Міністерство освіти і науки України
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Кафедра прикладних інформаційних систем

122 Комп'ютерні науки
Освітня програма «Прикладне програмування»

Кваліфікаційна робота бакалавра

на тему:

Мобільний застосунок із планування бізнес-справ

Виконав студент 4 курсу групи ПП−41

_____
(Підпис)
Пащинський В.Д.
(прізвище, ім'я, по батькові)

_____

Керівник проф., д.т.н. Сайко В.Г.
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л. _____
(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цьому курсовому проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 року

Календарний план виконання роботи

№ п/п	Назва етапів бакалаврської роботи	Термін виконання етапів бакалаврської роботи	Відмітка про виконання
1.	Вибір теми та наукового керівника бакалаврської роботи	15.11.2019	
2.	Видача завдання бакалаврської роботи	22.11.2019	
3.	Настановча групова співбесіда з бакалаврської роботи	01.12.2019	
4.	Затвердження плану бакалаврської роботи	18.02.2020	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2020	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2020	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	10.04.2020	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2020	
9.	Подання роботи у першому варіанті	11.05.2020	
10.	Оформлення пояснювальної записки бакалаврської роботи	12.05.2020	
11.	Подання бакалаврської роботи на попередній захист	14.05.2020	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	29.05.2020	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	12.06.2020	
14.	Захист бакалаврської роботи	23.06.2020	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ЗМІСТ

АНОТАЦІЯ

До бакалаврської дипломної роботи Пащинського Володимира Денисовича на тему «Мобільний застосунок планування бізнес-справ»

Дана дипломна робота присвячена створенню застосунків та систем планування справ для мобільних пристроїв.

Для розробки оптимізованої системи було проведено дослідження всіх актуальних, на даний час, технологій, що призначен для створення мобільних застосунків під керуванням системи Android. Після досліджень було здійснено порівняння популярних планувальників задач, їх простота використання для звичайного користувача, швидкість роботи, додаткові можливості для планування.

Вибір мови програмування для дипломної роботи був зроблений по трьом основним критеріям: об’єм інформації про використання даної мови у тому числі для створення подібних застосунків у вільному доступі, легкість та зручність програмного супроводження системи та вирішення проблем які виникнуть при розробці за допомогою власних пошуків інформації або прямих запитань спеціалістам за допомогою відповідних сервісів.

В даній роботі приведені результати досліджень найпопулярніших планувальників, програмне супроводження застосунка та детальний опис використання основних розроблених функцій, компонентів.

Загальний обсяг роботи: 69 сторінок, 26 рисунків, 14 посилань.

Ключові слова: Android, ToDo app, система планування справ, мобільний застосунок.

SUMMARY

To the bachelor's thesis of Pashchynsky Vladimir Denisovich on the topic "Mobile application of business planning"

This thesis is devoted to the creation of applications and case planning systems for mobile devices.

To develop an optimized system, a study of all current technologies was conducted, which is designed to create mobile applications running Android. After the research, a comparison of popular task schedulers, their ease of use for the average user, speed, additional features for planning.

The choice of programming language for the thesis was made according to three main criteria: the amount of information about the use of this language, including to create similar applications in free access, ease and convenience of software support system and solving problems that arise during development through their own information retrieval or direct questions to specialists with the help of appropriate services.

This paper presents the results of research of the most popular planners, software support of the application and a detailed description of the use of the main developed functions and components.

Total volume of work: 69 pages, 26 figures, 14 links.

Keywords: Android, ToDo app, case planning system, mobile application.

ВСТУП

Сучасна людина щодня виконує безліч справ. Для ефективного використання свого часу необхідно заздалегідь розпланувати майбутні справи. Організатор часу – незамінний атрибут успішної людини. Звичка записувати і планувати справи міцно зміцнилася в нашому житті, різні замітки і нагадування робить абсолютно кожна людина.

Актуальність дослідження. Ні для кого не секрет, що для успішного бізнесу вкрай важливо ефективно розподіляти свій час. Будь-яка успішна компанія приділяє особливу увагу здійсненню планування і організації роботи співробітників. Для вирішення подібних завдань існують цілі системи, що дозволяють максимально оптимізувати роботу фірми. Це дуже зручно і ефективно. Однак, навіть в повсякденних, простих справах часто необхідно планувати власний час - це дозволяє максимально раціонально вирішувати завдання, завдяки чому залишається більше часу на відпочинок. Також різні нагадування допомагають не забути виконати будь-яку справу. Саме тому на сьогоднішній день так популярні різні сервіси для планування часу - web-сервіси, десктопні програми і мобільні додатки. У вирі життєвих подій і справ часто можна забути про запланованих спільно з кимось справах, якщо подібні завдань не розподілені у відповідному сервісі. Підсумовуючи вищесказане, планування справ є важливим завданням кожної людини. До того ж, необхідно в будь-який момент мати доступ до перегляду запланованих справ. В даний час практично кожна людина має смартфон, тому планувальник справ у вигляді мобільного додатка є найкращим вибором.

Метою дипломної роботи є розробка мобільного додатка для пристроїв з ОС Android для планування справ та заходів.

Завдання дослідження:

- Аналіз існуючих систем, що володіють схожим функціоналом;

- Визначення функціоналу програми на основі аналізу існуючих систем;
- Вивчення та вибір засобів для здійснення розробки мобільного додатка для ОС Android;
- Проектування розробки програми;
- Розробка програми для планування спільних справ.

Об'єктом дослідження у цій роботі є мобільний застосунок для планування справ, а ***предметом*** – системи контролю контентом.

РОЗДІЛ 1: АНАЛІЗ РИНКУ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

На сьогоднішній день, мабуть, кожна людина має мобільний телефон. Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій в останні роки. Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці і в будь-якій ситуації. Тому важливу роль грає користувач оперувати і ділитися даними з пристрою. Наприклад, одним з безлічі типів додатків для мобільних пристроїв є додатки для подорожей і туризму. Під час таких подорожей вони повинні краще працювати в офлайн-режимі, тому що то не завжди в будь-якому місці і в будь-якій ситуації, у вас буде доступ до інтернету або мобільного оператора.

Поміж того, людина щодня контактує з безліччю інших людей і йому просто необхідно постійно «бути на зв'язку». Саме для вирішення подібних завдань і використовуються мобільні додатки. Неможливо обійтися без додатків, що дозволяють легко спілкуватися - листуватися, телефонувати, щось планувати, здійснювати покупки та інше. Існують різні мобільні додатки для планування завдань. Досить поширені і популярні додатки, що дозволяють ефективно управляти своїм часом – як і додаток TDPlanner. Цей додаток дозволяє здійснювати планування справ, надаючи можливість повноцінно використовувати всі можливості для ефективного планування.

Однак, через великий функціонал цього додатка користувач може випробувати деякі труднощі в тому випадку, якщо вперше стикається з даними програмами. Зрозуміло, додаток TDPlanner є дуже зручним і функціональним, тому можна з упевненістю сказати, що користувач цього додатка досить швидко звикне до даної програми і буде успішно використовувати її. Не дивлячись на це, велика ймовірність того, що багато функцій даного додатки не будуть використовуватися у плануванні простих, повсякденних справ.

Підводячи підсумок, слід зазначити, що наданий додаток слід зробити максимально простим у використанні. Додаток має надавати можливості для зручного та ефективного планування справ.

1.1 Вибір операційної системи

В даний час існують наступні операційні системи для мобільних пристроїв:

- Windows Phone;
- Apple iOS;
- Android;
- Symbian;
- Blackberry OS.

На сьогоднішній день найпопулярнішими і затребуваними системами є Windows Phone, Apple IOS і Android, тому дані системи слід розглянути докладніше.

1.1.1 Операційна система Windows Phone

Дана операційна система розроблена компанією Microsoft, що є лідером на світовому ринку виробництва операційних систем. Операційна система Windows є найпоширенішою операційною системою для ПК. Саме тому Windows Phone, операційна система для мобільних пристроїв, дуже схожа на настільну версію, набула широкого поширення і величезного попиту. Однак, з кожним роком інтерес користувачів до даної системи стає все менше. Це пов'язано з досить високою вартістю пристроїв, що використовують Windows Phone, а також їх невеликим кількістю.

Рисунок 1.1 - Головний екран Windows Phone

Windows Phone була розроблена для заміни сімейства мобільних операційних систем Windows Mobile. Уперше вона була випущена в Європі, Сінгапурі, Австралії, Новій Зеландії, США, Канаді і Мексиці в 2010 році, а в Азії - в 2011 році. Перші Windows Phones були випущені компаніями HTC, Dell, Samsung і LG. Перші продажі споживчих пристроїв відбулися 21 жовтня 2010 року.

Windows Mobile поставлялася з базовим набором додатків, який був розроблений так, щоб бути схожим на настільну версію Windows. У комплект постачання також входив набір додатків, включаючи Internet Explorer Mobile, Windows Media Player і Microsoft Office Mobile. Спочатку облаштування Windows Mobile вимагали використання стилуса. Пізніше вони стали оснащуватися ємнісними сенсорними екранами.

Пристрої, призначені для роботи з Windows Mobile, не можуть працювати з програмами для новішою Windows Phone, оскільки вони не відповідають апаратним вимогам для цього потужнішого програмного

11 липня 2017 року компанія Microsoft оголосила про негайне припинення підтримки Windows Phone. Телефони Windows Phone можуть продовжувати працювати, але більше не матимуть права на оновлення програмного забезпечення або безпеки від Microsoft.

1.1.2 Операційна система Apple iOS

Перша версія мобільної операційної системи була побудована на тому ж ядрі Unix, що і Mac OS X. Глава Apple Стів Джобс, що представляв на презентації перший iPhone образно назвав систему iPhone OS портованої Mac OS на новий смартфон. Але з перших хвилин презентації стало ясно, що відмінності будуть колосальними. Яким би інноваційним не був iPhone на момент презентації, його функціональність була вкрай обмежена.

Рисунок 1.2 - Головний екран iOS

Система iOS компанії Apple дуже популярна на сьогоднішній день.Вона є дуже стабільною, «доброзичливою» і захищеною системою. Але, на жаль, в

деякому роді захищеною і від розробників. Справа в тому, що обліковий запис розробника з можливістю розміщення своїх додатків в магазині додатків AppStore досить дороге задоволення - його необхідно щорічно оплачувати.

