Державний вищий навчальний заклад

«Чернівецький політехнічний коледж»

Циклова комісія інженерії програмного забезпечення

**КУРСОВИЙ ПРОЕКТ**

з навчальної дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Запис на ремонт авто»

Студента IІІ курсу 531 групи

спеціальності 121

«Інженерія програмного забезпечення»

Швеця Є. В.

Керівник Гуменна Т.М.

м. Чернівці – 2020 рік

**Зміст**

[Вступ 3](#_Toc41594721)

[1 Обґрунтування необхідності розробки 4](#_Toc41594722)

[1.1 Опис предметної області 4](#_Toc41594723)

[1.2 Постановка задачі 5](#_Toc41594724)

[1.3 Технічне завдання на розробку 5](#_Toc41594725)

[2 Технічний проект 7](#_Toc41594726)

[2.1 Об’єктна модель 7](#_Toc41594727)

[2.2 Моделювання даних 7](#_Toc41594728)

[2.3 Функціональне моделювання 8](#_Toc41594729)

[3 Робочий проект 10](#_Toc41594730)

[3.1 Засоби розробки 10](#_Toc41594731)

[3.2 Інтерфейс програмного забезпечення 10](#_Toc41594732)

[3.3 Розробка документів на супроводження програмного забезпечення 11](#_Toc41594733)

[Висновки 22](#_Toc41594734)

[Перелік джерел 23](#_Toc41594735)

[Додатки 24](#_Toc41594736)

# Вступ

Метою курсового проекту є закріплення, поглиблення та узагальнення дисциплін «Об'єктно-орієнтоване програмування» та створення Android-додатку.

Інструмент для складання списку справ – відмінний помічник для зайнятих людей. Проте, не завжди один додаток справляється з усіма необхідностями. Прикладом може слідувати те, що майстер по ремонту автомобілів записує на обстеження автомобіль, і йому потрібно завжди записувати марку, модель автомобіля, номер телефона власника, і це забирає багато часу для створення цієї події.

Саме такий додаток і став об’єктом дослідження курсового проекту.

Інструмент для запису на обстеження автомобілів – це мобільний додаток, який об’єднує в собі створення робочого графіку для майстра у вигляді події, створення замовлень для обрахунку доходів та ведення статистики. Система стає зручним додатком для нагадуванням та спостереження за доходами. Програмний продукт повинен містити зрозумілий інтерфейс та реалізовані швидкодіючі алгоритми для розв’язання певних дій.

Предметом дослідження є розгляд описаних вище функцій додатку.

Для розробки проекту було вибрано середовище розробки Android Studio.

Звіт складається з трьох розділів. Перший розділ пояснювальної записки містить дослідження предметної області, постановку задачі та технічне завдання на розробку.

У другому розділі: об’єктна модель, моделювання даних, функціональне моделювання.

У третьому розділі пояснювальної записки: засоби розробки, інтерфейс програмного забезпечення, розробка документів на супроводження програмного продукту.

# 1 Обґрунтування необхідності розробки

## Опис предметної області

«Нагадування» – це система, яка може об’єднувати в собі деякий фунціонал створення подій, і керувати ними одним натисканням кнопки.

Функції, які реалізовані в «Нагадування»:

* створення подій – подія характеризується такими властивостями: дата, час, пріоритет, повтор.
* редагування подій – коригування властивостей та видалення події.
* перегляд історії події – всі події, які створював користувач.

В наш час існує чимало додатків, які реалізують «Нагадування». Серед них Todoist, Any.do, To Do Reminder with Alarm та інші.

Todoist - Це, скоріше, інструмент для складання списку справ, ніж нагадувань, проте він стане відмінним помічником для зайнятих людей. Додаток завойовує користувачів своїм стильним інтерфейсом і функціональністю. Воно відмінно працює і, крім того, синхронізується з ПК за допомогою розширення Chrome або автономного додатки Windows. При цьому працювати можна навіть в офлайн-режимі. Нажаль цей додаток є платний.

Any.do - багато в чому схоже на Тудуіст, починаючи з реєстрації та закінчуючи преміум-функціями. Однак є і принципові відмінності. В першу чергу, це призначений для користувача інтерфейс і то, як ви взаємодієте з додатком. У Any.do відображаються всі події: сьогоднішні, завтрашні, майбутні і без термінів. Таким чином ви відразу бачите загальну картину того, що належить зробити.

To Do Reminder with Alarm - вузьконаправлене додаток, призначене саме для створення нагадувань. Найбільш корисні функції: голосове введення Google , Можливість налаштувати нагадування за якийсь час до початку події, автоматичне додавання днів народження друзів з профілів Facebook ,

поштового аккаунта і контактів, створення нагадувань для інших людей шляхом відправки на пошту або в додаток (якщо воно встановлено у адресата).

Провівши огляд, було виявлено одну відмінностей у дадатків для нагадуванням між тим, що ми хочемо реалізувати. Це охопленість, ми реалізовуємо для майстрів по ремонту автомобілів з зручнішим для них інтерфейсом та автозаповненням полів.

Отже, вивчивши предметну область, ми дізналися, що дадатки для нагадування подібні, але створюючи додатки для окремих груп працівників, ми спрощуємо їм життя.

## 1.2 Постановка задачі

Потрібно розробити програмний продукт на платформі Android для створення подій по ремонту авто. В програмі потрібно реалізувати вибір марки та моделі автомобіля з випадаючого меню. Потрібно створити відображення подій у вигляді списку та календаря, добавити можливіть редагувати події, змінювати дані користувачів. Зробити можливість вибору номера користувача для створення подій з списку контактів або зі списку останніх набраних номерів.

В загальному – потрібно розробити програмний продукт для швидкого створення подій для майстра по ремонту авто. Додаток повинен мати зручний та інтуїтивний інтерфейс.

## 1.3 Технічне завдання на розробку

Android – додаток «Запис на ремонт авто» (далі – Проект), створюється з метою забезпечення автоматизації роботи майстра по ремонту авто з електронним календарем для створення подій, а саме спрощення заповнення інформації події, зручному відображенню подій та можливості взаємодії додатка з іншими системними додатками.

Замовником проекту визначено ФОП «Поплавський К.» (далі Замовник).

Термін виконання Проекту, це термін розробки курсового проекту, але не повинен перевищувати 6 місяців.

Результатом проекту є готовий додаток та документація на супроводження програмного продукту.

Для досягнення мети Проекту, а саме спрощення створення подій, відображення їх у списку та взаємодії з іншими системними додатками, розробка призначена для:

* надання швидкого доступу до перегляду подій;
* зменшення часу заповнення форми події.

Користувачами додатку будуть майстри по ремонту авто, яким потрібно швидко створити події на певну дату та при створенні події подивитись чи занята дата, на яку хочуть створити подію.

Потрібно розробити програмний продукт, за допомогою якого користувачі (майстри) зможуть створювати події, редагувати їх та переглядати у різних форматах (список чи календар).

Додаток повинен включати базу даних для зберігання даних події, замовників, марок та моделей авто.

Функціонал додатку:

* створення подій, передбачити створення подій на певну дату;
* перегляд подій, використовуючи списки та календар;
* редагування даних;
* заповнення форми використовуючи випадаючі меню;
* отримання номера з системного додатку телефона «Контакти».

Проект повинен відповідати наступним вимогам:

* інтерфейс українською мовою;
* захист від дурня;
* додаток повинен працювати від версії 6.0 Marshmallow.

