані для кожного клієнта (ім'я, адреса, телефон тощо). Коли до таблиці додається новий клієнт, створюється новий рядок, а дані для цього клієнта зберігаються у відповідних стовпцях нового рядка.

Рядки також іноді називають записами, і ці терміни зазвичай можна використовувати як взаємозамінні.

5) Представимо первинні ключі

Кожна таблиця бази даних повинна містити один або кілька стовпців, за допомогою яких можна однозначно ідентифікувати кожен рядок у таблиці. Це відомо в термінології бази даних як первинний ключ. Наприклад, у таблиці може використовуватися стовпець номера банківського рахунку як первинного ключа. Крім того, таблиця клієнтів може використовувати номер соціального страхування клієнта як первинний ключ.

Первинні ключі дозволяють системі управління базами даних однозначно ідентифікувати певний рядок у таблиці. Без первинного ключа неможливо буде отримати або видалити певний рядок у таблиці, оскільки не може бути впевненості, що вибрано правильний рядок. Наприклад, припустимо, що існує таблиця, де прізвище замовника було визначено як первинний ключ. Уявіть тоді проблему, яка може виникнути, якщо в базі даних буде записано більше одного замовника на ім'я "Сміт". Без будь-якого гарантованого способу однозначної ідентифікації конкретного рядка було б неможливо забезпечити доступ до правильних даних у будь-який момент часу.

Первинні ключі можуть містити один стовпець або кілька стовпців у таблиці. Щоб кваліфікуватися як первинний ключ у одному стовпці, жоден рядок не може містити відповідних значень первинного ключа. При використанні кількох стовпців для побудови первинного ключа значення окремих стовпців не повинні бути унікальними, але всі об'єднані стовпці повинні бути унікальними.

6) Що таке SQLite?

SQLite - це вбудована реляційна система управління базами даних (СУБД). Більшість реляційних баз даних (Oracle та MySQL є основними прикладами) - це автономні серверні процеси, які працюють незалежно та у співпраці з програмами, які потребують доступу до баз даних. SQLite називається вбудованим, оскільки він надається у формі бібліотеки, яка пов'язана з програмами. Таким чином, у фоновому режимі не працює автономний сервер баз даних. Усі операції з базою даних обробляються всередині програми за допомогою викликів функцій, що містяться в бібліотеці SQLite.

Розробники SQLite передали цю технологію у загальнодоступне надбання, в результаті чого зараз це широко розгорнуте рішення бази даних.

SQLite написаний мовою програмування C, і, як такий, Android SDK забезпечує "обгортку" на основі Java навколо базового інтерфейсу бази даних. По суті, це складається з набору класів, які можуть бути використані в коді Java програми для створення та управління базами даних на основі SQLite.

7) Structured Query Language (SQL)

Доступ до даних здійснюється в базах даних SQLite, використовуючи мову високого рівня, відому як Structured Query Language. Зазвичай скорочується до SQL. SQL - це стандартна мова, що використовується більшістю реляційних систем управління базами даних. SQLite відповідає здебільшого стандарту SQL-92.

SQL - це, по суті, дуже проста і легка у використанні мова, розроблена спеціально для забезпечення читання та запису даних бази даних. Оскільки SQL містить невеликий набір ключових слів, її можна швидко вивчити. Крім того, синтаксис SQL більш-менш однаковий між більшістю реалізацій СУБД, тому, вивчивши SQL для однієї системи, цілком ймовірно, що ваші навички перенесуться в інші системи управління базами даних.

8) SQLite на віртуальному пристрої Android (AVD)

Для того, хто не знайомий з базами даних загалом та SQLite зокрема, заглиблення у створення програми Android, яка використовує SQLite, може здатися трохи лякаючим. На щастя, Android постачається з попередньо встановленим SQLite, включаючи інтерактивне середовище для видачі команд SQL із сеансу оболонки adb, підключеного до запущеного екземпляра емулятора AVD Android. Це одночасно корисний спосіб дізнатись про SQLite та SQL, а також безцінний інструмент для виявлення проблем з базами даних, створеними програмами, що працюють в емуляторі.

Щоб запустити інтерактивний сеанс SQLite, почнемо із запуску сеансу AVD. Цього можна досягти в межах Android Studio, запустивши диспетчер віртуальних пристроїв Android (Інструменти -> Android -> AVD Manager), вибравши попередньо налаштований AVD і натиснувши кнопку Пуск.

Після того, як AVD запущений, відкриємо вікно терміналу або командного рядка та підключимося до емулятора за допомогою інструмента командного рядка adb наступним чином:

```
adb -e shell
```

Після підключення середовище оболонки надасть командний рядок, в якому можна вводити команди:

```
root@android:/ #
```

Дані, що зберігаються в базах даних SQLite, насправді зберігаються у файлах баз даних у файловій системі пристрою Android, на якому запущена програма. За замовчуванням шлях до файлової системи для цих файлів бази даних такий:

```
/ data / data / <назва пакета> / databases / <назва файлу бази даних> .db
```

Наприклад, якщо програма з іменем пакета com.example.MyDBApp створює базу даних з іменем mydatabase.db, шлях до файлу на пристрої матиме такий вигляд:

```
/data/data/com.example.MyDBApp/databases/mydatabase.db
```

Отже, для приклада змінимо каталог на /data/data усередині оболонки adb і створимо ієрархію підкаталогів, придатну для деяких експериментів з SQLite:

```
cd /data/data
mkdir com.example.dbexample
cd com.example.dbexample
mkdir databases
cd databases
```

За допомогою розташування, створеного для файлу бази даних, запусимо інтерактивний інструмент SQLite наступним чином:

```
root@android:/data/data/databases # sqlite3
./mydatabase.db
sqlite3 ./mydatabase.db
SQLite version 3.7.4
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

У рядку sqlite> можна вводити команди для виконання таких завдань, як створення таблиць, вставка та отримання даних. Наприклад, для створення нової таблиці в нашій базі даних із полями для полів ідентифікатора, імені, адреси та номера телефону потрібна наступна команда:

```
create table contacts (_id integer primary key
autoincrement, name text, address text, phone text);
```

Звернемо увагу, що кожен рядок таблиці повинен мати первинний ключ, унікальний для цього рядка. У наведеному вище прикладі ми визначили поле ID як первинний ключ, оголосили його цілим числом і попросили SQLite автоматично збільшувати число кожного разу, коли додається рядок. Це поширений спосіб переконатися, що кожен рядок має унікальний первинний ключ. На більшості інших платформ вибір імені для первинного ключа є довільним. У випадку з Android, важливо, щоб ключ був названий `_id`, щоб база даних була повністю доступною з використанням усіх класів, пов'язаних з базою даних Android. Решта полів оголошено типовим текстом.

Щоб перерахувати таблиці у вибраній на даний момент базі даних, використовується оператор `.tables`:

```
sqlite> .tables  
contacts
```

Щоб вставити записи в таблицю:

```
sqlite> insert into contacts (name, address, phone)  
values ("David Smith", "123 Main Street, California", "123-  
555-2323");  
sqlite> insert into contacts (name, address, phone)  
values ("Mark Parks", "10 Upping Street, Idaho", "444-444-  
1212");
```

Щоб отримати всі рядки з таблиці:

```
sqlite> select * from contacts;  
1|David Smith|123 Main Street, California|123-555-2323  
2|Mark Parks|10 Upping Street, Idaho|444-444-1212
```

Щоб витягти рядок, який відповідає певним критеріям:

```
sqlite> select * from contacts where name="Mark Parks";
```

Щоб вийти з інтерактивного середовища sqlite3:

```
sqlite> .exit
```

Під час запуску програми Android у середовищі емулятора будь-які файли бази даних будуть створені у файловій системі емулятора, використовуючи попередньо обговорену конвенцію про шлях. Ця перевага полягає в тому, що ви можете підключитися до adb, перейти до розташування файлу бази даних, завантажити його в інтерактивний інструмент sqlite3 та виконати завдання над даними для виявлення можливих проблем, що виникають у коді програми.

Важливо також зазначити, що, хоча можливо з'єднати оболонку adb з фізичним пристроєм Android, оболонка за замовчуванням не отримує достатніх привілеїв для створення та управління базами даних SQLite. Тому налагодження проблем з базами даних найкраще виконувати за допомогою сеансу AVD.

9) Класи Java на Android SQLite

Як уже згадувалося, SQLite написаний мовою програмування C, тоді як програми для Android розробляються переважно за допомогою Java. Щоб подолати цей «мовний розрив», Android SDK включає набір класів, які забезпечують рівень Java поверх системи управління базами даних SQLite. Решта цього розділу надасть базовий огляд кожного з основних класів цієї категорії.

Cursor

Клас, спеціально наданий для забезпечення доступу до результатів запиту до бази даних. Наприклад, операція SQL SELECT, виконана в базі даних, потенційно поверне з бази даних кілька відповідних рядків. Екземпляр

курсора може використовуватися для проходження цих результатів, до яких потім можна отримати доступ із коду програми, використовуючи різні методи. Деякі ключові методи цього класу:

- `close()` - звільняє всі ресурси, що використовуються курсором, і закриває його.
- `getCount()` - повертає кількість рядків, що містяться в наборі результатів.
- `moveToFirst()` - Переходить до першого рядка в наборі результатів.
- `moveToLast()` - Переходить до останнього рядка в наборі результатів.
- `moveToNext()` - Переходить до наступного рядка в наборі результатів.
- `move()` - переміщується на певний зсув від поточної позиції в наборі результатів.
- `get<type>()` - Повертає значення вказаного `<type>`, що міститься в зазначеному індексі стовпця рядка в поточній позиції курсору (варіації складаються з `getString()`, `getInt()`, `getShort()`, `getFloat()` і `getDouble()`).

База даних SQLiteDatabase

Цей клас забезпечує основний інтерфейс між кодом програми та базовими базами даних SQLite, включаючи можливість створювати, видалити та виконувати операції на базі даних на основі SQL. Деякі ключові методи цього класу:

- `insert()` - вставляє новий рядок у таблицю бази даних.
- `delete()` - Видаляє рядки з таблиці бази даних.
- `query()` - виконує вказаний запит до бази даних і повертає відповідні результати через об'єкт курсору.
- `execSQL()` - виконує єдиний оператор SQL, який не повертає дані результату.
- `rawQuery()` - виконує оператор запиту SQL і повертає відповідні результати у вигляді об'єкта курсору.

SQLiteOpenHelper

Допоміжний клас, призначений для спрощення створення та оновлення баз даних. Цей клас повинен бути підкласований у кодї програми, яка шукає доступ до бази даних, і таких методів зворотного виклику, реалізованих у цьому підкласі:

- `onCreate()` - Викликається, коли база даних створюється вперше. Цей метод передається як аргумент об'єкта `SQLiteDatabase` для новоствореної бази даних. Це ідеальне місце для ініціалізації бази даних з точки зору створення таблиці та вставки будь-яких початкових рядків даних.
- `onUpgrade()` - Викликається в тому випадку, якщо код програми містить найновіший посилання на номер версії бази даних. Зазвичай це використовується, коли програма оновлюється на пристрої, і вимагає оновлення схеми бази даних для обробки додаткових даних.

На додаток до вищезазначених обов'язкових методів зворотного виклику, метод `onOpen()`, викликаний при відкритті бази даних, також може бути реалізований в рамках підкласу.

Конструктор для підкласу також повинен бути реалізований для виклику супер класу, проходячи через контекст програми, ім'я бази даних та версію бази даних.

До відомих методів класу `SQLiteOpenHelper` належать:

- `getWritableDatabase()` - Відкриває або створює базу даних для читання та запису. Повертає посилання на базу даних у вигляді об'єкта `SQLiteDatabase`.
- `getReadableDatabase()` - Створює або відкриває базу даних лише для читання. Повертає посилання на базу даних у вигляді об'єкта `SQLiteDatabase`.
- `close()` - Закриває базу даних.

ContentValues

ContentValues - це зручний клас, який дозволяє оголошувати пари ключ/значення, що складаються з ідентифікаторів стовпців таблиці та значень, що зберігаються в кожному стовпці. Цей клас особливо використовується під час вставки або оновлення записів у таблиці бази даних.

10) Підсумок

SQLite - це легка вбудована реляційна система управління базами даних, яка входить до складу фреймворку Android і забезпечує механізм реалізації організованого постійного зберігання даних для додатків Android. На додаток до бази даних SQLite, фреймворк Android також включає цілий ряд класів Java, які можуть бути використані для створення та управління базами даних та таблицями на основі SQLite.

3.4 Створюємо базу даних

Для роботи з базою даних створимо окремий клас з назвою DatabaseHelper в якому будуть міститись такі змінні та методи:

- String DB_NAME – для збереження назви бази даних
- String DB_PATH – для присвоєння шляху бази даних на диску
- Int DB_VERSION
- SQLiteDatabase mDataBase – об'єкт бази даних
- Context mContext
- boolean mNeedUpdate – для перевірки, чи потрібно оновити нашу базу даних
- public DatabaseHelper(Context context) – конструктор в якому створюється база даних та записується на диск
- public void updateDataBase() – метод для оновлення бази даних, який видаляє існуючий файл бази даних та викликає метод для копіювання бази даних

- `private boolean checkDataBase()` – метод для перевірки, чи існує файл бази даних на диску
- `private void copyDataBase()` – метод для виклику методу копіювання бази даних, який перед цим перевіряє чи існує файл бази даних на диску
- `private void copyDBFile()` – метод, який створює на диску оновлений файл бази даних
- `public synchronized void close()` – метод для закриття об'єкту бази даних
- `public void onCreate(SQLiteDatabase db)` – метод створений за замовчуванням
- `public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)` – метод створений за замовчуванням

3.5 Створюємо об'єкти

Для всіх об'єктів на наших формах потрібно створити відповідні об'єкти в класах цих форм. Для цього пишемо такий код, який взятий з MainActivity:

```
b1 = (android.widget.Button) findViewById(R.id.button);
b2 = (android.widget.Button) findViewById(R.id.button2);
tv = (TextView) findViewById(R.id.tv);
```

Для всіх інших Activity створюємо відповідні об'єкти кнопок та EditText'ів. Пишемо код для того, щоб при натисканні на кнопки відбувався перехід на відповідну форму. Нижче наведено такий код для кнопки «Пацієнти»:

```
b1.setOnClickListener(new View.OnClickListener() {
    @Override
    Intent intent = new Intent(MainActivity.this,
Main2Activity.class);
    startActivity(intent);
});
```

```
        finish();  
    }  
});
```

На даний момент наш додаток працює: можна переходити із однієї форми в іншу, вписувати дані в поля EditText та натискати кнопки навігації, правда доки вони не мають ніякого функціоналу.

3.6 Робота з базою даних

Настав час підключити базу даних до наших Activity. Для цього в обох класах (пацієнти та прийом) створюємо об'єкти DatabaseHelper. При першому запуску додатку в його директорії буде створена база даних, за допомогою раніше створеного конструктора в класі DatabaseHelper.

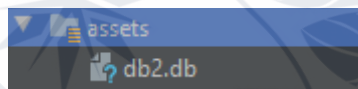



Рисунок 3.5 – Файл бази даних

Також напишемо код для EditText, які будуть відображати перший рядок при відкритті форм «пацієнти» та «прийом». Для цього імпортуємо нашу базу даних в cursor та відобразимо отриманий рядок таблиці на об'єктах EditText. Щоб це зробити, пишемо наступний код (Activity з пацієнтами):


```
cursor = mDb.rawQuery("SELECT * FROM [pacient]", null);  
cursor.moveToFirst();  
if (!cursor.isAfterLast()) {  
    key=cursor.getString(0);  
    fio.setText(cursor.getString(0));  
    tel.setText(cursor.getString(1));  
    pol.setText(cursor.getString(2));  
    datroj.setText(cursor.getString(3));  
    passport.setText(cursor.getString(4));  
}
```

На даний момент наш додаток може отримувати таблиці для обох форм, та відображати їх на обох формах. Проте, ми нічого з ними не можемо зробити: ні переключити, ні видалити, ні додати новий рядок в таблицю. Тому потрібно задати функціонал для кнопок навігації.

Спочатку напишемо код для перших 5-ти кнопок, і залишимо на потім кнопку «Відправити CSV-файлом на електронну пошту».

-  Для того щоб повертатись назад, виконаємо просту команду для курсору `cursor.moveToPrevious()`, та відобразимо отриманий рядок на об'єктах `EditTest`:

```
nazad.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        cursor.moveToPrevious();  
        if (!cursor.isBeforeFirst()) {  
            key=cursor.getString(0);  
            fio.setText(cursor.getString(0));  
            tel.setText(cursor.getString(1));  
            pol.setText(cursor.getString(2));  
            datroj.setText(cursor.getString(3));  
            passport.setText(cursor.getString(4));  
        }  
    }  
});
```


-  Для того щоб перейти вперед, виконаємо просту команду для курсору `cursor.moveToNext()`, та відобразимо отриманий рядок на об'єктах `EditTest`:

```
vpered.setOnClickListener(new View.OnClickListener()  
{  
    @Override  
    public void onClick(View v) {  
        cursor.moveToNext();  
        if (!cursor.isAfterLast()) {  
            key=cursor.getString(0);  
        }  
    }  
});
```

```

        fio.setText(cursor.getString(0));
        tel.setText(cursor.getString(1));
        pol.setText(cursor.getString(2));
        datroj.setText(cursor.getString(3));
        passport.setText(cursor.getString(4));
    }
}
});

```

-  - для того, щоб зберегти відредагований рядок в базу даних, поміщаємо його в ContentValues та оновлюємо базу даних методом update():

```

sohr.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        ContentValues cv = new ContentValues();

        cv.put("fio", fio.getText().toString());
        cv.put("telephone",
tel.getText().toString());
        cv.put("pol", pol.getText().toString());
        cv.put("datroj",
datroj.getText().toString());
        cv.put("passport_dannie",
passport.getText().toString());

        int updCount = mDb.update("pacient", cv,
"fio = ?", new String[] { key });
        Log.d(LOG_TAG, "updated rows count = " +
updCount);


        cursor.close();
        cursor = mDb.rawQuery("SELECT * FROM
pacient", null);
        cursor.moveToFirst();
        if (!cursor.isAfterLast()) {}
        while (!cursor.isAfterLast()) {
            if (key==cursor.getString(0)) break;

```