Також слід зазначити, що вибір пристроїв, що використовують дану систему, досить невеликий - лише пристрої компанії Apple. Однак, дана система і пристрої, її використовують, дуже популярні і затребувані, а за деякими дослідженнями операційна система iOS є найпопулярнішою на сьогоднішній день.

1.1.3 Операційна система Android

Рисунок 1.3 - Головний екран Android

Операційна система Android є досить молодий, проте дана система досить поширена і є дуже популярною. Популярність даної системи обумовлена тим, що більшість пристроїв на світовому ринку використовують саме її. Також ці пристрої є більш бюджетними, в порівнянні з продукцією Apple на операційній системі iOS.

Крім величезного вибору пристроїв, існують також безліч варіантів самої операційної системи, оскільки її вихідний код знаходиться у відкритому доступі. Система Android надає велике кількість API для розробників, що істотно допомагає при розробці мобільних додатків для пристроїв, що використовують цю операційну систему.

Дана система є серйозним конкурентом iOS і за деякими дослідженням є найпопулярнішою системою, топ мобільних платформ пристроїв, з яких користувачі відвідують веб, виглядає наступним чином:

- Android (55% користувачів);
- iOS (26% користувачів);
- Windows Phone (3% користувачів);
- Інші операційні системи (16% користувачів).

Підводячи підсумок, можна зробити висновок, що розробка програми для операційної системи Android є оптимальним рішенням.

1.2 Актуальність мобільних додатків

Смартфони зробили наше життя набагато простіше і швидше. У міру розвитку технологій, що лежать в основі смартфонів, індустрія мобільних додатків також постійно розвивається. З більш ніж 2,96 млн додатків (і їх кількість зростає) в магазині Google Play, Android з моменту своєї появи набрав обертів і випередив конкурентів з часткою ринку більше 85%. З кожним роком Android продовжує пропонувати нові ідеї, інновації, методи і інструменти в розробці мобільних додатків. Майбутнє розробки додатків для Android дуже багатообіцяючі, що допоможе вам випередити своїх конкурентів.

Впровадження Android росло семимильними кроками, в основному завдяки покращеному інтерфейсу. Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій в останні роки. Кількість розробників мобільних додатків збільшується, кількість доступних додатків зростає. Все більше компаній зацікавлені в розробці програми, яка допоможе їм

досягти успіху в своїй галузі і обійти конкурентів. Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці і в будь-якій ситуації. Це відкриває великі можливості щодо впровадження різних сервісів, які раніше були недоступні.

Важливу роль також відіграє соціалізація додатків, тобто додавання функцій соціальних мереж, які дозволяють користувачем ділитися своїми досягненнями і знаходити нових друзів. Одним з безлічі типів додатків для мобільних пристроїв є додатки для подорожей і туризму. Особливо під час таких подорожей мобільні додатки повинні краще працювати в оффлайн режимі, наприклад у ті моменти коли мобільний пристрій не має сигналу оператора мобільного зв’язку, або доступу до мережі інтернет. На даний момент багато туристів мають мобільні пристрої і використовують їх під час подорожей, і компанії, що не мають таких додатків, втрачають безліч потенційних клієнтів

Сучасна людина робить все для того щоб досягти максимального комфорту. Сьогодні одним з бажань більшості людей є вихід в Інтернет. Причому вони завжди хочуть залишатися онлайн. Саме тому величезною актуальністю користується така послуга, як розробка мобільних додатків під ios. Все це стало актуальним разом з появою мобільного Інтернету. Під час поїздок завжди є можливість підключитися до мережі за допомогою телефону, планшета або іншого пристрою. Але, відразу ж варто відзначити, що без спеціальних додатків навряд чи б була досягнута необхідна ефективність. Без них не обійтися і при вирішенні таких завдань, як архітектурна 3D візуалізація.

Сьогодні фахівцями в області інформаційні технології розробляються мобільні додатки, які дозволяють вирішувати величезна кількість задач, наприклад, створення 3D анімації. Деякі служать для того щоб встановлювати з'єднання з мережею. Інші допомагають оптимізувати маршрут. Треті призначені для тих, хто шукає найвигідніші магазини. Є й такі, за допомогою яких можна замовити їжу додому. В основу кожної з таких програм легкі певні

утиліти, що в результаті дозволяє швидко вирішувати поставлену задачу, економити час і досягати максимально комфортного рівня життя.

1.3 Аналіз ринку планувальників справ

Кожен день люди планують безліч справ. Багато з них є персональними, розраховані на залучення тільки одну людину - самого організатора. Однак, найчастіше необхідно виконувати спільні справи. І для успішного виконання такої справи дуже важливо розподілити обов'язки учасників. Існують різні варіанти сервісів, використання яких дозволяє спростити і оптимізувати планування спільних справ. Такі сервіси можна розділити на три категорії:

- Web-сервіси;
- Десктопні додатки;
- Мобільні додатки.

Слід докладніше розглянути кожну з перерахованих категорій.

Такі сервіси представлені у вигляді сайтів і надають великий функціонал. Співробітники компанії можуть призначати виконання завдань своїм підлеглим, встановлювати терміни виконання і спостерігати за прогресом. Використання подібних сервісів істотно спрощує організацію роботи фірми і допомагає краще контролювати діяльність співробітників. Web-сервіс «Мегаплан» є корпоративною CRM-системою і позиціонує себе як помічника в підвищенні продажів і плануванні бізнесу через інтернет. За допомогою даного сервісу можна контролювати всю щоденну роботу компанії: співробітники записують справи, вирішують завдання, стежать за проектами, знайомляться з колегами, спілкуються з клієнтами, призначають зустрічі, укладають угоди, зберігають документи, будують звіти і знаходяться в курсі останніх подій.

Таким чином, за допомогою описаного сервісу можна ефективно контролювати функціонування цілої компанії, що успішно здійснюють безліч великих компаній, таких, як, наприклад, Avon, 2GIS, Rutube, ESET і інші. Однак, в разі, коли необхідно організувати прості повсякденні спільні справи,

участь в яких братимуть, наприклад, члени сім'ї, недоцільно і незручно використовувати таку потужну систему. До того ж, даний сервіс є безкоштовним тільки протягом пробного 14-денного періоду. Після закінчення цього терміну використання сервісу «Мегаплан» стає платним. Існує також спрощена безкоштовна версія для особистого використання - сервіс «Мініплан». Однак даний web-сервіс не надає можливостей для спільного використання і призначений для індивідуального планування.

1.3.1 Producteev

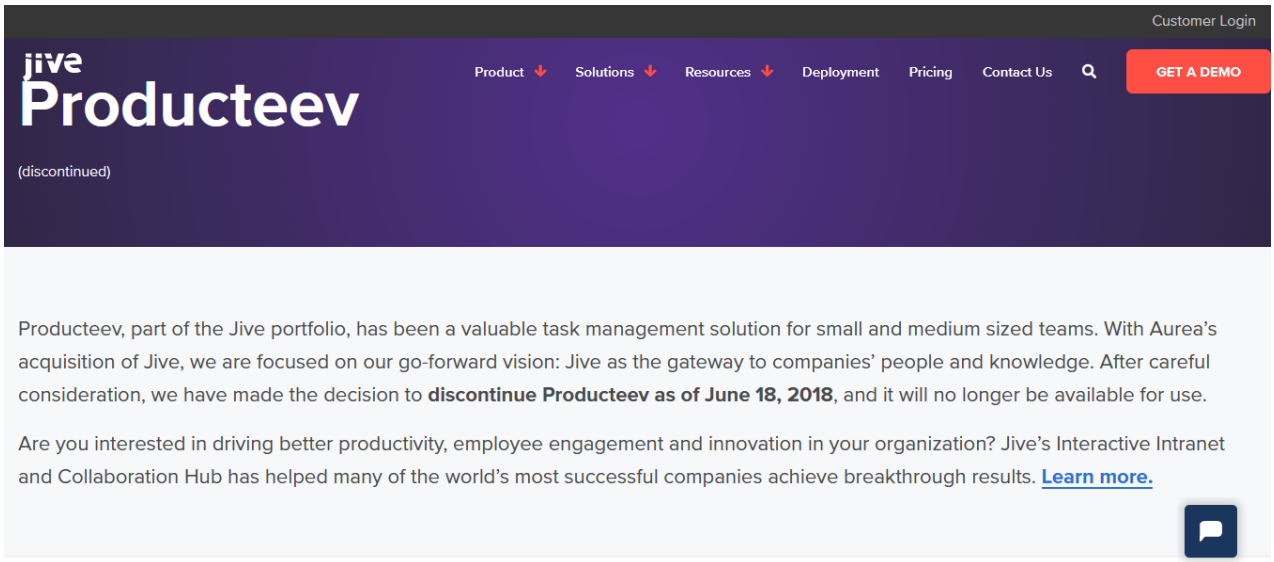

Рисунок 1.4 - Головна сторінка Producteev

Першим з розглянутих продуктів є Producteev. Ця програма має багатий набір функцій. Кожен проект є єдиним робочим простором з унікальними набором користувачів і настройками доступу. Користувачі представлені як список контактів, з якого можна швидко отримати доступ до завдань, відповідальним за якими є обраний користувач, а так само створити для нього нове завдання. Завдання можна створювати і прив'язувати до будь-якому користувачеві зі списку контактів, кожна задача має власну переписку, дату закінчення, налаштування повідомлень про зміну властивостей цього завдання або додаванні нового коментаря в переписку, список відстежують цю задачу користувачів, прикріплені до задачі файли і мітки. Мітки можуть бути

призначені завданням з метою ідентифікації їх приналежності до різних видів робіт.

Однак, організація роботи над проектами, в якій кожен проект є унікальним сховищем завдань і користувачів може бути незручною в разі, коли над декількома проектами працює один набір користувачів. Для створення нового проекту з набором користувачів, вже певним в іншому проекті в Producteev необхідно вручну додавати кожного користувача в новий проект, заново створювати набори міток, які є унікальними для кожного проекту і налаштовувати доступи.

Так само в Producteev завдання можуть володіти тільки двома статусами завершеності: зроблена або не зробили.

Інтерфейс в Producteev є типовим для додатків такого класу. Екран розбитий на кілька колонок, кожна з яких містить певний набір даних. Зліва знаходиться колонка з навігацією і списком користувачів, правіше - колонка зі списком завдань, і крайня права колонка містить інформацію, пов'язану з обраної в даний момент завданням. Вибір проекту або робочого простору знаходиться у верхній частині програми. Так як мається на увазі, що вибір робочого простору - це нечасте дію, то має сенс приховувати цю частину сторінки через її непотрібність.