# 2 Технічний проект

## 2.1 Об’єктна модель

Проаналізувавши предметну область визначив, що для маніпулювання даними, які розміщені в базі даних потрібно розробити клас з статичними методами. Використовуючи цей клас, ми легко зможемо маніпулювати даними в базі даних.

У аналогічних додатках використовується або стандартні списки, або вони взагалі не використовуються. У проекті були створені адаптери, які реалізовують створення користувацьких списків, над якими можна виконувати різний маніпуляції.

Так як база даних об’єктно – орієнтована, було створено класи, які розширюють клас «RealmObject».

У майбутньому планується реалізувати налаштування додатку та налаштування даних користувача додатку, де будуть реалізовані і абстрактні класи і класи-наслідники.

Діаграма класів відсутня через те, що при створенні проекту не використовувалось наслідування від створених власноруч класів.

## 2.2 Моделювання даних

Для роботи з об’єктною базою даних використовував систему Realm.

Realm – це система управління об'єктною базою даних з відкритим вихідним кодом, спочатку для мобільних пристроїв, також доступна для таких платформ, як Xamarin або React Native і інших, в тому числі для настільних додатків, і ліцензується на умовах ліцензії Apache.

Підключивши систему до додатку ми зможемо створювати об’єктну модель. Для створення об’єктної моделі потрібно визначити свій модельний клас, розширивши RealmObject. Лістинг коду модельних класів розміщені в додатку А.

На рисунку 1 зображена концептуальна модель даних.

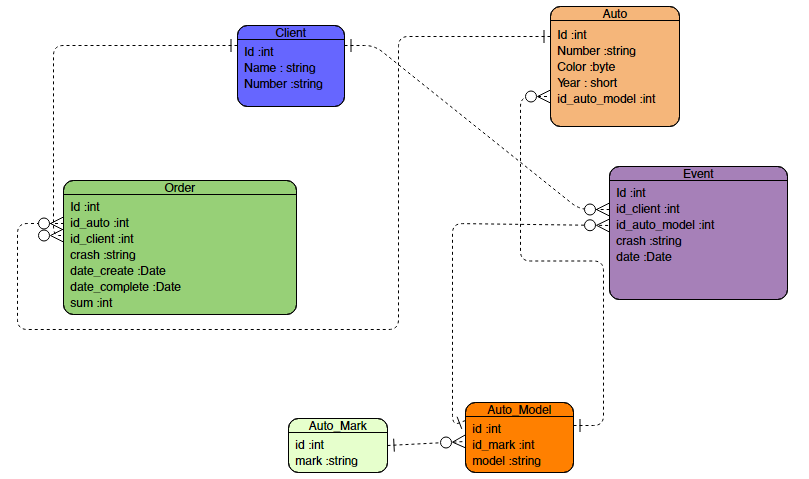


Рисунок 1 – Концептуальна модель даних

Вищезазначені класи завжди розширюють RealmObject, та мають стандартні гетери та сетери.

Realm дає можливість легко працювати з базою даних, а за допомогою транзакцій, база гарантує цілісне збереження даних.

Використовуючи Realm дуже легко користуватись вибіркою, пошуком, оновленням та видаленням даних. Приклади роботи з Realm розміщені у додатку A.

Отже, у додатку використовувалась об’єктна база даних.

## 2.3 Функціональне моделювання

Першим кроком до визначення функціональності системи є аналіз і моделювання вимог до неї. При цьому досягається згода між розробниками, замовниками і майбутніми користувачами стосовно функцій системи, поліпшується розуміння розробниками їх поведінки, обмежується системна функціональність, створюється базис для планування розробки проекту, визначається користувацький інтерфейс.

Можливості, які будуть доступні користувачу та базі даних описані на діаграмі прецедентів, що зображена на рисунку 2.

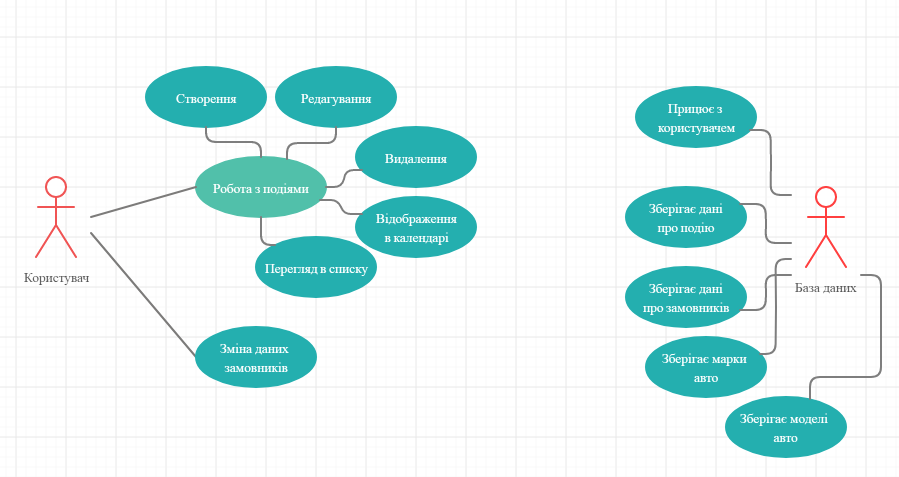


Рисунок 2 – Діаграма прецедентів

Для кращого розуміння вза’ємодії об’єктів за часом створюються діаграми послідовності. На рисунку 3 зображена діаграма, яка відображує послідовність комунікації користувача з системою, що веде до відображення подій у списку або у календарі.

Дана послідовність завжди виконується при вході в додаток.

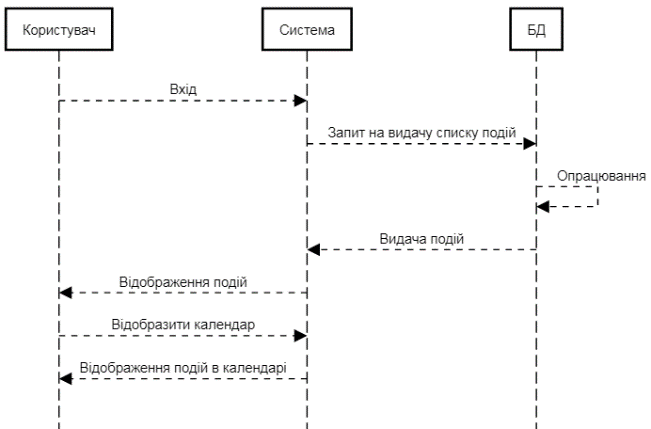


Рисунок 3 – Діаграма послідовності (відображення подій)

На рисунку 4 зображена діаграма послідовності «Створення подій».

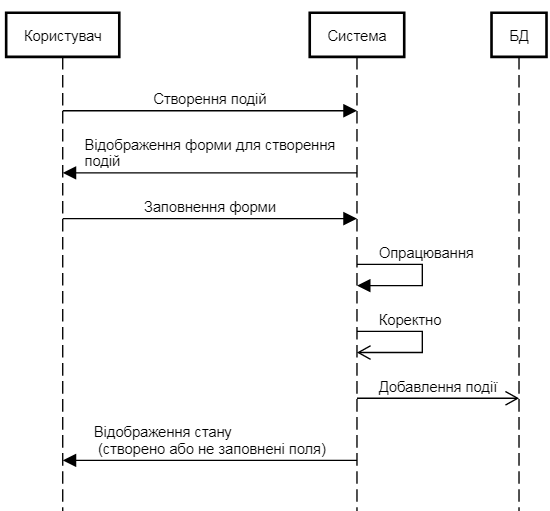


Рисунок 4 – Діаграма послідовності (створення подій)

# 3 Робочий проект

## 3.1 Засоби розробки

Бурхливий розвиток інформаційних технологій останнім часом призвів до того, що з'явилося багато нових пристроїв і технологій, таких, як планшети, смартфони, нетбуки, інші гаджети. Вони все більш міцно входять в наше життя і стають звичною справою. Лідируючої платформою серед подібних гаджетів на сьогоднішній день є ОС Android.