```

        cursor.moveToNext();
    }
});

```

-  - для додавання нового рядку створюємо новий пустий рядок, та видаляємо текст з об'єктів EditText. Після цього в поля вводяться дані та зберігаються за допомогою кнопки «Зберегти»:

```

dobit.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) {


        ContentValues cv = new ContentValues();

        cv.put("fio", "");
        cv.put("telephone", "");
        cv.put("pol", "");
        cv.put("datroj", "");
        cv.put("passport_dannie", "");

        long rowID = mDb.insert("pacient", null,
cv);
        Log.d(LOG_TAG, "row inserted, ID = " +
rowID);

        cursor.close();
        cursor = mDb.rawQuery("SELECT * FROM
pacient", null);
        cursor.moveToLast();
        if (!cursor.isAfterLast()) {
            key=cursor.getString(0);
            fio.setText(cursor.getString(0));
            tel.setText(cursor.getString(1));
            pol.setText(cursor.getString(2));
            datroj.setText(cursor.getString(3));
            passport.setText(cursor.getString());
        }
    }
});

```

-  - для видалення рядку використаємо метод SQLiteDatabase під назвою delete, який видалить наш рядок використовуючи ПІБ як первинний ключ. Потім заповнимо наші об'єкти EditText попереднім рядком:

```
        udalit.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v) {

                Log.d(LOG_TAG, "key = " + key);
                int delCount = mDb.delete("pacient", "fio = '" + key + "'", null);
                Log.d(LOG_TAG, "deleted rows count = " + delCount);

                cursor.moveToPrevious();
                key="";
                fio.setText("");
                tel.setText("");
                pol.setText("");
                datroj.setText("");
                passport.setText("");
                Log.d(LOG_TAG, "key11 = " + key);

                if (!cursor.isBeforeFirst()) {
                    key=cursor.getString(0);
                    fio.setText(cursor.getString(0));
                    tel.setText(cursor.getString(1));
                    pol.setText(cursor.getString(2));
                    datroj.setText(cursor.getString(3));
                    passport.setText(cursor.getString(4));
                }

                cursor.close();
                cursor = mDb.rawQuery("SELECT * FROM pacient", null);
                cursor.moveToFirst();
                if (!cursor.isAfterLast()) {}
                while (!cursor.isAfterLast()) {
```

```

        if (key.length()==0)
        {key=cursor.getString(0);}
        if (key==cursor.getString(0)) {break;}
        cursor.moveToNext();
    }
    Log.d(LOG_TAG, "key22 = " + key);
}
});

```

На даний момент в наш додаток можна додавати та зберігати нові рядки в таблицю, можна перемикались між рядками таблиці, які будуть відображатись на форму, за допомогою кнопок навігації, та можна видаляти рядки з таблиці. На даному етапі наш додаток майже готовий. В умові роботи було сказано, що додаток повинен проводити невеликий аналіз показників даних, а саме: підсвічувати небезпечні для пацієнта показники червоним, а безпечні – зеленим. Тому ми робимо це за допомогою такого коду:

```

        key=cursor.getString(0);
        nom.setText(cursor.getString(0));
        ArrayAdapter adapter2 = (ArrayAdapter)
        pfio.getAdapter(); int position2 =
        adapter2.getPosition(cursor.getString(1));pfio.setSelection
        on(position2);
        dat.setText(cursor.getString(2));
        cena.setText(cursor.getString(3));
        temperat.setText(cursor.getString(4));
        if
        (Float.parseFloat(cursor.getString(4))>36.6f){temperat.set
        TextColor(Color.parseColor("#FF0000"));}
        else
        {temperat.setTextColor(Color.parseColor("#42d209"));}
        davl_verh.setText(cursor.getString(5));
        if
        (Integer.parseInt(cursor.getString(5))>120){davl_verh.set
        TextColor(Color.parseColor("#FF0000"));}
        else
        {davl_verh.setTextColor(Color.parseColor("#42d209"));}
        davl_nijn.setText(cursor.getString(6));
        if
        (Integer.parseInt(cursor.getString(5))<80){davl_nijn.setT

```