Навігаційне меню додатка дозволяє фільтрувати список всіх завдань проекту по їх матюками, завершеності, з відповідального користувачеві і по рейтингу. У Producteev є можливість оцінки завдань у вигляді рейтингу, що є зайвою функцією, так як рідко виникає ситуація, в якій завдання необхідно оцінювати.

При роботі з інтерфейсом програми можна відзначити, що сторінка ніколи не перезавантажується, за винятком випадку ручної перезавантаження.

Producteev є платним програмним забезпеченням, але для проектів, в яких бере участь до 2-х осіб є можливість безкоштовного використання.

Беручи до уваги все вищесказане по кожній з розглянутих характеристик,

можна сказати наступне:

- Функціональність. Програма має стандартний набір функцій для додатків даного класу, а так само надає можливість оцінки завдань.
- Інтерфейс. У Producteev інтерфейс представляє з себе кілька колонок, у кожної своє функціональне призначення. Такий вид інтерфейсу типовий для багатьох рішень даного класу.
- Завантаження даних без перезавантаження сторінки. При роботі з інтерфейсом завантаження будь-яких даних відбувається без перезавантаження сторінки.
- Організація даних. Producteev зберігає всі дані тільки в рамках одного проекту, без можливості створення проектів з таким же набором користувачів, міток і т.п.
- Статуси завершеності завдань. У Producteev завдання може бути або завершеною, або незавершеною.
- Платність. Producteev надає можливість безкоштовного користування для проектів з кількістю користувачів до 2-х. Якщо учасників більше, то необхідно оплатити товар.

1.3.2 Asana

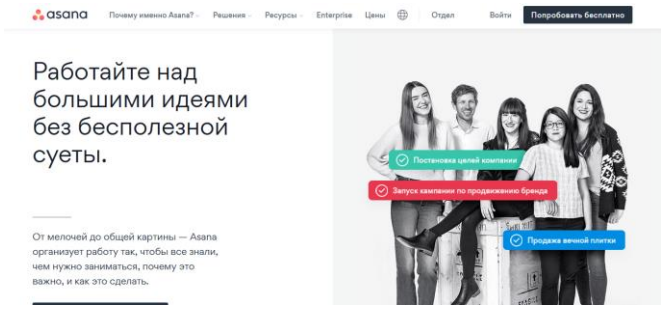

Рисунок 1.5 - Головна сторінка Asana

Другим розглянутим продуктом є Asana. Його функціональність і інтерфейс схожі на Producteev. Основною відмінністю є організація даних. У Asana проекти, завдання, мітки, користувачі і настройки доступу можуть бути об'єднані в одне робоче простір, в якому всі дані видно всім користувачам

даного робочого простору. Це головна перевага даного продукту, яке робить його придатним для використання в середовищі, де одна група людей бере участь у багатьох проектах, на відміну від Producteev.

Крім того, в Asana передбачена функція вхідних повідомлень, яка дозволяє відслідковувати всі зміни в проектах, які відбувалися з моменту останнього використання програми користувачем і в яких він відзначений як цікавиться або відповідальний. Вхідні повідомлення можуть бути особливо зручні для менеджерів проектів, так як вони дозволяють відслідковувати в цілому просування в реалізації проектів.

Як і Producteev, Asana надає тільки 2 статусу завершеності завдань - завершено або не завершено.

Не має сенсу описувати цю програму більш докладно, так як воно аналогічно вже розглянутого Producteev за винятком функцій, описаних вище.

1.3.3 Redmine

Наступне програмне забезпечення, розглянуте в главі, - це Redmine. Redmine є більш всеосяжним рішенням, ніж програма по організації робіт в проекті. У ньому передбачена функціональність корпоративного порталу, що включає в себе новини, зберігання документів, сторінки wiki, сховище загальнодоступних файлів і планувальник завдань.

У планувальнику Redmine представлений список всіх проектів і завдань по ним, доступ на читання до яких має користувач. Так само кожен проект може мати підпроектів з власними налаштуваннями доступу. Одна установка системи надає одне робоче місце.

На сторінці обраного проекту відображається список всіх завдань по ньому, а так само гнучко настроюються фільтри з будь-якого властивості завдання. Список представлений у вигляді таблиці з налаштованим колонками, кожна з яких показує значення певної властивості. Всі перераховані функції завжди видно користувачеві, хоча не у всіх випадках потрібні, тому інтерфейс може здаватися перевантаженим і складним.

У Redmine всі переходи по посиланнях викликають перезавантаження сторінки. Кожне завдання володіє власною унікальною сторінкою зі списком усіх властивостей, обговоренням, списком прикріплених файлів і описом. Так само в Redmine передбачена функція відстеження всіх змін по завданню, що теж відображається на сторінці.

На відміну від розглянутих раніше систем, в Redmine передбачено 7 статусів завдань: нова, очікування відповіді, підтверджена, в роботі, вирішена, закрита, відхилена. Це вичерпний список, але було б зручно мати можливість відключати деякі статуси за непотрібністю.

Як було сказано, кожен перехід по посиланню в Redmine призводить до повної перезавантаження сторінки. Це веде до погіршення продуктивності, так як кожен раз, коли користувач змінює стан системи, з сервера запитується повна сторінка, що містить не тільки запитувану інформацію, але і навігаційне меню, список проектів і т. п. Цей процес споживає значно більше ресурсів на відміну від підходу, реалізованого в Producteev і Asana.

Redmine є додатком з відкритим вихідним кодом, що дає можливість змінювати його під свої потреби. Як наслідок, система є повністю безкоштовною, але для її роботи знадобитися установка на власному або орендованому сервері.

Підводячи підсумки, можна сказати про Redmine наступне по кожній розглянутій характеристиці:

- Функціональність. Крім стандартного набору функцій в Redmine є гнучкі фільтри завдань, а так само можливість створення підпроектів.
- Інтерфейс. У Redmine інтерфейс був розроблений досить давно, що має свої наслідки: Redmine має незручний і несучасний інтерфейс.
- Завантаження даних без перезавантаження сторінки. Повністю відсутня в даній системі.

- Організація даних. Всі дані зберігаються в рамках одного проекту або підпроекту.
- Статуси завершеності завдань. У Redmine передбачено 7 статусів завдань.
- Платність. Redmine повністю безкоштовний для використання.

РОЗДІЛ 2: РОЗРОБКА ПРОДУКТУ ДЛЯ ПЛАНУВАННЯ СПРАВ

В даному розділі проводиться опис роботи по розробці мобільного додатку для планування справ. Для якісної розробки будуть проведені наступні роботи:

- аналіз вихідних даних;
- розробка стратегії створення мобільного застосунку;
- реалізація мобільного застосунку;
- опис технологій та додатків для мобільного застосунку;
- опис функціоналу;
- тестування мобільного застосунку.

2.1 Аналіз вихідних даних

Мобільний застосунок для планування справ – це інформаційна структура з використанням бази даних SQLite, що не використовує мережу Інтернет та є автономним додатком. Основна цільова аудиторія – це люди, що хочуть ефективно планувати свої справи та мати звучні нагадування про вже заплановані справи.

Для створення веб-застосунку було виведено наступну реалізацію веб-сайту:

- головна сторінка, яка містить список усіх справ, а також є можливість пошуку серед списку справ, корегування справ, додавання нових справ;
- календар, де користувач може переглянути на які дні у нього заплановано існуючі справи, переглянути список справ у певні дні;
- смітник, у якому накопичуються видалені справи перед кінцевим видаленням;
- списки справ, що можуть вільно створюватися користувачем та існують для зручного сортування справ користувача.

Основна вхідна інформація мобільного застосунку буде складатися з таких даних:

Дані справи, що створюється:

- Назва
- Опис справи
- Дедлайн
- Час нагадування
- Прогрес
- Пріорітет
- Список, у який буде додано справу

Список справ, що створюється:

- Назва списку

Підзадача, що створюється:

- Назва підзадачі
- Справа, до якої буде додано підзадачу

Розглянувши інформацію вхідних ресурсів, можна зробити висновки, що веб-сайт потребує повинен відповідати ряду таких вимог:

- створення сучасного та зручного веб-дизайну;
- адаптивність для зручного використання з різних мобільних пристроїв;
- для текстової інформації обрати єдиний стиль, шрифт та розмір;
- виконати зручне редагування справ на головній сторінці;
- виконати зручне додавання нових справ з головної сторінки;
- вивести загальну інформацію про мобільний застосунок в окремому activity;
- виконати додавання, видалення та редагування справ на головній сторінці.

2.2 Реалізація розробки веб-застосунку

В даному параграфі описується реалізація розробки мобільного застосунку для планування справ згідно вимогам, зазначеним вище.