Android використовується на самих різних пристроях. Це і смартфони, і планшети, і телевізори, і смарт-годинник і ряд інших гаджетів. За різними підрахунками за 2020 рік цієї операційною системою користуються близько 86% власників смартфонів, а загальна кількість користувачів сматрфона на ОС Android оцінюється в 2 млрд.

Для програмування вибрав мову програмування Java.

Java є основою практично для всіх типів мережевих додатків і загальним стандартом для розробки і поширення вбудованих і мобільних додатків, ігор, веб-контенту та корпоративного програмного забезпечення. У світі налічується понад 9 мільйонів фахівців, які розробляють додатки на Java, яка дозволяє ефективно розробляти, впроваджувати і використовувати чудові додатки і послуги.

Від портативних комп'ютерів до центрів збору даних, від ігрових консолей до комп'ютерів, які використовуються для наукових розробок, від стільникових телефонів до мережі Інтернет - Java використовується всюди.

Також для Java є потужна документація для створення мобільних додатків на платформі Android Studio.

Отже, вибравши мову програмування Java працювати з платформою Android буде легше, так як підтримка та оновлення версій відбувається досить часто.

## 3.2 Інтерфейс програмного забезпечення

Під час розробки інтерфейсу необхідно врахувати наступні принципи:

* Використовувати стандартні, перевірені багатьма програмістами і користувачами інтерфейсні рішення;
* Інтерфейс повинен бути зручний, простий та зрозумілий. Слід врахувати, що всі дії повинні легко запам’ятовуватися і щоб вони не вимагали стомлюючих процедур;
* Дизайн інтерфейсу. Інтерфейс не повинен стомлювати зір, він повинен мати один стиль.

Для забезпечення діалогу між програмою та користувачем було використано спливаючі повідомлення Toast.

## 3.3 Розробка документів на супроводження програмного забезпечення

Для кращого розуміння програмного коду, при необхідності вдосконалення програмного забезпечення, створюють інструкцію програмісту, а для розуміння процесу експлуатації програмного забезпечення – інструкцію користувачу.

### 3.3.1 Інструкція програмісту

Для зручної роботи з базою даних було створено клас «MyRealm», в якому реалізовані статичні методи для добавлення, редагування, вибірку, видалення даних в базі даних. Лістинг класу розміщений у Додатку А.

Для роботи з xml – файлами також був створений клас з статичними методами «MySharedPreferences».

public class MySharedPreferences {

private static final String namePreferensec = "MyPref";//змінна, назва файлу, для зберігання даних у xml файлах

private static final String FIRST = "First"; //так би мовити теги, за якими буде зберігатися дані.

private static SharedPreferences sharedPreferences;

public static boolean isFirst(Context context){//метод перевіряє, чи користувач входить вперше, якщо це так, то метод повертає true, інакше false.

sharedPreferences = context.getSharedPreferences(namePreferensec,Context.MODE\_PRIVATE);

boolean ret = sharedPreferences.getBoolean(FIRST,false);

if(ret == false){//якщо перший раз, то встановлюємо, що вже не перший

setFirst(context);

}

return ret;

}

private static void setFirst(Context context){//метод, який встановлює, перше встановлення

sharedPreferences = context.getSharedPreferences(namePreferensec,Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putBoolean(FIRST,true);

editor.apply();

}}

Класи проекту, які мають у назві слово «Adapter» створені для розробки користувацького списку, в цих класах реалізовані методи розширені класом RecyclerView.Adapter та створений клас RecyclerView.ViewHolder.

public class SettingsAdapter extends RecyclerView.Adapter<SettingsAdapter.ViewHolder> {

private Context context;

private List<String> name\_settings;

private List<Integer> icon\_settings;

private static onClickListner ONCLICK;//оброботчик подій натиску на строку списку

public SettingsAdapter(Context context, List<String> name\_settings, List<Integer> icon\_settings) {

this.context = context;

this.name\_settings = name\_settings;

this.icon\_settings = icon\_settings;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {//встановлення користувацького шаблону строки списку

LayoutInflater inflater = LayoutInflater.from(context);

View view = inflater.inflate(R.layout.row\_settings,parent,false);

return new SettingsAdapter.ViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {//встановлення даних у поля

holder.name.setText(name\_settings.get(position));

holder.icon.setImageResource(icon\_settings.get(position));

}

@Override

public int getItemCount() {//кількість елементів в списку

return name\_settings.size();

}

public void setItemOnClickListener(onClickListner ONCLICK) {//всановлення події

SettingsAdapter.ONCLICK = ONCLICK; }

class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {

TextView name;

ImageView icon;

MaterialCardView cardView;

public ViewHolder(@NonNull View itemView) {

super(itemView);

name = (TextView) itemView.findViewById(R.id.row\_setting\_text);

icon = (ImageView) itemView.findViewById(R.id.row\_setting\_icon);

cardView = (MaterialCardView) itemView.findViewById(R.id.materialCard\_row\_settings);

cardView.setClickable(true);

cardView.setOnClickListener(this);}

@Override

public void onClick(View v) {//виклик події

ONCLICK.onItemClick(getAdapterPosition(),v);

}}

public interface onClickListner {//інтерфейс для реалізації обробки подій

void onItemClick(int position,View v);}}

У вищезазначеному класі реалізовані абстрактні методи реалізуючиш класів. Також для обробки даних списку було реалізовано подію OnClick.

Класи, які містять у назві «Activity» або «Fragment» реалізовують вікна додатку, в класах створені методи для роботи з цим вікном.

### 3.3.2 Інструкція користувачеві

Після запуску додатка відкривається головне вікно програми (рис. 5) – вікно, в якому розміщенні події, так як це перший запуск, головне вікно пусте.

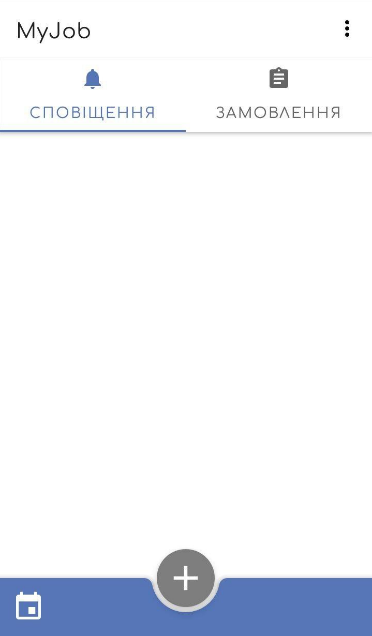


Рисунок 5 – Вікно відображення подій

Для перегляду подій у режимі «Календар» потрібно натиснути на кнопку календаря і відкриється наступне вікно. Сьогоднішня дата зафарбована в червоний колір, а числа місяця, які пройшли та числа наступного місяця зафарбовані в сірий колір.