```
extColor(Color.parseColor("#FF0000"));}
else
{davl_nijn.setText(Color.parseColor("#42d209"));}
rost.setText(cursor.getString(7));
ves.setText(cursor.getString(8));
```

На даний момент заповнені дані в рядках на формах виглядають таким чином:

The screenshot shows a mobile application interface for a hospital. The title bar is purple with the text 'Больница'. Below the title bar is a navigation bar with five icons: back, forward, save, add, and delete. The form contains the following fields and values:

Field Label	Value
ФИО	Петрова А.Н.
Телефон	896536925
Пол	Ж
Дата рождения	01.02.1981
Паспортные данные	01 01 010101

Рисунок 3.6 – Форма «Пацієнти»

Больница

← → [Icon] + [Icon] →

№
1

Пациент ФИО
Петрова А.Н.

Дата приема
01.02.2020

Цена
10000

Температура
36.6

Давление верхняя граница
160

Давление нижняя граница
120

Рост
170

Вес

Рисунок 3.7 – Форма «Приём»

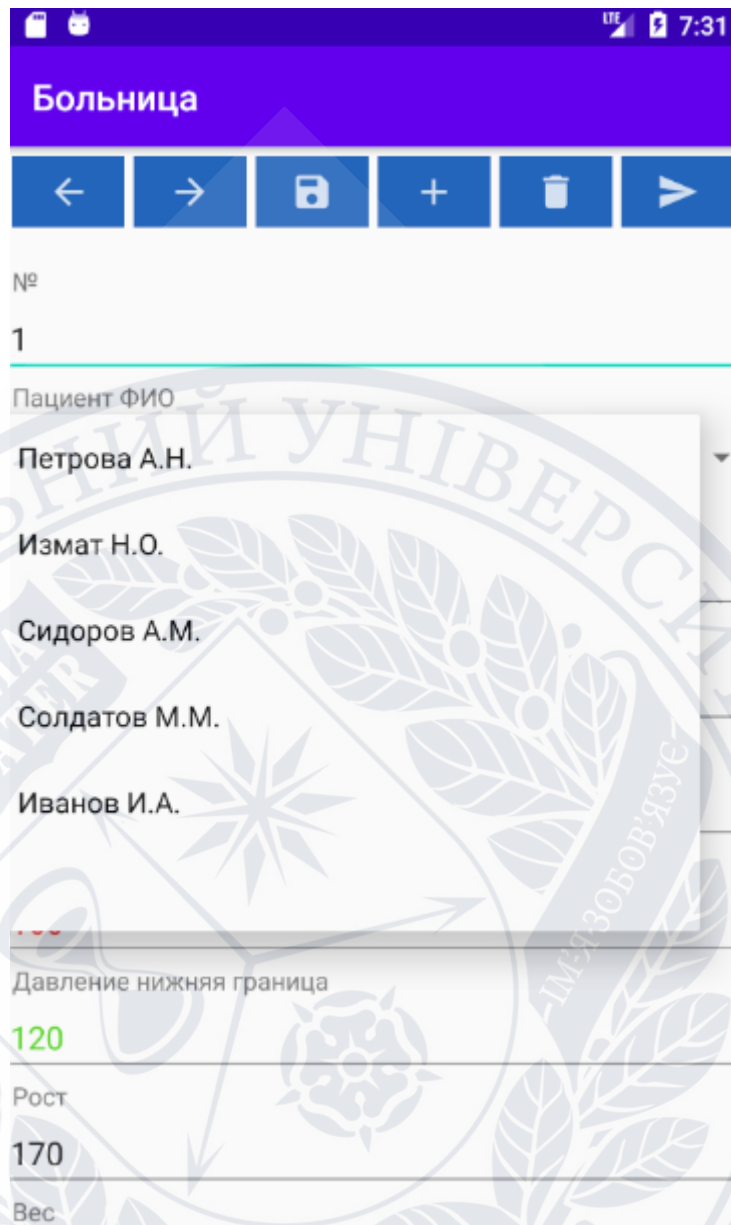


Рисунок 3.8 – Випадаючий список ПІБ

3.7 Відправка бази даних електронною поштою

Останні що потрібно зробити для повноцінного функціонування нашого додатку – це додати можливість відправляти CSV-файл за вказаною електронною поштою. Для цього створимо String filename = “myFile.csv” для того щоб присвоїти йому значення з назвою csv файлу, яке ми будемо використовувати пізніше. Потім створюємо сам об’єкт файлу File mFile. Щоб файл створився, перевіримо спочатку чи має додаток доступ до сховища

телефону, а потім прописуємо шлях для збереження файлу за допомогою такого коду:

```
if (!isAvailable() || isReadOnly()) {  
  
    otprav.setEnabled(false);  
  
} else {  
  
    mFile = new File(getExternalFilesDir(filePath),  
fileName);  
}
```

На даний момент було створено файл в сховищі телефона з розширенням CSV. Потрібно відмітити, що якщо додаток не буде мати доступу до сховищу, або якщо він не зможе сховище редагувати, то кнопка відправки перестане бути доступною, а файл не створиться.

Після цих дій створимо `FileWriter` для нашого файлу, а потім `BufferedWriter` для `FileWriter`, щоб записати в нього необхідні дані. Так як CSV-файл – це таблиця, дані ми передаємо через «;». Таким чином, програма розділяє наші значення на стовпці і заносе кожне значення в новий стовпець. В кінці вводу рядка додамо «\n» для того щоб дані перестали записуватись в один рядок та перейшли на новий. Таким чином, вписуємо спочатку назви стовпців на англійській мові (так як в CSV-файл не передається кирилиця), а потім з допомогою циклу та об'єкту `cursor` отримуємо всі інші рядки. Як тільки в додатку користувач натисне кнопку «відіслати» файл збережеться на диску та буде виглядати таким чином:

	A	B	C	D	E	F	G	H	I
1	Nomer	Date	Price	Temp	PressUp	PressDow	Height	Weight	
2	1	01.02.2020	10000	36.6	160	120	170	120	
3	2	01.02.2020	10000	37.9	140	100	150	55	
4	3	02.02.2020	50000	38.1	120	80	155	45	
5	4	02.02.2020	50000	36.6	110	80	165	60	
6	5	02.02.2020	15000	36.6	120	80	163	57	
7									
8									
9									

Рисунок 3.9 – Відправлений файл CSV

Як бачимо на рисунку, кожен рядок має всі необхідні показники окрім ПІБ-пацієнта. Замість цього в таблицю передається номер пацієнта, тому потрібно додатково мати список пацієнтів та їхній порядковий номер.

Таким чином ми змогли реалізувати збереження файлу в CSV розширенні, але файл ще має відправлятися по електронній пошті.

Для цього ми використаємо об'єкт класу Intent за допомогою якого зручно можна втілити в життя можливість відіслати електронний лист через Android Додаток. За допомогою цього об'єкту ми передаємо тип листа, можна вказати його тему та текст, а також додати адреси електронної пошти тих, кому ми хочемо відіслати файл з базою даних. Після цього ми поміщаємо наш файл в об'єкт класу Uri, і саме за допомогою цього класу ми зможемо прикріплювати файл до нашого листа. Код для кнопки «відправити» виглядає так:

```

i otprav.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mData = "" ;

        try {

            FileWriter fw = new
FileWriter(mFile.getAbsolutePath());
            BufferedWriter bw = new

```

```

BufferedWriter(fw);

bw.write("Nomer;Date;Price;Temp;PressUp;PressDown;Height;
Weight\n");

        cursor.moveToFirst();

        while(!cursor.isAfterLast()){
            for (int i=0; i<9; i++){
                if(i!=1) {
                    bw.write(cursor.getString(i));
                    bw.write(";");
                }
            }
            bw.write("\n");
            cursor.moveToNext();
        }

        bw.close();

        StrictMode.VmPolicy.Builder builder =
new StrictMode.VmPolicy.Builder();
        StrictMode.setVmPolicy(builder.build());

        Intent emailIntent = new
Intent(Intent.ACTION_SEND);
        emailIntent.setType("text/plain");
        emailIntent.putExtra(Intent.EXTRA_EMAIL,
new String[]{"yes342@ukr.net"}); //указанный email
        emailIntent.putExtra(Intent.EXTRA_SUBJECT,
"Информация о больных");
        emailIntent.putExtra(Intent.EXTRA_TEXT, "");