2.2.1 Архітектура застосунку

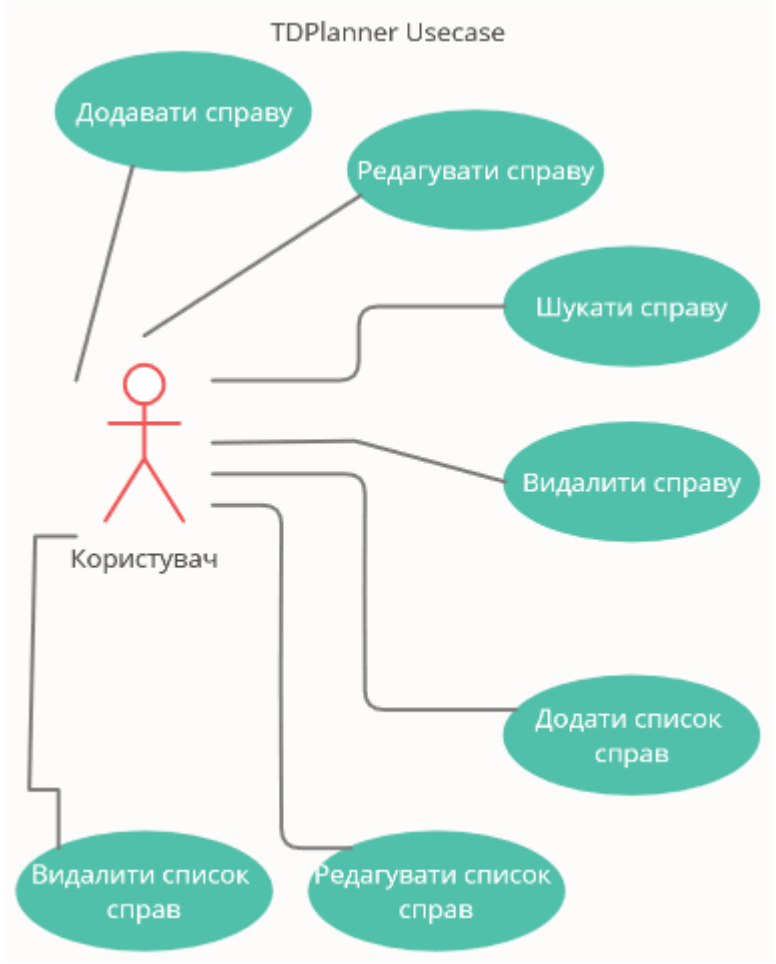

Рисунок 2.1 - Usecase користувача застосунку

Додаток для ОС Android складається з набору активностей, кожній з яких відповідає вікну застосунку. Кожна активність представлена в проекті класом, реалізованим на мові Java, що зберігається в однойменному файлі з розширенням .java. Кожній активності відповідає xml-файл-опис. В xml-файлі описано у вигляді xml-коду розташування візуалізуються об'єктів. При запуску активності система Android автоматично розпізнає розмір екрану мобільного пристрою і призводить виведений контент у відповідність з розміткою, описаною в xml-файлі. Таким чином, одна і та ж активність буде виглядати

однаково незалежно від діагоналі використовуваного пристрою. Також, для кожного додатка Android повинен існувати xml-файл, в якому у вигляді xml-коду будуть прописані мінімальні вимоги до системи, а також активність, яка викликається при запуску програми.

```
// columns + tablename
public static final String TABLE_NAME = "todo_task";
public static final String COLUMN_ID = "_id";
public static final String COLUMN_TODO_LIST_ID = "todo_list_id";
public static final String COLUMN_NAME = "name";
public static final String COLUMN_DESCRIPTION = "description";
public static final String COLUMN_DEADLINE = "deadline";
public static final String COLUMN_DONE = "done";
public static final String COLUMN_PRIORITY = "priority";
public static final String COLUMN_PROGRESS = "progress";
public static final String COLUMN_NUM_SUBTAKS = "num_subtasks";
public static final String COLUMN_DEADLINE_WARNING_TIME = "deadline_warning_time"; // absolut value in seconds
public static final String COLUMN_LIST_POSITION = "position_in_todo_list";
public static final String COLUMN_TRASH = "in_trash";


// sql table creation
public static final String TABLE_CREATE = "CREATE TABLE " + TABLE_NAME + "(" +
        COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        COLUMN_TODO_LIST_ID + " INTEGER NOT NULL, " +
        COLUMN_LIST_POSITION + " INTEGER NOT NULL, " +
        COLUMN_NAME + " TEXT NOT NULL, " +
        COLUMN_DESCRIPTION + " TEXT NOT NULL, " +
        COLUMN_PRIORITY + " INTEGER NOT NULL DEFAULT 0, " +
        COLUMN_DEADLINE + " DATETIME DEFAULT NULL, " +
        COLUMN_DONE + " INTEGER NOT NULL DEFAULT 0, " +
        COLUMN_PROGRESS + " INTEGER NOT NULL DEFAULT 0, " +
        COLUMN_NUM_SUBTAKS + "INTEGER NOT NULL DEFAULT 0, " +
        COLUMN_DEADLINE_WARNING_TIME + " NUMERIC NULL DEFAULT NULL, " +
        COLUMN_TRASH + " INTEGER NOT NULL DEFAULT 0, " +
            "FOREIGN KEY (" + COLUMN_TODO_LIST_ID + ") REFERENCES " + TTodoList.TABLE_NAME + "(" + TTodoList.COLUMN_ID + "));";
```

Рисунок 2.2 - код для створення таблиці справи БД

Застосунок працює зі вбудованої реляційної базою даних SQLite. SQLite не використовує парадігму кліент-сервер, тобто SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а предоставляет 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 виповнюється програма.

2.2.2 Розробка дизайну

Прототипи стали незамінним етапом професійної розробки програмного забезпечення. Прототипи полегшують життя всім людям, залученим до процесу

створення того чи іншого проекту: клієнту, менеджерам, дизайнерам і розробникам. Прототип візуалізує кінцевий продукт і допомагає виявити серйозні проблеми на самих ранніх етапах, допомагаючи уникнути проблем на пізніх стадіях розробки. Звичайно, початкові ескізи, виконані, наприклад, на папері і втілюють перші ідеї дизайнера ще ніхто не відміняв. Але всю подальшу роботу, в яку будуть залучені всі члени робочої групи, найкраще проводити в спеціальному програмному забезпеченні, яке підтримує редагування, управління версіями і спільну роботу.

Прототип додатки буде створюватися в досить популярною графічній програмі Adobe Photoshop [27]. Ця програма є досить зручною у використанні і швидкому освоєнні для побудови прототипу. Починати прототипирование ми почнемо з розробки лівого меню, так як даний функціонал є головним в додатку. Перше, що буде додано в макет це так звані StatusBar і Toolbar (Рис 3).

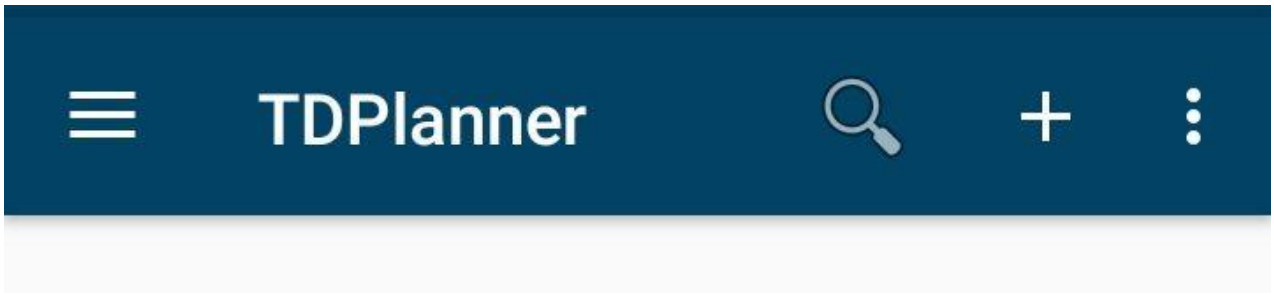

Рисунок 2.3 - StatusBar

Statusbar - це візуальний графічний елемент системи Android для взаємодії користувача з системою.

Toolbar - також відомий як панелі дій, є одним з найбільш важливих елементів дизайну в діяльності нашого застосування, так як він забезпечує візуальну структуру і інтерактивні елементи, які знайомі користувачам. Для створення меню ми звернемося до Google Material Pattern, в якому описуються кращі рішення для побудови дизайну програми.

У рекомендаціях матеріального дизайну Google йдеться, що об'єкти інтерфейсу користувача повинні відкидати тіні, як і об'єкти реального світу. Коли для подання задається властивість elevation, Android автоматично генерує

тінь від цього уявлення. Чим більше значення elevation, тим чіткіше виражена тінь. У рекомендаціях матеріального дизайну наведені бажані значення elevation для різних екранних компонентів - наприклад, для діалогових вікон рекомендоване значення elevation одно 24dp, а для меню - 8dp.

Розробники додатків часто призводять кольору теми у відповідність з фірмовим стилем компанії. Якщо вам буде потрібно змінити кольори теми, рекомендації матеріального дизайну Google по використанню кольору радять вибрати колірну палітру, що складається з основного кольору (не більше ніж з трьома відтінками) і акцентного кольору. Основні кольори зазвичай використовуються для фарбування рядки стану та панелі програми у верхній частині екрану; крім того, вони можуть використовуватися в графічному інтерфейсі.

Згідно з рекомендаціями дизайнерів Google, меню повинно містити елементи вигляді списку з іконками і написами, так само мати заголовок з необхідною функціональністю. Ми вибрали так званий Navigation Drawer для реалізації навігації в додатку.

Значок в документації називається "гамбургером" (Hamburger menu). Це офіційна позиція Google. При натисканні зліва вилізе навігаційна шторка. По висоті вона займає весь екран, включаючи системну область. Можете посувати шторку вперед-назад, щоб побачити, що верхня кромка шторки в системній області напівпрозора і не закриває системні значки. Подібна поведінка є на пристроях під Android 5 і вище. На старих пристроях шторка знаходиться під системної панеллю. Сама шторка складається з двох основних частин - у верхній частині знаходиться картинка і текст, а в нижній - меню зі значками. Меню в свою чергу поділено на дві групи. У верхній частині значки можна вибрати, і обраний пункт залишиться виділеним. У нижній частині меню пункти не виділяються. Приберіть шторку назад і викличте тепер її не натисканням на значок гамбургера, а рухом пальця від краю екран в центр. Ми

побачимо, що під час руху значок трансформується. На жаль, шторка закриває значок і незрозуміло, на що перетворюються три смужки.

Наш Navigation Drawer містить заголовок з логотипом компанії в стилі Material Design [28], нижче йдуть список розділів програми, якими здійснюється навігація по розділах (Рис 4).