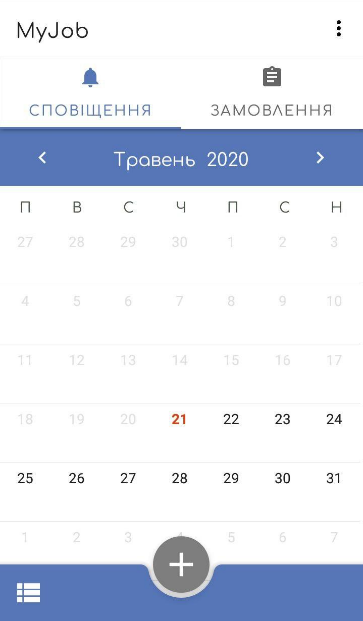


Рисунок 6 – Відображення подій в календарі

Для створення події на певне число місяця потрібно просто натиснути на дата і відкриється вікно для створення подій (рис. 7) , де вже автоматично заповнена дата.

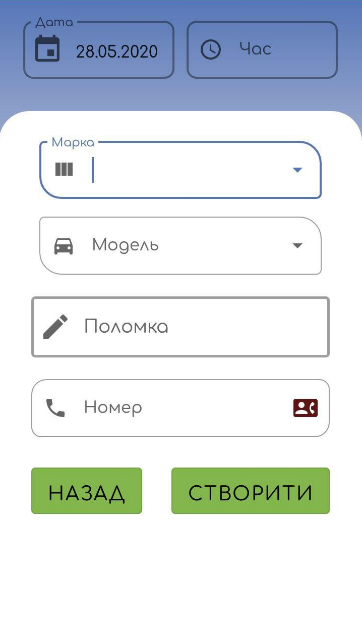


Рисунок 7 – Форма для створення подій з датою

Для того, щоб відкрити пусту форму для заповнення потрібно з головного вікна (див. рис. 5) натиснути кнопку добавлення. Приклад пустої форми для заповнення зображена на рисунку 8.

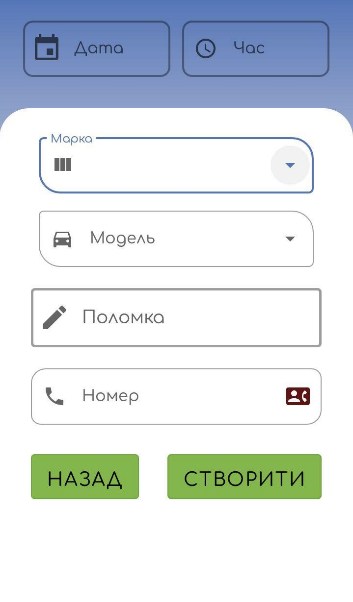


Рисунок 8 – Пуста форма створення подій

Для вибору дати/часу потрібно натиснути на ці поля і після чого відкриються діалогові вікна. На рисунку 9 зображене головне вікно для створення дати.

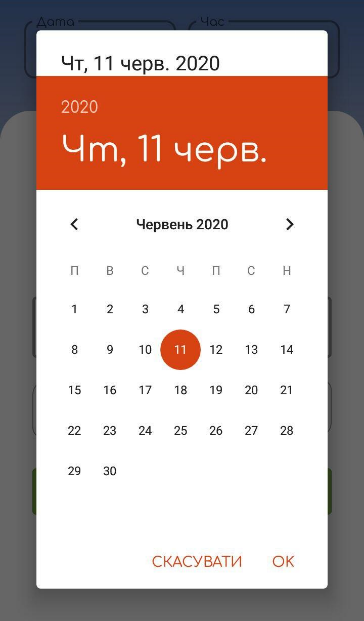


Рисунок 9 – Створення дати

На рисунку 10 зображене діалогове вікно для створення часу.

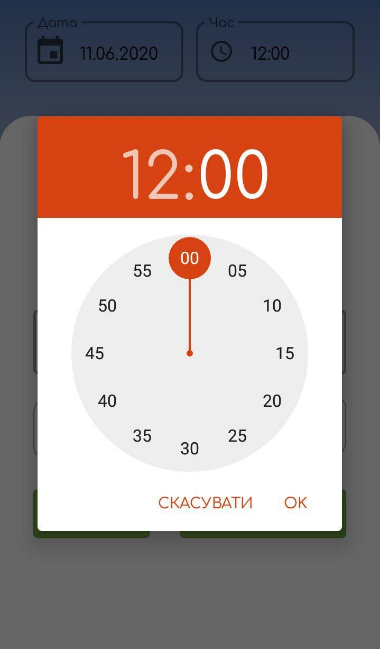


Рисунок 10 – Створення часу

Якщо відкривати діалогові вікна для створення дати та часу, то дані за замовчуванням будуть сьогоднішні, тобто дата та час системні.

Для вибору марки автомобіля потрібно вибрати її із випадаючого меню. На рисунку 11 зображений цей список.

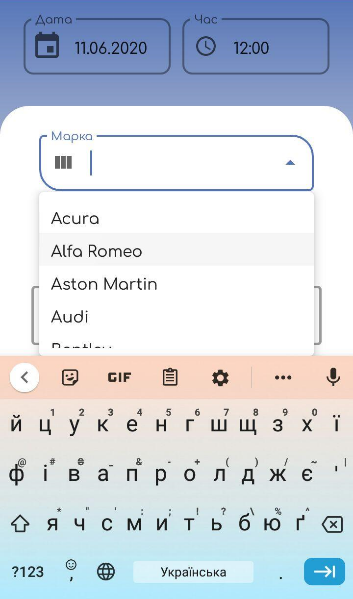


Рисунок 11 – Вибір марки автомобіля

Вибір моделі доступний після вибору марки. Приклад списку моделей зображений на рисунку 12.

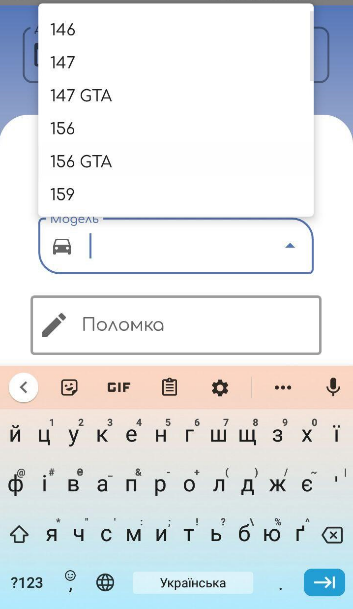


Рисунок 12 – Вибір моделі автомобіля

Наступне поле «Поломка» це поле користувач заповнює сам. Наступне поле після «Поломки» - «Номер» це поле користувач може заповнити сам, або натиснути на кнопку навпроти поля і відкриється вікно з контактами телефону, де користувач може вибрати номер телефону замовника, після чого система автоматично заповнить поле «Номер». На рисунку 13 зображена заповнена форма створення подій.

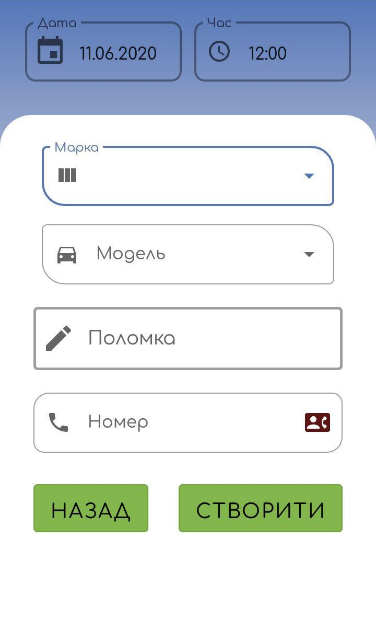


Рисунок 13 – Заповнена форма

Якщо якесь поле не заповнене і ми нажимаємо кнопку «Створити», то система нас повідомить про помилку (рис. 14).