        File file = new
File(getExternalFilesDir(filePath), fileName);
        Uri uri = Uri.fromFile(file);
        emailIntent.putExtra(Intent.EXTRA_STREAM,
uri);

startActivity(Intent.createChooser(emailIntent, "Pick an
Email provider"));
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```
    }  
    showToast("Files was saved");  
}  
  
});
```

Тепер можна відправляти файл з базою даних наших пацієнтів електронною поштою CSV-файлом. Щоб це відбулося, спочатку в додатку з'являється вікно з текстом «Pick and Email provider» в якому потрібно вибрати, за допомогою якого поштового провайдеру ми хочемо відправити лист. Потім, з'являється лист з вже прикріпленим файлом, та вписаними адресами отримувачів, які були вказані в коді.

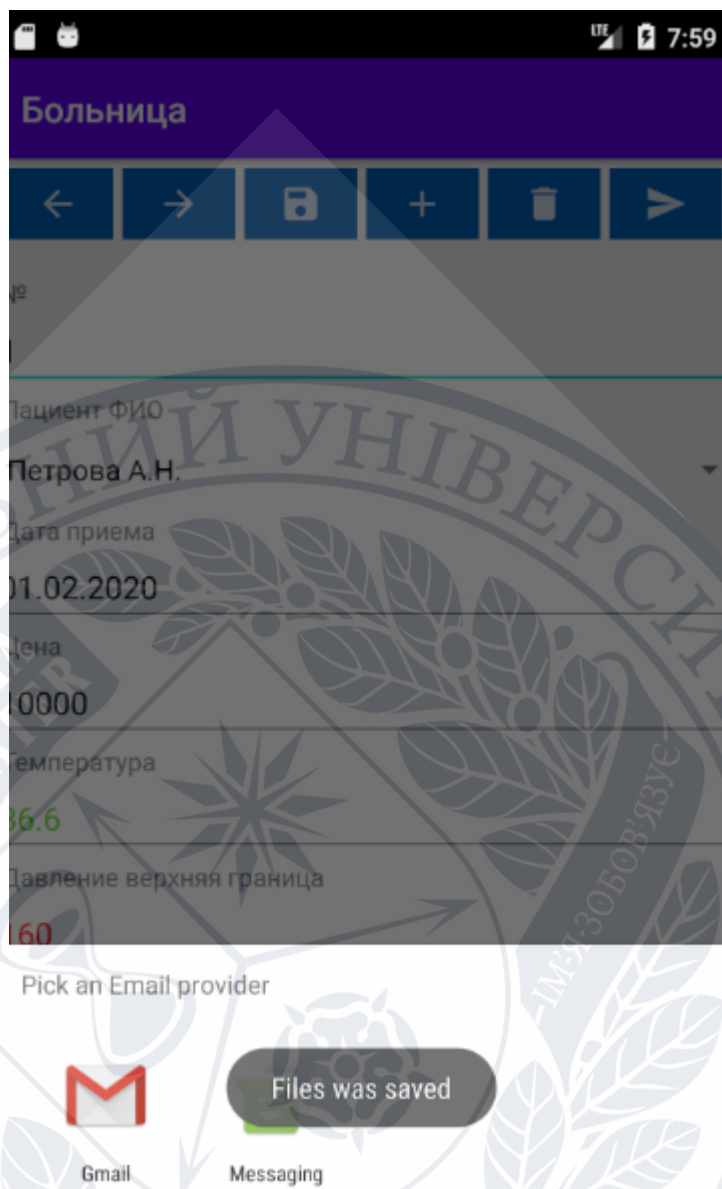


Рисунок 3.10 – Процес відправки CSV-файлу

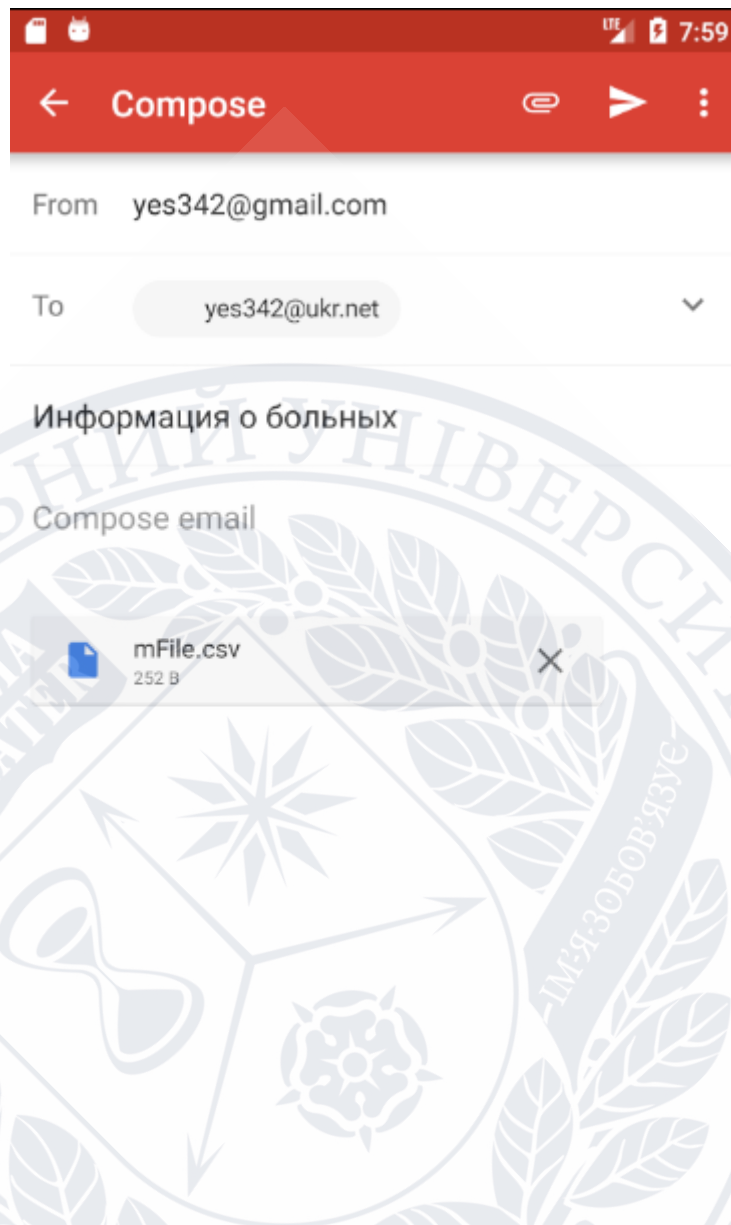


Рисунок 3.11 – Процес відправки CSV-файлу

При натисканні на кнопку «відправити» в додатку електронної пошти все відправляється на вказаний адрес, що можна перевірити на практиці.

○ Юра Иванов <yes342@gmail.com>

Кому: yes342@ukr.net

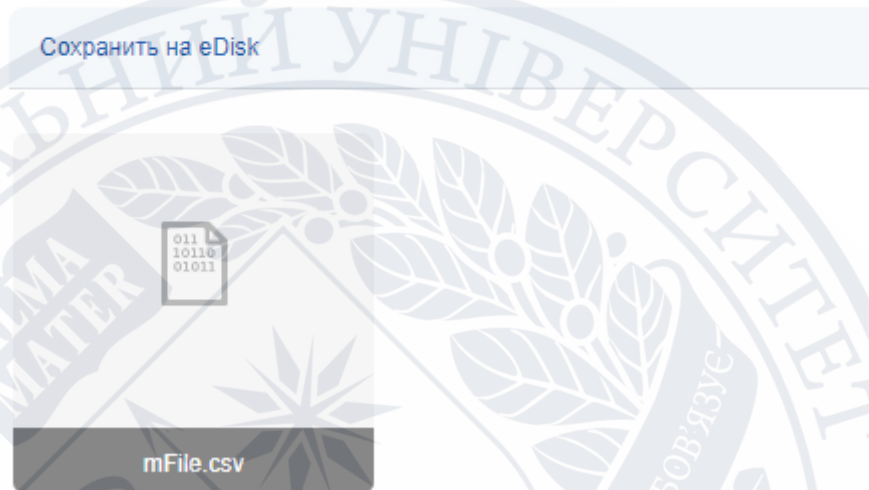


Рисунок 3.12 – отриманий па пошті CSV-файл

І дійсно як бачимо, файл прийшов на пошту, так виглядає так само як на скріншоті вище.

На даному етапі розробка додатку була виконана. Додаток справляється з усіма поставленими задачами: додавання нових рядків для таблиць з пацієнтами та прийомом, аналіз показників в виді виділення рядків з небезпечними та безпечними для здоров'я цифрами, редагування, збереження, та видалення існуючих рядків, перемикання між рядками та відправка бази даних по електронній пошті.

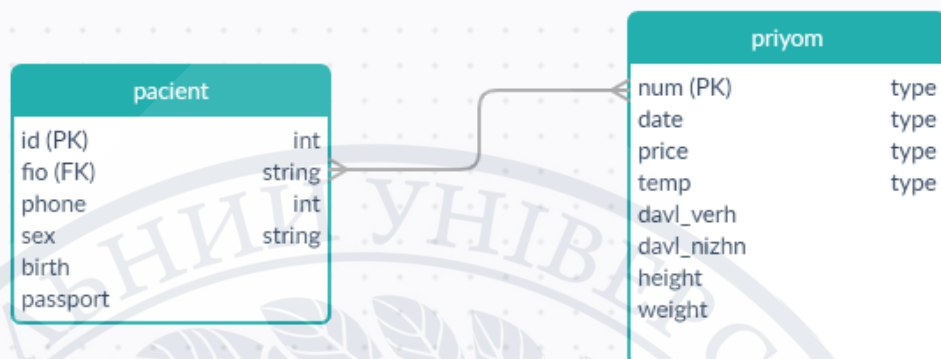


Рисунок 3.13 – Схема бази даних додатку

ВИСНОВКИ

В сучасних передових країнах світу дуже актуальною є тема комп'ютеризації всіх можливих видів діяльності. Це спрощує роботу з інформацією, так як для її ведення не потрібно зберігати велику кількість паперів, з якими складно працювати, а достатньо лише ввімкнути комп'ютер та запустити необхідний додаток.