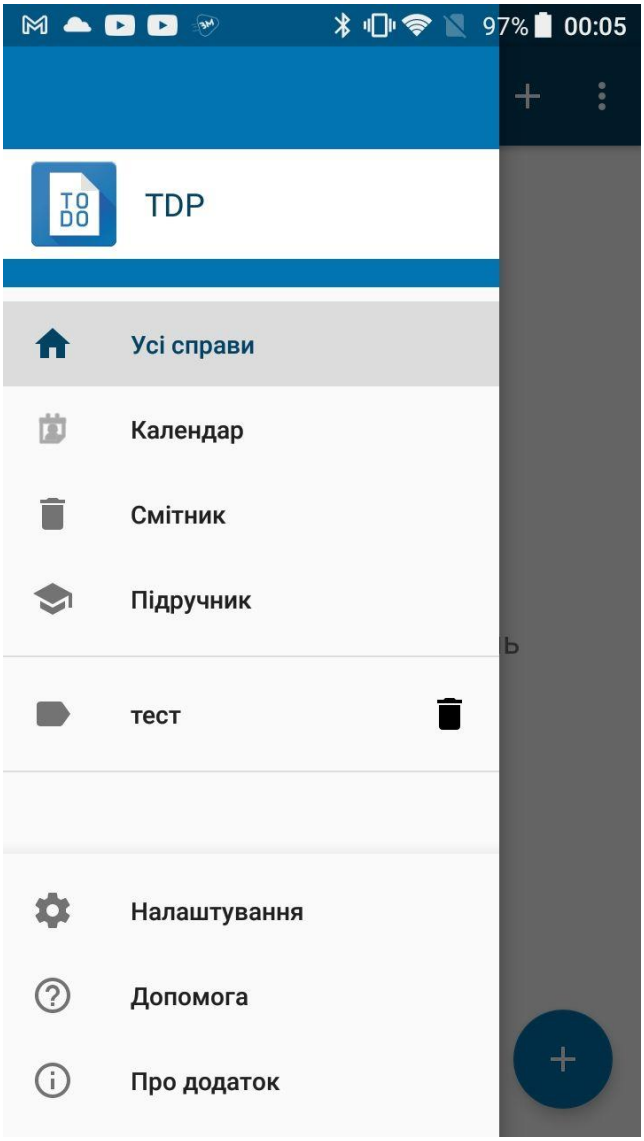

Рисунок 2.4 - NavigationDrawer

Спрототіпірованний Navigation Drawer відповідає вимогам Material Design для Android і надає користувачеві додатки зручні можливості для навігації по розділах, також містить логотип організації, що поддёргівает приналежність додатки до цієї організації, яка займалася розробкою цього додатка для зручності користування користувача.

На дисплеї елементів в списках є дуже поширеною картини в мобільних додатках. Користувач бачить список елементів, і можна прокручувати їх. Якщо він вибирає один з елементів списку, це може оновити ActionBar або тригери детальну екран для вибору.

Android надає ListView класу, який здатний відображати прокручувати список елементів. Ці елементи можуть бути будь-якого типу.

2.2.3 Реалізація дизайну

Створимо графічний інтерфейс користувача для програми. Макетний редактор (Layout Editor) дозволяє створити графічний інтерфейс користувача шляхом перетягування у вікно програми компонентів GUI, таких як Button, TextView, ImageView і ін. За замовчуванням опис макета для шаблону Empty App зберігається в XML-файлі з ім'ям activity_main. xml в папці res / layout. Ми скористаємося макетним редактором і вікном Component Tree для побудови програми. Розмітка XML у файлі activity_main.xml буде відредаговано тільки для того, щоб змінити спосіб розміщення компонентів TextView і ImageView, використовуваних в додатку.

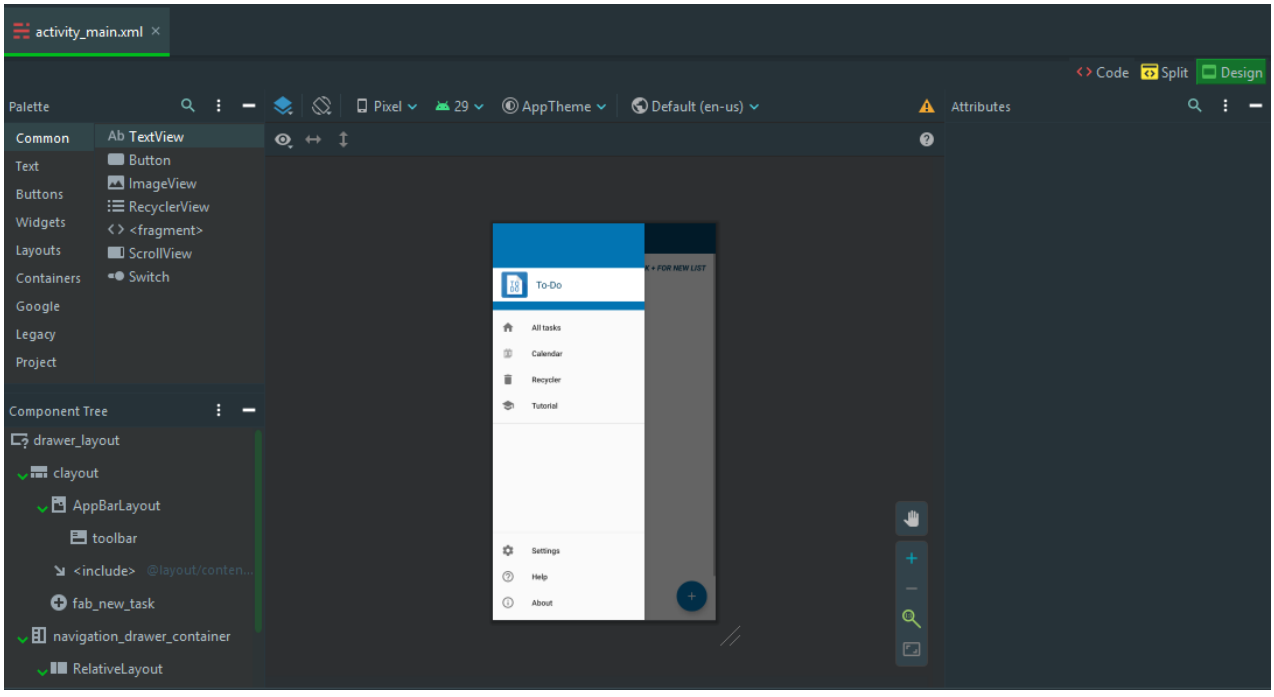

Рисунок 2.5- Layout Editor

Так як екрани пристроїв Android володіють різними розмірами, дозволом і щільністю пікселів (DPI, Dot Per Inch), розробник зазвичай надає зображення з різними дозволами, а операційна система вибирає графіку на підставі щільності пікселів пристрою. З цієї причини папка res вашого проекту містить кілька вкладених папок, імена яких починаються із префікса drawable. Наприклад, зображення для пристроїв з щільністю пікселів, близькою до щільності екрану телефону Google Nexus 6 (560 dpi) для нашого AVD, будуть зберігатися в папці drawable-xxxhdpi.

Завдання розмірів в пікселах, незалежних від щільності (dp або dip), дозволяє платформі Android автоматично масштабувати графічний інтерфейс користувача в залежності від щільності пікселів екрану фізичного пристрою. Розмір пікселя, незалежного від щільності, еквівалентний розміру фізичного пікселя на екрані з роздільною здатністю 160 dpi (точок на дюйм). На екрані з роздільною здатністю 240 dpi розмір пікселя, незалежного від щільності, буде масштабироваться з коефіцієнтом 240/160 (тобто 1,5). Таким чином, компонент, розмір якого становить 100 пікселів, незалежних від щільності, буде масштабований до розміру в 150 фізичних пікселів на такому екрані. На екрані з дозволом 120 точок на дюйм кожен незалежний від щільності пікселів масштабується з коефіцієнтом 120/160 (тобто 0,75). Значить, 100 незалежних від щільності пікселів перетворяться на такому екрані в 75 фізичних пікселів. Пікселі, незалежні від масштабування, масштабируются так само, як і пікселі, незалежні від щільності, але їх масштаб залежить також і від пріоритетного розміру шрифту, обраного користувачем (в установках пристрою).

Ми створили простий графічний інтерфейс в макетному редакторі і налаштували властивості компонентів у вікні властивостей. У XML-файлі розмітки буде використовуватися під компонент RelativeLayout був замінений компонентом LinearLayout, який потім був налаштований для вертикального розташування уявлень. Додаток виводить текст в компоненті TextView, а зображення - в компоненті ImageView. Ми змінили компонент TextView

графічного інтерфейсу за замовчуванням, щоб текст вирівнювався по центру, збільшеним шрифтом і в одному з квітів стандартної теми. Компоненти ImageView перетягували мишею з палітри компонентів макетного редактора. Як і належить, все рядки, і числові значення були визначені в файлах ресурсів в папці res проекту.

Рисунок 2.6 - XML-код для відображення списку задач

У класі MainActivity задіяні багато засобів об'єктно-орієнтованого програмування на мові Java, включаючи класи, об'єкти, інтерфейси, анонімні внутрішні класи і успадкування. Також була представлена концепція заповнення графічного інтерфейсу, тобто перетворення вмісту файлу XML в його екранне уявлення. Ми познайомилися з класом Android Activity і життєвим циклом активності. Зокрема, ми переопределили метод onCreate для ініціалізації програми при запуску. У методі onCreate метод findViewById класу Activity використовувався для отримання посилань на візуальні компоненти, з якими додаток взаємодіє на програмному рівні. Нарешті, ми відредагували файл AndroidManifest.xml, щоб вказати, що MainActivity підтримує тільки

портретну орієнтацію, а клас MainActivity завжди повинен відображати віртуальну клавіатуру.

Заодно були представлені інші елементи, включені Android Studio в маніфест при створенні проекту. Ми показали, як за допомогою об'єкта Configuration визначити, чи виконується додаток на планшеті альбомного формату. Також в цьому розділі було показано, як управляти великою кількістю графічних ресурсів з використанням вкладених папок в папці assets додатки і як звертатися до цих ресурсів через AssetManager. Також були розглянуті інші папки з папки res додатки - menu для зберігання файлів ресурсів меню, xml для зберігання файлів з розміткою XML. Також ми дізналися, як використовувати кваліфікатори при створенні папки для зберігання макета, який повинен використовуватися тільки на великих пристроях альбомного формату. Ми також показали, як використовувати ресурс списку квітів станів, для того щоб гарантувати легкість для читання тексту на кнопках як для доступного, так і для заблокованого стану.

2.3.4 Реалізація програмної частини

Для розробки додатків для ОС Android потрібно встановити Android Studio. Інструменти для розробки Android SDK можна завантажити на сайті для розробників. При установці можна вибрати потрібні для розробки платформи і елементи SDK. При розробці додатків Android використовується Java - один з найбільш поширених мов програмування. Використання Java стало логічним вибором для платформи Android, тому що це потужний, вільний і відкритий мову, відомий мільйонам розробників. Досвідчені програмісти Java можуть швидко освоїти Android-програмування, використовуючи інтерфейси Google Android API (Application Programming Interface) та інші розробки незалежних фірм. Мова Java є об'єктно-орієнтованим, надає розробникам доступ до потужних бібліотекам класів, прискорюють розробку додатків.