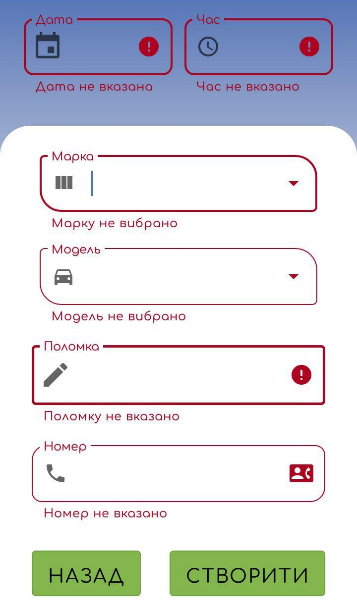


Рисунок 14 – Помилка при заповненні

Якщо всі дані введено коректно і ми натиснули на кнопку «Створити», подія створиться і відобразиться у списку подій на головному вікні. Відображена подія зображена на рисунку 15.

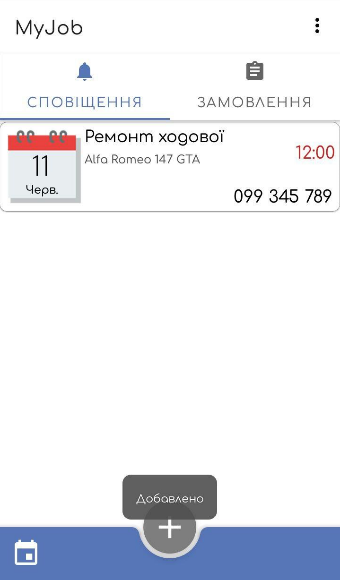


Рисунок 15 – Відображення подій

Для редагування/видалення подій потрібно довгим натиском утримувати події, після чого з’явиться меню для керування подіями.

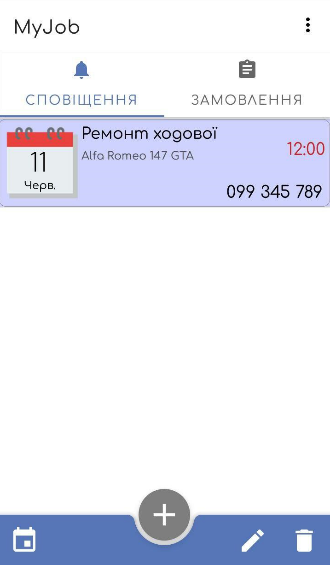


Рисунок 16 – Керування подіями

Якщо натиснути на іконку «Видалити» з’явиться діалогове вікно з підтвердженням, якщо підтвердити, то подія видалиться.

Якщо натиснути на іконку «Редагувати» то відкриється форма для створення подій в режимі «Оновлення даних» з заповненими полями. На рисунку 17 зображено оновлення даних події.

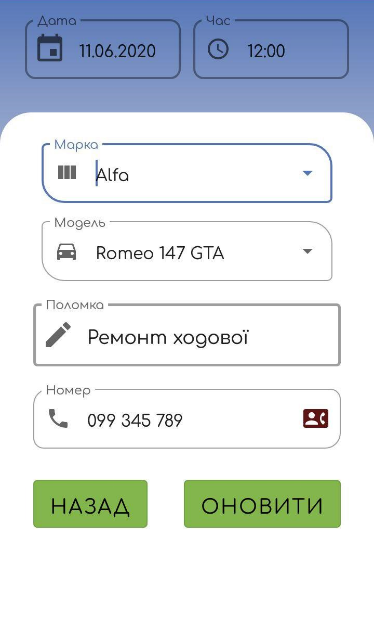


Рисунок 17 – Оновлення події

Якщо на певну дату є вже створені події, то у календарі дата події буде виділена (рис 18).

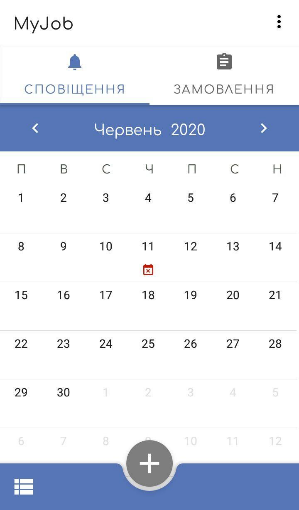


Рисунок 18 – Дата з існуючими подіями

Натиснувши на цю дату (див. рис. 18) відкриється вікно з усіма подіями на цю дату. Приклад зображений на рисунку 19.

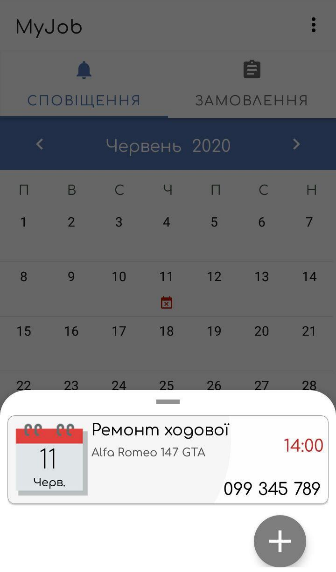


Рисунок 19 – Перегляд подій на вибрану дату

Також у цьому вікні є кнопка для створення подій на цю дату.

У головному меню є два пункти: перший для редагування користувачів, друге – перегляд всіх створених подій. Відкрити головне меню можна натиснувши на іконку «Меню» на головному вікні.

Меню зображене на рисунку 20.



Рисунок 20 – Головне меню

Вікно для редагування та перегляду користувачів зображене на рисунку 21.



Рисунок 21 – Вікно для перегляду користувачів

Натиснувши на іконку «Редагувати» відкривається діалогове вікно редагування. На рисунку 22 зображене це вікно.

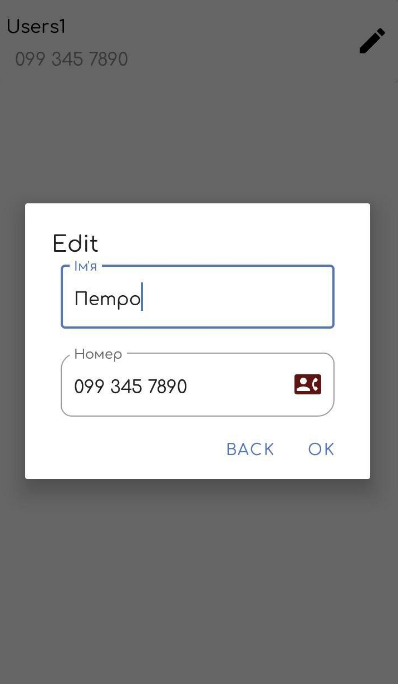


Рисунок 22 – Вікно редагування

Вікно для перегляду всіх подій зображена на рисунку 23.



Рисунок 23 – Вікно перегляду всіх подій

Отже, тест додатку пройшов успішно.

# Висновки

Результатом курсового проекту є android – додаток «MyJob» для нагадування робочого графіку майстру по ремонту автомобіля.