Але навіть комп'ютер не є максимально мобільним, і отримати до нього доступ можна далеко не завжди. Для вирішення цієї проблеми можна запропонувати телефон, який завжди є під рукою та зручний в використанні. Саме цим я займався в даній лабораторній роботі – розробляв додаток для роботи з базами даних для лікарень.

Потреба подібних розробок в нашій країні все ще є актуальною. Хоча й досягнення в цій області на даний момент достатньо переконливі, комп'ютеризація медичних закладів виконана не на сто відсотків. Наприклад, в маленьких містах та сілах документація все ще ведеться в виді паперів. Тому дана бакалаврська робота несе цінність як для медичних закладів, так і для будь-яких інших державних закладів, тому що достатньо універсальна і може містити в собі таблиці з різними даними, достатньо лише перейменувати існуючі поля та додати нові.

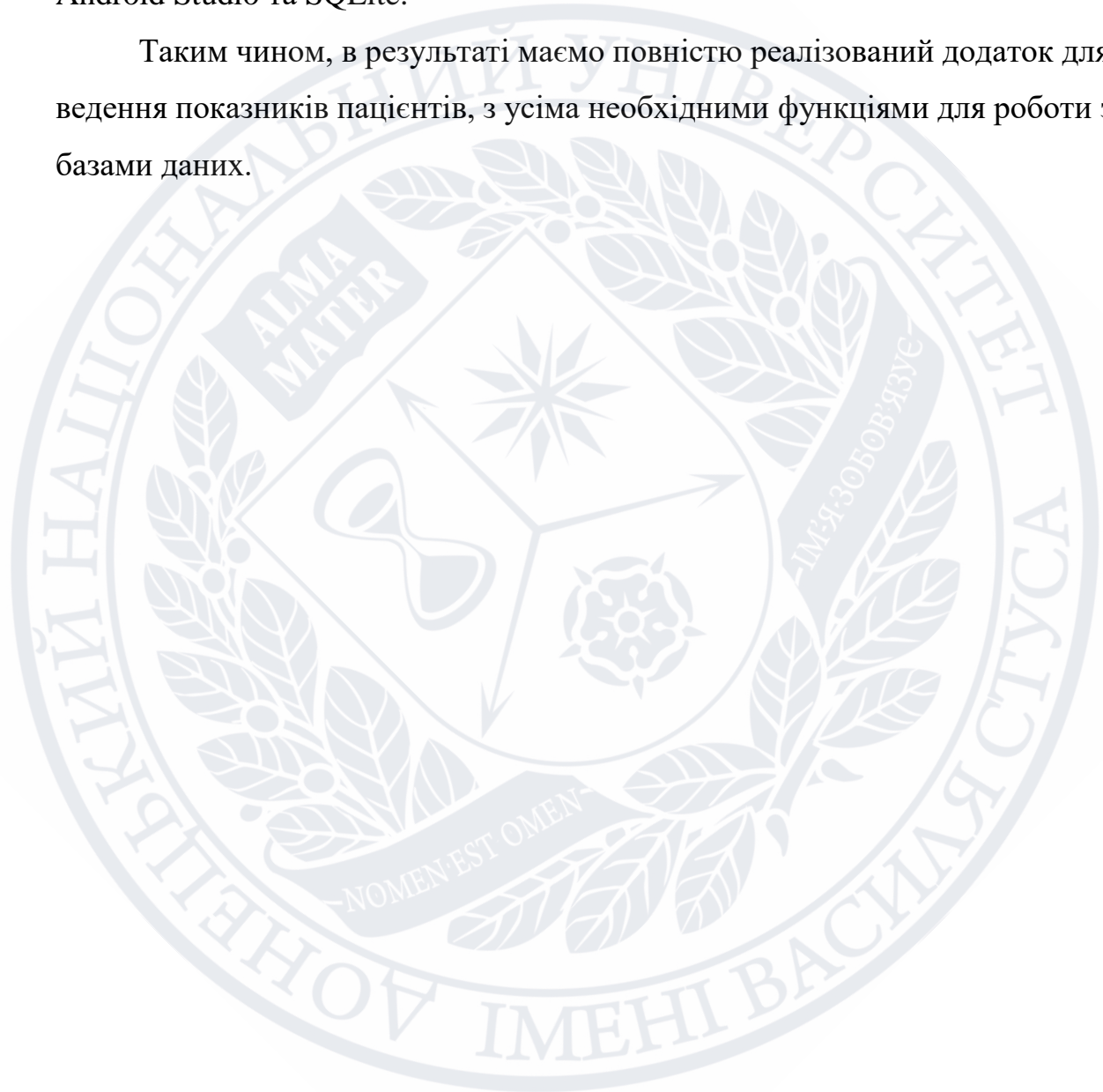
Додаток був виконаний згідно умов поставлених науковим викладачем. В ньому реалізована база даних, яка містить в собі дві таблиці: для пацієнтів та для прийомів. В другій таблиці містяться результати обстежень пацієнтів. Ці таблиці зв'язані між собою по ПІБ, так як в другій таблиці ПІБ вибирається з випадуючого списку, який імпортується з першої таблиці. В кожену таблицю можна додавати дані, редагувати їх та видаляти. Дані відображаються по-рядково, тобто на одній сторінці відображається один рядок, між якими можна перемикається за допомогою кнопок навігації. Проводиться невеликий аналіз даних: небезпечні показники для здоров'я підсвічуються червоним кольором. Всі дані зберігаються на диск телефону, а

також їх можна зберегти в CSV-файл та відправити по електронній пошті на вказані адреси.

Під час роботи були вивчені існуючі аналоги додатку та підтверджено актуальність розробки програм в цій сфері.

Для розробки додатку були використані наступні інструменти: Java, Android Studio та SQLite.

Таким чином, в результаті маємо повністю реалізований додаток для ведення показників пацієнтів, з усіма необхідними функціями для роботи з базами даних.