Програмування графічного інтерфейсу користувача управляється подіями, які реагують на ініційовані користувачами події, такі як торкання

екрана. Крім безпосереднього написання коду додатків можна скористатися середовищами розробки Eclipse і Android Studio, що дозволяють збирати графічний інтерфейс з готових об'єктів, таких як кнопки і текстовіполя, перетягуючи їх в певні місця екрану, додаючи підписи і змінюючи їх розміри. Ці середовища розробки дозволяють швидко і зручно створювати, тестувати і налагоджувати додатка Android.

Компоненти графічного інтерфейсу в Android називаються уявленнями (views). Вертикальне уявлення LinearLayout використовується для розміщення тексту і графіки так, щоб кожну виставу займало половину вертикального простору LinearLayout. Компонент LinearLayout також дозволяє розміщувати уявлення по горизонталі. Для виведення тексту в додатку буде використовуватися компонент TextView, а графіка буде відображатися в компоненті ImageView. Графічний інтерфейс, створений для зі стандартними програмами, містить компонент TextView. Різні параметри цього компонента - текст, розмір шрифту, колір тексту, розмір щодо компонента ImageView в LinearLayout і т. Д. - налаштовуються у вікні властивостей середовища розробки. Потім ми перетягнемо компонент ImageView з палітри в макет графічного інтерфейсу і налаштуємо його властивості, включаючи джерело графічних даних і позицію в LinearLayout.

Мова XML (eXtensible Markup Language, тобто розширювана мова розмітки) є природним способом опису графічних інтерфейсів. Розмітка XML добре читається як людиною, так і комп'ютером; в контексті Android вона використовується для опису макетів використовуваних компонентів і їх атрибутів: розміру, позиції, кольору, розміру тексту, полів і відступів. Android Studio розбирає розмітку XML, щоб відобразити макет в макетному редакторі і згенерувати код Java, яка формує графічний інтерфейс на стадії виконання. Також файли XML використовуються для зберігання ресурсів програми: рядків, чисел, кольорів і т.д.

У кожної програми існує тема, яка визначає оформлення стандартних компонентів, які ми використовуємо. Тема програми вказується у файлі AndroidManifest.xml додатки. Ми можемо налаштувати різні аспекти теми (наприклад, складові колірної схеми), визначаючи ресурси в файлі styles.xml, що знаходиться в папці res / values додатки.

Ресурсний файл style.xml містить стиль з ім'ям "AppTheme", посилання на який включається в файл AndroidManifest.xml додатки для призначення теми. Цей стиль також визначає батьківську тему, яка може розглядатися як аналог суперкласу в Java - новий стиль наслідує атрибути батьківської теми і їх значення за замовчуванням. Як і субклассов Java, стиль може перевизначити атрибути батьківської теми значеннями, адаптованими для конкретних додатків (наприклад, для використання в додатку фірмової колірної гами компанії).

Ми використовуємо цю концепцію для настройки трьох кольорів, використовуваних в темі програми. Як згадувалося раніше, шаблони додатків Android Studio тепер включають підтримку бібліотек AppCompat, що дозволяють використовувати нові можливості Android в старих версіях платформи. За замовчуванням Android Studio вибирає батьківську тему Theme.AppCompat.Light.DarkActionBar, одну з декількох стандартних тим в бібліотеці AppCompat - додатки, що використовують цю тему, відображаються на світлому тлі, а у верхній частині додатка розташовується темна панель програми. Всі теми AppCompat використовують рекомендації матеріального дизайну Google для оформлення графічних інтерфейсів.

Також для розробки потрібно середовище виконання Java Runtime Environment (JRE), комплект розробника Java Development Kit (JDK), які можна завантажити з офіційного сайту Oracle.

Створення віртуального пристрою Android. Android tools включає в себе емулятор «Android Virtual Device» (AVD). Емулятор AVD дозволяє тестувати програми на віртуальному мобільному пристрої з ОС Android. Емулятор дозволяє створювати кілька віртуальних пристроїв з різними конфігураціями.

Проект - це група пов'язаних файлів (наприклад, файли коду, ресурси і графічні файли), що утворюють додаток. Робота над додатком починається зі створення проекту. Щоб створити додаток в середовищі Android Studio для операційної системи Android. Відкриємо Android Studio. Для створення нового проекту треба перейти до пункту меню File -> New-> New Project .... Після цього у нас з'явиться діалогове вікно створення нового проекту:

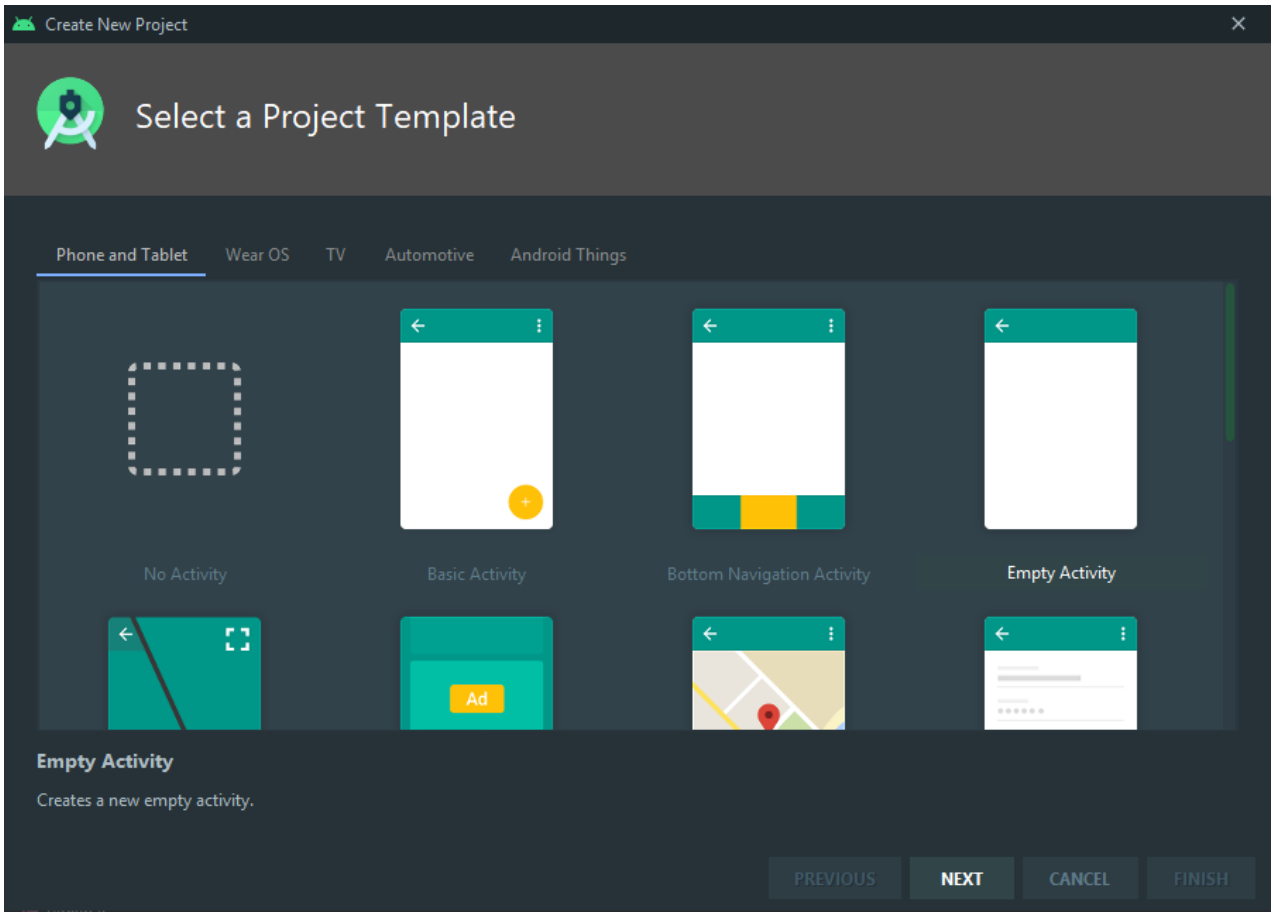

Рисунок 2.7 - Вікно створення нового проекту

На кроці Configure your new project майстра Create New Project введемо наступну інформацію.

1. Application Name: - назва програми.

2. Company Domain: - доменне ім'я веб-сайту компанії. створення можна використовувати ім'я example.com.

3. Package Name: - ім'я пакета Java для вихідного коду програми. Android і магазин Google Play використовують це ім'я в якості унікального

ідентифікатора додатка, яке повинно залишатися постійним у всіх версіях програми, які ми будемо відправляти в магазин Google Store. Ім'я пакета зазвичай починається з доменного імені вашої компанії або установи, записаного в зворотному порядку. За загальноприйнятим угодами в імені пакету використовуються тільки букви нижнього регістра без пробілів. За замовчуванням IDE вибирає ім'я пакета на підставі тексту, що вводиться в полях Application Name і Company Domain. Щоб змінити ім'я Package, клацніть на посиланні Edit праворуч від згенерованого імені пакета.

4. Project Location: - шлях до папки на вашому комп'ютері, в якій буде зберігатися проект. За замовчуванням Android Studio розміщує папки нових проектів у вкладеній папці AndroidStudioProjects каталогу облікового запису користувача. Ім'я папки проекту складається з імені проекту, з якого віддаляються прогалини. Ми можемо змінити шлях до папки проекту; для цього введіть шлях або клацніть на кнопці (...) праворуч від поля і виберіть папку для зберігання проекту. Після того як папка буде обрана, клацніть на кнопці OK, а потім перейдіть до наступного кроку кнопкою Next.