Під час розробки було враховано всі вимоги і стандарти. Інтерфейс доволі простий і зручний саме для майстрів, яким потрібно швидко заповнити поля і створити подію. На даний момент реалізовано лише створення подій по ремонту автомобіля. У майбутньому буде реалізовано ще створення замовлень та реалізовано push – сповіщення.

Було створено багато класів розширюваних ArrayAdapter для реалізації користувацького списку, також створено клас з статичними методами, за допомогою яких користування з базою даних та xml файлами відбувалося зручніше.

На стадії тестування програмного продукту було взято до уваги та усунуто всі можливі ситуації, які можуть зустрітися під час експлуатації програми, і тим самим посприяти на її некоректну роботу.

Дані зберігаються в об’єктній базі даних Realm та у xml файлах. Для бази даних були реалізовані класи, за допомогою яких було зручно користуватись з базою даних.

# Перелік джерел

1. Документація по Android Studio – [Електронний ресурс]. – Режим доступу: <https://developer.android.com/docs>
2. Допомога програмісту – [Електронний ресурс]. – Режим доступу: <https://stackoverflow.com/>
3. Вивчення мови Java – [Електронний ресурс]. – Режим доступу: <https://www.w3schools.com/java/default.asp>
4. Стандарт для створення технічого завдання - [Опис стандарту]. - <https://standards.ieee.org/standard/29148-2011.html>
5. Дизайн для додатка – [Електронний ресурс]. – Режим доступу: <https://material.io/design>

# Додатки

**Додаток А**

**Тексти програмного коду**

А.1 «Auto.java»

package com.yevhen.myjob.model;

import io.realm.RealmObject;

import io.realm.annotations.PrimaryKey;

public class Auto extends RealmObject {

@PrimaryKey

private int id;

private String number;

private int color;

private int year;

private int id\_auto\_model;

public int getId() {

return id;}

public void setId(int id) {

this.id = id;

}

public String getNumber() {

return number;

}

public void setNumber(String number) {

this.number = number;

}

public int getColor() {

return color;

}

public void setColor(int color) {

this.color = color;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

public int getId\_auto\_model() {

return id\_auto\_model;

}

public void setId\_auto\_model(int id\_auto\_model) {

this.id\_auto\_model = id\_auto\_model;

}}

А.2 «Auto\_Mark.java»

package com.yevhen.myjob.model;

import io.realm.RealmObject;

import io.realm.annotations.PrimaryKey;

public class Auto\_Mark extends RealmObject {

@PrimaryKey

private int id;

private String mark;

public Auto\_Mark(int id, String mark) {

this.id = id;

this.mark = mark;

}

public Auto\_Mark(){}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getMark() {

return mark;

}

public void setMark(String mark) {

this.mark = mark;

}}

А.3 «Auto\_Model.java»

package com.yevhen.myjob.model;

import io.realm.RealmObject;

import io.realm.annotations.PrimaryKey;

public class Auto\_Model extends RealmObject {

@PrimaryKey

private int id;

private int id\_mark;

private String model;

public Auto\_Model(int id,String model, int id\_mark) {

this.id = id;

this.id\_mark = id\_mark;

this.model = model;

}

public Auto\_Model() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getId\_mark() {

return id\_mark;

}

public void setId\_mark(int id\_mark) {

this.id\_mark = id\_mark;

}

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}}

А.4 «Client.java»

package com.yevhen.myjob.model;

import io.realm.RealmObject;

import io.realm.annotations.PrimaryKey;

public class Client extends RealmObject {

@PrimaryKey

private int id;

private String name;

private String number;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getNumber() {

return number;

}

public void setNumber(String number) {

this.number = number;

}}

А.5 «Event.java»

package com.yevhen.myjob.model;

import android.util.Log;

import java.util.Date;

import io.realm.RealmObject;

import io.realm.annotations.PrimaryKey;

public class Event extends RealmObject {

@PrimaryKey

private int id;

private int id\_client;

private int id\_auto\_model;

private String crash;

private Date date;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getId\_client() {

return id\_client;

}

public void setId\_client(int id\_client) {

this.id\_client = id\_client;

}

public int getId\_auto\_model() {

return id\_auto\_model;

}

public void setId\_auto\_model(int id\_auto\_model) {

this.id\_auto\_model = id\_auto\_model;

}

public String getCrash() {

return crash;

}

public void setCrash(String crash) {

this.crash = crash;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public void setDate(int year,int month,int day,int hour,int minute){

Log.i("Event\_",year+" "+month+" "+day+" "+hour+" "+minute+" rhfdg");

this.date = new Date();

this.date.setYear(year);

this.date.setMonth(month);

this.date.setDate(day);

this.date.setHours(hour);

this.date.setMinutes(minute);

this.date.setSeconds(0);

}}

А.6 «Order.java»

package com.yevhen.myjob.model;

import java.util.Date;

import io.realm.RealmObject;

import io.realm.annotations.PrimaryKey;

public class Order extends RealmObject {

@PrimaryKey

private int id;

private int id\_auto;

private int id\_client;

private String crash;

private Date date\_create;

private Date date\_complete;

private int sum;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getId\_auto() {

return id\_auto;

}

public void setId\_auto(int id\_auto) {

this.id\_auto = id\_auto;

}

public int getId\_client() {

return id\_client;

}

public void setId\_client(int id\_client) {

this.id\_client = id\_client;

}

public String getCrash() {

return crash;

}

public void setCrash(String crash) {

this.crash = crash;

}

public Date getDate\_create() {

return date\_create;

}

public void setDate\_create(Date date\_create) {

this.date\_create = date\_create;

}

public Date getDate\_complete() {

return date\_complete;

}

public void setDate\_complete(Date date\_complete) {

this.date\_complete = date\_complete;

}

public int getSum() {

return sum;

}

public void setSum(int sum) {

this.sum = sum;

}}

А.7 «MyRealm.java»

package com.yevhen.myjob;

import android.os.strictmode.CleartextNetworkViolation;

import android.util.Log;

import com.yevhen.myjob.model.Auto\_Mark;

import com.yevhen.myjob.model.Auto\_Model;

import com.yevhen.myjob.model.Client;

import com.yevhen.myjob.model.Event;

import java.sql.Array;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Date;

import java.util.GregorianCalendar;

import java.util.List;

import java.util.Locale;

import io.realm.Realm;

import io.realm.RealmResults;

import io.realm.Sort;

public class MyRealm {

public static void getEvent(Realm realm, List<Date> date, List<Integer> id, List<String> crash, List<String> auto, List<String> number,boolean isAll){

RealmResults<Event> events = null;

Calendar calendar = Calendar.getInstance();

Date today = new Date();

today.setYear(calendar.get(Calendar.YEAR));

today.setMonth(calendar.get(Calendar.MONTH));

today.setDate(calendar.get(Calendar.DAY\_OF\_MONTH));

today.setHours(calendar.get(Calendar.HOUR\_OF\_DAY));

today.setMinutes(calendar.get(Calendar.MINUTE));

if(isAll==true) {

events=realm.where(Event.class).findAll().sort("date", Sort.ASCENDING);

}else{

events = realm.where(Event.class).greaterThan("date",today).findAll().sort("date", Sort.ASCENDING);

}

Log.i("Event\_",events.asJSON().toString());