СПИСКИ ВИКОРИСТАНИХ ПОСИЛАНЬ

1. https://ru.wikipedia.org/wiki/Android_Studio
2. <https://developer.android.com/studio>
3. <http://web.spt42.ru/index.php/что-такое-android-studio>
4. <https://metanit.com/java/android/1.1.php>
5. <https://habr.com/ru/post/433604/>
6. <https://sqliteonline.com/>
7. <https://ru.wikipedia.org/wiki/SQLite>
8. <https://habr.com/ru/post/149356/>
9. <https://www.sqlite.org/index.html>
10. <https://proglib.io/p/sqlite-tutorial>
11. <https://wiki.dieg.info/sqlite>
12. <http://developer.alexanderklimov.ru/android/sqlite/android-sqlite.php>
13. <https://lecturesdb.readthedocs.io/databases/sqlite.html>
14. <https://ru.wikipedia.org/wiki/Java>
15. <https://habr.com/ru/hub/java/>
16. <https://metanit.com/java/tutorial/>
17. https://javarush.ru/quests/QUEST_JAVA_SYNTAX
18. <https://aws.amazon.com/ru/sdk-for-java/>
19. <https://www.dr-hempel-network.com/digital-health-technology/top-10-mobile-apps-for-personal-medical-records/>
20. https://play.google.com/store/apps/details?id=com.cliniconline&hl=en_US&gl=US
21. <https://www.onlinedoctor.com/15-best-online-medical-apps-that-make-personal-health-easier/>
22. <https://www.apple.com/healthcare/health-records/>
23. <https://www.javatpoint.com/how-to-send-email-in-android-using-intent>
24. https://www.tutorialspoint.com/android/android_sending_email.htm

25. <https://ru.wikipedia.org/wiki/CSV>
26. <https://www.bigcommerce.com/ecommerce-answers/what-csv-file-and-what-does-it-mean-my-ecommerce-business/>
27. <https://tallanto.com/ru/chto-takoe-csv-fayl-i-kak-ego-preobrazovat-v-excel>



Декларація щодо унікальності текстів роботи та невикористання
матеріалів інших авторів без посилань

Шевчук Юрій Анатолійович

Прізвище, ім'я, по батькові

Інформаційних і прикладних технологій

Факультет

122 «Комп'ютерні науки»

Шифр і назва спеціальності

«Сучасні інформаційні технології та програмування»

Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Розробка додатку для ведення стану здоров'я пацієнта» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно». дата підпи

дата

підпис