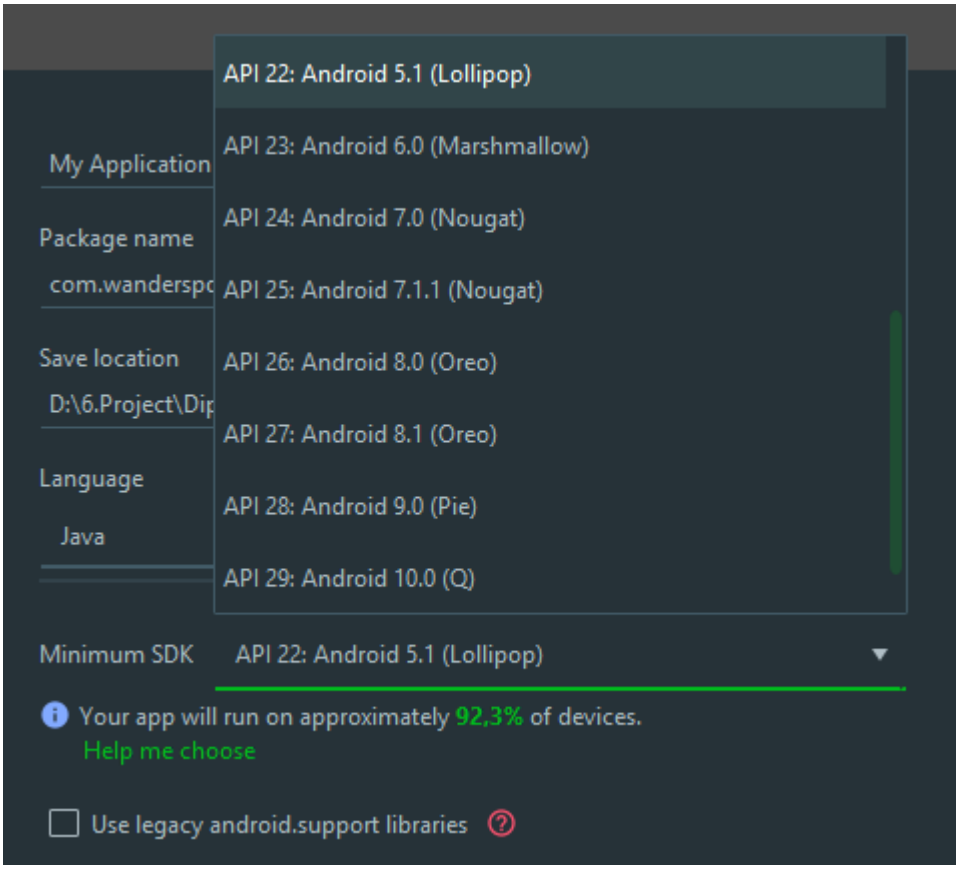

Рисунок 2.8 - Вибір версії Android

На цьому кроці буде запропоновано встановити мінімальну підтримувану версію проекту. За замовчуванням встановлюється версія Android 4.0.3, що покриває майже 97% пристроїв Android. Залишимо за замовчуванням і натиснемо на кнопку Next.

У нашому додатку ми не будемо додавати Activity звичайним чином, тому що відбувається не ефективна генерація програмного коду.

Розглянемо структуру проекту програми під ОС Android, яка створюється за замовчуванням.

Проект може включати різні модулі. І все модулі описуються файлом setting.gradle. І якщо ми подивимося на структуру проекту, то весь значимий код - файли інтерфейсу, класи java і т.д. у нас за замовчуванням знаходяться в папці (модулі) app. Файл build.gradle містить інформацію, яка використовується при побудові проекту.

Кожен модуль має свій файл build.gradle, який визначає конфгурацію побудови проекту, специфічну для даного модуля. Так, якщо ми подивимося на вміст папки app, то, як раз знайдемо в ній такий файл. На початковому етапі дані файли не настільки важливі, досить лише розуміти, для чого вони потрібні.

За замовчуванням кожен проект включає один модуль - app. Власне весь код, з яким ми будемо працювати, розташовується всередині цього модуля.

У цьому модулі ми можемо побачити кілька папок і файлів, з яких для нас найважливішими є:

- каталог libs - призначений для зберігання бібліотек, використовуваних додатком
- каталог src - призначений для зберігання вихідного коду. Він містить ряд підкаталогів. Тексти програм розташовуються в папці main.

Папка main має складну структуру:

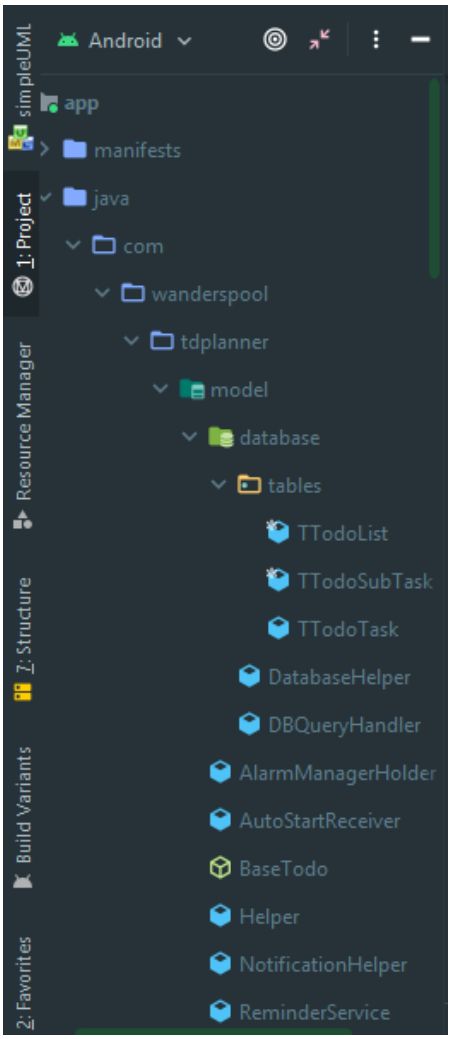

Рисунок 2.9 - Структура Android проекту

Після розгляду загальних питань побудови проекту ми вже можемо приступити до самої реалізації програми. Activity є основою для побудови візуального інтерфейсу Android додатків, і ми в нашому випадку розробимо каркас для основного елемента планувальника завдань - самого завдання: для цього ми створимо клас BaseTodo який буде оголошений абстрактним, що означає що створити на пряму без успадкування буде неможливо:

```
public abstract class BaseTodo {

    protected int id;

    protected int progress;

    public int getProgress(){
        return progress;
    }
```

```
    protected String name, description;
    protected DBQueryHandler.ObjectStates dbState;

    public BaseTodo() {
        dbState = DBQueryHandler.ObjectStates.NO_DB_ACTION;
    }

    public DBQueryHandler.ObjectStates getDBState() {
        return dbState;
    }

    public void setCreated() {
        this.dbState = DBQueryHandler.ObjectStates.INSERT_TO_DB;
    }

    public void setChanged() {
        if(this.dbState == DBQueryHandler.ObjectStates.NO_DB_ACTION)
            this.dbState = DBQueryHandler.ObjectStates.UPDATE_DB;
    }

    public void setChangedFromPomodoro() {
            this.dbState = DBQueryHandler.ObjectStates.UPDATE_FROM_POMODORO;
    }



    public void setUnchanged() {
        this.dbState = DBQueryHandler.ObjectStates.NO_DB_ACTION;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setProgress(int progress) {
        this.progress = progress;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }
    public void setDescription(String description) {
        this.description = description;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

}
```

На відміну від багатьох програм Java, додатки Android не містять методу main. Замість цього в них використовуються чотири типи виконуваних компонентів - активності (activities), служби (services), провайдери контенту і широкомовні приймачі (broadcast receivers). Додаток може мати багато активностей, одна з яких - перше, що ми бачимо під час запуску програми. Користувачі взаємодіють з активностями через уявлення (views) - компоненти GUI, успадковують откласса View (пакет android.view).

До виходу Android 3.0 з кожним екраном програми зазвичай пов'язувалася окрема активність. Активність може керувати кількома фрагментами (fragments). На телефоні кожен фрагмент зазвичай займає цілий екран, а активність переключається між фрагментами на підставі взаємодій користувача. На планшетах активності часто відображають кілька фрагментів на екран, щоб більш ефективно використовувати наявний космос.

Протягом свого існування активність може перебувати в одному з декількох станів - активному (тобто виконується), загальмованому або зупиненому. Переходи активностей між цими станами відбуваються у відповідь на різні події. «Активна активність» відображається на екрані і «володіє фокусом» - тобто взаємодіє з користувачем. Призупинена активність видно на екрані, але не володіє фокусом (наприклад, на час відображення діалогового вікна з повідомленням). Користувач не може взаємодіяти з призупиненої активністю, поки вона знову не стане активною - наприклад, після того, як користувач закриє діалогове вікно. Зупинена активність не відображається на екрані і, ймовірно, буде знищена системою, коли буде потрібно звільнити займану нею пам'ять. Активність зупиняється, коли інша активність переходить в активний стан. Наприклад, коли ми відповідаємо на телефонний дзвінок, додаток, що управляє дзвінками, стає активним, а попередній додаток зупиняється. При переходах активності між цими станами виконавче середовище Android викликає різні методи життєвого циклу (всі ці методи визначаються в класі Activity з пакета android.app). У додатках для кожної активності буде

перевизначатися метод onCreate. Цей метод викликається виконавчою системою Android при запуску активності - тобто коли її графічний інтерфейс готовий до відображення, щоб користувач міг взаємодіяти з активністю. Також у активностей існують інші методи життєвого циклу: onStart, onPause, onRestart, onResume, onStop і onDestroy.

Кожен переобумовленої вами метод життєвого циклу активності повинен викликати версію методу з суперкласу; в іншому випадку відбувається виключення. Виклик версії суперкласу необхідний, тому що кожен метод життєвого циклу в суперкласі Activity містить код, який повинен виконуватися крім коду, що визначається вами в перевизначених методах життєвого циклу.

При використанні нових можливостей на більш ранніх платформах Android розробник стикається з проблемою забезпечення сумісності. Тепер Google відкриває доступ до багатьох нових можливостей Android через Android Support Library - набір бібліотек, завдяки яким розробник може використовувати ці можливості на сучасних і старих платформах Android. Одна з таких бібліотек - AppCompat - забезпечує підтримку панелі програми (яка раніше називалася панеллю дій) і інших можливостей на пристроях Android 2.1 (API 7) і вище; нагадаємо, що панель додатки вперше з'явилася в Android 3.0 (API 11). Оновлені шаблони додатків Android Studio теж використовують бібліотеку AppCompat, тому нові програми, які ми створимо, будуть працювати майже на всіх пристроях Android. Шаблон Android Studio Empty Activity визначає клас MainActivity додатки як субклассов AppCompatActivity (пакет package android.support.v7.app) -непряма субкласса Activity, що забезпечує використання нових засобів Android на сучасних і старих платформах Android.