Date dat;

for(Event e:events){

dat = new Date();

dat.setYear(e.getDate().getYear());

dat.setMonth(e.getDate().getMonth());

dat.setDate(e.getDate().getDate());

dat.setHours(e.getDate().getHours());

dat.setMinutes(e.getDate().getMinutes());

date.add(dat);

id.add(e.getId());

crash.add(e.getCrash());

auto.add(MyRealm.getModel(realm,e.getId\_auto\_model()));

number.add(MyRealm.getPhoneNumber(realm,e.getId\_client()));}}

public static void getEvent(Realm realm, List<Date> date, List<Integer> id, List<String> crash, List<String> auto, List<String> number,Calendar calendar){

RealmResults<Event> events = null;

Date today\_from = new Date();

today\_from.setYear(calendar.get(Calendar.YEAR));

today\_from.setMonth(calendar.get(Calendar.MONTH));

today\_from.setDate(calendar.get(Calendar.DAY\_OF\_MONTH));

today\_from.setHours(0);

today\_from.setMinutes(0);

Date today\_to = new Date();

today\_to.setYear(calendar.get(Calendar.YEAR));

today\_to.setMonth(calendar.get(Calendar.MONTH));

today\_to.setDate(calendar.get(Calendar.DAY\_OF\_MONTH));

today\_to.setHours(23);

today\_to.setMinutes(59);

events = realm.where(Event.class).between("date",today\_from,today\_to).findAll().sort("date");

Date dat;

for(Event e:events){

dat = new Date();

dat.setYear(e.getDate().getYear());

dat.setMonth(e.getDate().getMonth());

dat.setDate(e.getDate().getDate());

dat.setHours(e.getDate().getHours());

dat.setMinutes(e.getDate().getMinutes());

date.add(dat);

id.add(e.getId());

crash.add(e.getCrash());

auto.add(MyRealm.getModel(realm,e.getId\_auto\_model()));

number.add(MyRealm.getPhoneNumber(realm,e.getId\_client()));}}

public static List<Date> getEventForCalendar(Realm realm){

RealmResults<Event> events = null;

Calendar calendar = Calendar.getInstance();

Date today = new Date();

today.setYear(calendar.get(Calendar.YEAR));

today.setMonth(calendar.get(Calendar.MONTH));

today.setDate(calendar.get(Calendar.DAY\_OF\_MONTH));

today.setHours(calendar.get(Calendar.HOUR\_OF\_DAY));

today.setMinutes(calendar.get(Calendar.MINUTE));

events = realm.where(Event.class).greaterThan("date",today).findAll().sort("date",Sort.ASCENDING);

List<Date> dates = new ArrayList<>();

for (Event e:

events) {

dates.add(e.getDate());

}

return dates;

}

public static void deleteEvent(Realm realm,int id){

realm.beginTransaction();

Event ev = realm.where(Event.class).equalTo("id",id).findFirst();

ev.deleteFromRealm();

realm.commitTransaction();

}

public static Event getEvent(Realm realm, int id){

Event event = realm.where(Event.class).equalTo("id",id).findFirst();

return event;

}

public static boolean upgradeEvent(Realm realm, final Event event){

realm.executeTransactionAsync(new Realm.Transaction() {

@Override

public void execute(Realm realm) {

Event event1 = realm.where(Event.class).equalTo("id", event.getId()).findFirst();

event1.setDate(event.getDate());

event1.setId\_client(event.getId\_client());

event1.setCrash(event.getCrash());

event1.setId\_auto\_model(event.getId\_auto\_model());

}

}, new Realm.Transaction.OnSuccess() {

@Override

public void onSuccess() {

TEMP = 1;

}

}, new Realm.Transaction.OnError() {

@Override

public void onError(Throwable error) {

TEMP = 0;

}

});

if(TEMP ==1)return true;

else return false;

}

public static String getModel(Realm realm, int id\_model){

Auto\_Model model = realm.where(Auto\_Model.class).equalTo("id",id\_model).findFirst();

Auto\_Mark mark = realm.where(Auto\_Mark.class).equalTo("id",model.getId\_mark()).findFirst();

String auto = mark.getMark()+" "+model.getModel();

if(auto.length()==0)auto="Невизначено";

return auto;

}

public static String getPhoneNumber(Realm realm,int id\_Client){

Client client = realm.where(Client.class).equalTo("id",id\_Client).findFirst();

String number = client.getNumber();

if(number.length()==0) number = "Невизначено";

return number;

}

public static boolean addEvent(Realm realm, final Event event){

realm.beginTransaction();

realm.insert(event);

realm.commitTransaction();

Event event1 = realm.where(Event.class).equalTo("id",event.getId()).findFirst();

if(event1!=null)

{

return true;

}

return false;

}

public static int getMaxIdEvent(Realm realm){

Number id\_;

int id;

id\_ = realm.where(Event.class).max("id");

id = (id\_ == null) ? 1 : id\_.intValue() + 1;

return id;

}

public static int getClientId(Realm realm, String number,String name\_number){

Client client=null;

client = realm.where(Client.class).equalTo("number", number).findFirst();

Number id\_;

int id= 0;

if(client == null){

id\_ = realm.where(Client.class).max("id");

id = (id\_ == null) ? 1 : id\_.intValue() + 1;

client = new Client();

client.setId(id);

if(name\_number !=null) client.setName(name\_number);

else client.setName("Users"+ id);

client.setNumber(number);

realm.beginTransaction();

realm.insert(client);

realm.commitTransaction();

}else return client.getId();

return id;

}

public static int getAutoModelId(Realm realm,String val){

Auto\_Model model = realm.where(Auto\_Model.class).equalTo("model",val).findFirst();

if(model!=null){

return model.getId();

}else return 0;

}

public static void getClient(Realm realm,List<String> name,List<String> number){

RealmResults<Client> clients = realm.where(Client.class).findAll().sort("name");

for (Client c:clients) {

name.add(c.getName());

number.add(c.getNumber());

}

}

public static void upgradeClient(Realm realm, final String number, final String new\_name, final String new\_number){

realm.executeTransactionAsync(new Realm.Transaction() {

@Override

public void execute(Realm realm) {

Client client = realm.where(Client.class).equalTo("number",number).findFirst();

client.setName(new\_name);

client.setNumber(new\_number);

}

}, new Realm.Transaction.OnSuccess() {

@Override

public void onSuccess() {

}

}, new Realm.Transaction.OnError() {

@Override

public void onError(Throwable error) {

}

});

}

static int TEMP =0;

public static int deleteClient(Realm realm, final String number){

TEMP = 0;

boolean b = false;

Client client = realm.where(Client.class).equalTo("number",number).findFirst();

RealmResults<Event> events = realm.where(Event.class).greaterThan("date",Calendar.getInstance().getTime()).findAll();

for (Event ev:

events) {

if(ev.getId\_client() == client.getId()){ b = false;

break;

}

else b = true;

}

if(b==true){

realm.executeTransactionAsync(new Realm.Transaction() {

@Override

public void execute(Realm realm) {

Client client = realm.where(Client.class).equalTo("number",number).findFirst();

client.deleteFromRealm();

}

}, new Realm.Transaction.OnSuccess() {

@Override

public void onSuccess() {

TEMP =1;

}

}, new Realm.Transaction.OnError() {

@Override

public void onError(Throwable error) {

TEMP =-1;

}

});

return TEMP;

}

else{

return 0;

}

}

public static int getMarkId(Realm realm, String val){

Auto\_Mark mark\_ = realm.where(Auto\_Mark.class).equalTo("mark",val).findFirst();

if(mark\_!=null){

return mark\_.getId();

}else

return 0;}}

А.8 «MySharedPreferences.java»

package com.yevhen.myjob;

import android.content.Context;

import android.content.SharedPreferences;

import android.provider.ContactsContract;

public class MySharedPreferences {

private static Context context;

private static final String namePreferensec = "MyPref";

private static final String FIRST = "First";

private static SharedPreferences sharedPreferences;

public static boolean isFirst(Context context){

sharedPreferences = context.getSharedPreferences(namePreferensec,Context.MODE\_PRIVATE);

boolean ret = sharedPreferences.getBoolean(FIRST,false);

if(ret == false){

setFirst(context);

}

return ret;

}

private static void setFirst(Context context){

sharedPreferences = context.getSharedPreferences(namePreferensec,Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putBoolean(FIRST,true);

editor.apply();

}

}

А.9 «MyApplication.java»

package com.yevhen.myjob;

import android.app.Application;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.os.Build;

import android.util.Log;

import io.realm.Realm;

import io.realm.RealmConfiguration;

public class MyApplication extends Application {

public static String CHANNEL\_ID = "NotiID";

@Override

public void onCreate() {

super.onCreate();

Realm.init(this);

RealmConfiguration config = new RealmConfiguration.Builder().name("myrealm.realm").build();

Realm.setDefaultConfiguration(config);

//Якщо ми зайшли перший раз, або очистили дані

if(!MySharedPreferences.isFirst(this)){

MyRealm.setAuto\_Model\_and\_Mark(Realm.getDefaultInstance());

}

createNotificationChannel();

}

private void createNotificationChannel() {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {

NotificationChannel serviceChannel = new NotificationChannel(

CHANNEL\_ID,

"Example Service Channel",

NotificationManager.IMPORTANCE\_DEFAULT

);

NotificationManager manager = getSystemService(NotificationManager.class);

manager.createNotificationChannel(serviceChannel);

}

}

}

А.10 «AndroidManifest.xml»

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.yevhen.myjob">

<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />

<application

android:name=".MyApplication"

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher\_1"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".SettingsActivity"></activity>

<activity

android:name=".NotificationActivity"

android:windowSoftInputMode="adjustPan" />

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<service

android:name=".JobSchedulerService"

android:permission="android.permission.BIND\_JOB\_SERVICE"></service>

<meta-data

android:name="preloaded\_fonts"

android:resource="@array/preloaded\_fonts" />

</application>

</manifest>

А.11 «Notification\_Recycler\_Adapter.java»

package com.yevhen.myjob;

import android.content.Context;

import android.graphics.Color;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.view.ViewStructure;

import android.widget.AdapterView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.card.MaterialCardView;

import com.yevhen.myjob.model.Event;

import java.lang.reflect.Array;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import java.util.zip.Inflater;

public class Notification\_Recycler\_Adapter extends RecyclerView.Adapter<Notification\_Recycler\_Adapter.MyViewHolder> {

private Context context;

private List<Date> date = new ArrayList<Date>();

private List<Integer> id = new ArrayList<Integer>();

private List<String> crash = new ArrayList<String>();

private List<String> auto = new ArrayList<String>();

private List<String> number = new ArrayList<String>();

private static onClickListner onClickListener;

public static boolean pointer = false;

public Notification\_Recycler\_Adapter(Context context, List<Date> date, List<Integer> id, List<String> crash, List<String> auto, List<String> number) {

this.context = context;

this.date = date;

this.id = id;

this.crash = crash;

this.auto = auto;

this.number = number;

}

@NonNull

@Override

public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

LayoutInflater inflater = LayoutInflater.from(context);

View view = inflater.inflate(R.layout.row\_notification,parent,false);

return new MyViewHolder(view,onClickListener);}

@Override

public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {

String[] month = context.getResources().getStringArray(R.array.month\_);

holder.mycrash.setText(crash.get(position));

holder.mynumber.setText(number.get(position));

holder.myauto.setText(auto.get(position));

holder.mytime.setText( zerofill(date.get(position).getHours())+":"+zerofill(date.get(position).getMinutes()));

holder.myday.setText(zerofill(date.get(position).getDate()));

holder.mymonth.setText(month[date.get(position).getMonth()]);

holder.cardView.setTag(R.id.Tag\_ID,id.get(position));

holder.cardView.setTag(R.id.Tag\_COLOR,"1");}

@Override

public int getItemCount() {

return id.size();}

public class MyViewHolder extends RecyclerView.ViewHolder implements View.OnLongClickListener, View.OnClickListener {

TextView mycrash;

TextView myauto;

TextView mynumber;

TextView mytime;

TextView myday;

TextView mymonth;

MaterialCardView cardView;

onClickListner adapter;

public MyViewHolder(@NonNull View itemView,onClickListner adapter) {

super(itemView);

this.adapter = adapter;

mycrash = itemView.findViewById(R.id.crash\_row\_notification);

myauto = itemView.findViewById(R.id.auto\_row\_notification);

mynumber = itemView.findViewById(R.id.number\_row\_notification);

mytime = itemView.findViewById(R.id.date\_time\_row\_notification);

myday = itemView.findViewById(R.id.day\_row\_notification);

mymonth = itemView.findViewById(R.id.month\_row\_notification);

cardView = itemView.findViewById(R.id.materialCard\_row\_notification);

cardView.setLongClickable(true);

cardView.setOnLongClickListener(this);

cardView.setClickable(true);

cardView.setOnClickListener(this);

//itemView.setTag("1");}

@Override

public boolean onLongClick(View v) {

adapter.onItemLongClick(getAdapterPosition(),v,pointer);

return false;

}

@Override

public void onClick(View v) {

adapter.onItemClick(getAdapterPosition(), v);

}

}

public void setOnItemClickListener(onClickListner onclicklistner) {

onClickListener = onclicklistner;

}

public interface onClickListner {

void onItemClick(int position,View v);

void onItemLongClick(int position, View v,boolean pointer);}

private String zerofill(int i){

return ((i<10)?"0":"")+i;}}

А.12 «Adapter\_Mark.java»

package com.yevhen.myjob;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.Filter;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import java.util.ArrayList;

import java.util.List;

public class Adapter\_Mark extends ArrayAdapter<String> {

List<String> array;

public Adapter\_Mark(@NonNull Context context, @NonNull List<String> countryList) {

super(context, 0, countryList);

array = new ArrayList<>(countryList);

}

@NonNull

@Override

public Filter getFilter() {

return countryFilter;

}

@NonNull

@Override

public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {

if (convertView == null) {

convertView = LayoutInflater.from(getContext()).inflate(

R.layout.row\_adapter, parent, false

);

}

TextView textViewName = convertView.findViewById(R.id.text\_view\_name);

String name = getItem(position);

if (name != null) {

textViewName.setText(name);

}

return convertView;

}

private Filter countryFilter = new Filter() {

@Override

protected FilterResults performFiltering(CharSequence constraint) {

FilterResults results = new FilterResults();

List<String> suggestions = new ArrayList<>();

if (constraint == null || constraint.length() == 0) {

suggestions.addAll(array);

} else {

String filterPattern = constraint.toString().toLowerCase().trim();

for (String item : array) {

if (item.toLowerCase().contains(filterPattern)) {

suggestions.add(item);

}

} }

results.values = suggestions;

results.count = suggestions.size();

return results;}

@Override

protected void publishResults(CharSequence constraint, FilterResults results) {

clear();

addAll((List) results.values);

notifyDataSetChanged();

}

@Override

public CharSequence convertResultToString(Object resultValue) {

return ((String) resultValue);

}};}