Android також підтримує неявні (implicit) інтенти - розробник не вказує компонент для обробки інтентів. Наприклад, можна створити інтент для відображення вмісту URL-адреси і доручити Android запустити найбільш підходящу активність (браузер) в залежності від типу даних. Якщо дія і інформація, передана startActivity, можуть бути оброблені декількома

активностями, система виводить діалогове вікно, в якому користувач вибирає активність. Якщо система не може знайти активність для обробки дії, метод startActivity ініціює виключення ActivityNotFoundException. У загальному випадку рекомендується передбачити обробку цього винятку в програмах. Також можна запобігти виникненню самого виключення - використовуйте метод resolveActivity класу Intent для визначення того, чи існує активність для обробки інтенту.

Для будь-якої Activity викликається метод onCreate, він відповідає за створення вікна. Для того щоб можна було відобразити вигляд вікна викликається метод setContentView в який передається числове значення ідентифікатора ресурсу відображення, одержуваного від класу нащадка. Тут так само встановлюється ActionBar, який є у всіх Activity додатки.

У методі onCreate ми створюємо обробник на час завантаження для відображення вікна програми. Після того як пройде час, буде виконаний фрагмент коду, що відкриває наступний Activity. Відбудеться запуск головного вікна і закриття поточного.

Тепер настав час розглянути функціональність головного подання додатка, яким є клас MainActivity. Код MainActivity через довжини знаходиться в додатках.

```
import com.wanderspool.tdplanner.R;

import java.util.ArrayList;

/*
 * This Activity handles the navigation and operation on lists and tasks.
 */

public class MainActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {


    public static final String COMMAND = "command";
    public static final int COMMAND_UPDATE = 3;
    public static final int COMMAND_RUN_TODO = 2;


    private static final String TAG = MainActivity.class.getSimpleName();

    // Keys
    private static final String KEY_TODO_LISTS = "restore_todo_list_key_with_savedinstancestate";
    private static final String KEY_CLICKED_LIST = "restore_clicked_list_with_savedinstancestate";
    private static final String KEY_DUMMY_LIST = "restore_dummy_list_with_savedinstancestate";
    private static final String KEY_IS_UNLOCKED = "restore_is_unlocked_key_with_savedinstancestate";
    private static final String KEY_UNLOCK_UNTIL = "restore_unlock_until_key_with_savedinstancestate";
    public static final String KEY_SELECTED_FRAGMENT_BY_NOTIFICATION = "fragment_choice";
    private static final String KEY_FRAGMENT_CONFIG_CHANGE_SAVE = "current_fragment";
    private static final String KEY_ACTIVE_LIST = "KEY_ACTIVE_LIST";
    private static final String POMODORO_ACTION = "org.wanderspool.privacyfriendlytodolist.TODO_ACTION";



    // Fragment administration
    private Fragment currentFragment;
    private FragmentManager fragmentManager = getSupportFragmentManager();
```

Рисунок 2.10 - Клас MainActivity, відкритий у Sublime Text для візуалізації обьєму класу

Даний клас є відправною точкою докладання, звідси можна відкрити всі подання додатка. Спочатку створюється ліве меню програми, далі встановлюється кнопка для відкриття меню.

Користувач може відкрити меню двома способами:

- Натиснути на кнопку меню
- Свайпом вправо

Відкриється меню і можна буде вибрати пункт меню, після чого в тому ж вікні відкриється інший розділ.

Прийшов час приступити до створення розділів програми, які базуються на новому для Android компоненті під назвою Fragment. Фрагмент (клас Fragment) представляє поведінку або частина призначеного для користувача інтерфейсу в Activity. Розробник може об'єднати кілька фрагментів в одну Activity для побудови багатопанельних призначеного для користувача інтерфейсу і повторного використання фрагмента в декількох Activity. Фрагмент можна розглядати як модульну частина Activity. Така частина має свій життєвий цикл і самостійно обробляє події введення. Крім того, її можна додати або видалити безпосередньо під час виконання Activity. Це щось на зразок вкладеної Activity, яку можна багаторазово використовувати в різних Activity.

Фрагмент (fragment) зазвичай представляє повторно використовувану частину призначеного для користувача інтерфейсу активності, але він також може представляти повторно використовуваний блок програмної логіки. Додаток використовує фрагменти для створення частин графічного інтерфейсу і управління ними. Фрагменти можна об'єднувати для створення інтерфейсів, ефективно використовують розмір екрану планшета. Крім того, простий механізм заміни фрагментів зробить інтерфейс програми більш динамічним. Базовим класом всіх фрагментів є клас Fragment (пакет android.app). При використанні субклассов AppCompatActivity з фрагментами необхідно використовувати версію цього класу з бібліотеки Android Support Library (пакет android.support.v4.app).

Кожен фрагмент, як і активність, має свій життєвий цикл і надає методи, які можна перевизначати для обробки подій життєвого циклу. У цьому додатку будуть перевизначено наступний метод:

onCreateView - цей метод викликається після onCreate; він повинен побудувати і повернути об'єкт View з графічним інтерфейсом фрагмента. Як ми незабаром побачимо, він отримує об'єкт LayoutInflater, який використовується для програмного заповнення графічного інтерфейсу фрагмента по компонентам, заданим в заранеео пределеніе макеті в форматі XML.

Життєвий цикл фрагмента зв'язується з життєвим циклом його батьківської активності. Існують шість методів життєвого циклу активності, у яких є відповідні методи життєвого циклу фрагмента - onCreate, onStart, onResume, onPause, onStop і onDestroy. Коли система викликає ці методи для активності, це призводить до виклику відповідних методів всіх приєднаних фрагментів активності (а можливо, і інших методів життєвого циклу фрагментів). В даному додатку використовуються методи життєвого циклу фрагмента onResume і onPause. Метод onResume викликається в той момент, коли фрагмент знаходиться на екрані і готовий до взаємодії з користувачем. Коли активність, керуюча фрагментами, відновлює роботу, викликаються методи onResume всіх її фрагментів. Коли активність, керуюча фрагментами, призупиняється, викликаються методи onPause всіх її фрагментів.

Фрагменти можуть додавати свої команди в меню керуючої активності. Як клас Activity, клас Fragment містить метод життєвого циклу onCreateOptionsMenu і метод обробки події onOptionsItemSelected.

Фрагмент завжди повинен бути вбудований в Activity, і на його життєвий цикл безпосередньо впливає життєвий цикл Activity. Наприклад, коли операція припинена, в тому ж стані знаходяться і всі фрагменти всередині неї, а коли Activity знищується, знищуються і всі фрагменти. Однак поки Activity виконується (це відповідає стану відновлення життєвого циклу), можна маніпулювати кожним фрагментом незалежно, наприклад, додавати або

видаляти їх. Коли розробник виконує такі транзакції з фрагментами, він може також додати їх у стек переходів тому, яким керує операція. Кожен елемент стека переходів назад в операції є записом виконаної транзакції з фрагментом. Стек переходів назад дозволяє користувачеві звернути транзакцію з фрагментом (виконати навігацію в зворотному напрямку), натискаючи кнопку "Назад".

Батьківська активність використовує для управління своїми фрагментами об'єкт FragmentManager (пакет android.app), що повертається методом getFragmentManager класу Activity. Якщо активності потрібно взаємодіяти з фрагментом, який оголошений в макеті активності і володіє ідентифікатором id, то для отримання посилання на заданий фрагмент активність може викликати метод findFragmentById класу FragmentManager. FragmentManager може використовувати об'єкти FragmentTransaction для динамічного додавання, видалення і перемикання між фрагментами.

Ми дізналися, що таке фрагменти і тепер можна приступити до реалізації базового класу для інших фрагментів. Всі фрагменти потрапляють в категорію уявлень і тому реалізують інтерфейс BaseView або реалізують нащадків даного інтерфейсу. Код TodoTaskFragment який буде базовим для завдання, представлений в додатках через довжину.

Фрагменти залежні від Activity тому завжди мають до них доступ. У методі show переданий контекст явно перетворюється до BaseTodo щоб викликати менеджер фрагментів для початку транзакції відображення фрагмента в Activity. Решта методи, які є в класі, схожі на методи, які ми реалізовували в BaseActivityView, і особливого інтересу не представляють.

Висновки

Виконавши аналіз вхідної інформації стало відомо основні необхідні функції застосунку. Зважаючи на це було створено дизайн, який має сучасний та зручний інтерфейс.

Також виконано роботи з підтримки адаптивності застосунку, що дозволить користуватися сайтом з різних на будь-якому пристрою, що має Android OS версії 4.1 або вище.

Для покращення функціоналу веб-застосунку було проведено роботи з встановленням додатків які відповідають за рівень безпеки, SEO-просування, зв’язок через контактну форму, регулярне створення резервної копії веб-застосунку та інше.

Для перевірки виконаних робіт було проведено тестування застосунку з підключенням користувачів, що дозволило виправити отримані помилки та запобігти їх виникнення в майбутньому.

Розробка веб-застосунку з власним веб-дизайном та функціоналом потребує якісної підготовки та чіткого плану по виконанню всіх робіт, які стосуються створення дизайну та визначення чіткого технічного завдання.

РОЗДІЛ 3: АНАЛІЗ ФУНКЦІОНАЛУ ТА АДМІНІСТРУВАННЯ ПРОДУКТУ

В розділі 3 ми проведемо аналіз основного функціоналу мобільного застосунку для планування задач.

3.1 Основний функціонал мобільного застосунку

Створений додаток можна роздивлятися як прикладну систему, що була побудована як елемент систем динамічних веб-ресурсів. Подібно звичайним веб-ресурсам, наш проект реалізує наступні основні функції: представлення інформації по тренуванням, фізичним вправам та зв’язок с тренерами.

Головною відмінністю проекту від звичайних веб-ресурсів є його розташування та організація взаємодії з веб-адміністратором та користувачами.

Функції веб-застосунку:

- швидке планування справ;
- представлення інтерфейсу до БД справ (у вигляді загального списку справ);
- Структурування справ за окремими списками;
- представлення інтерфейсу до БД програм тренування (у вигляді окремого списку зі справами);
- видалення та відновлення видалених справ за допомогою смітника;
- створення підзадач, кожна з яких буде у складі вже існуючої справи.

Головною відмінністю проекту від звичайних веб-ресурсів є його розташування та організація взаємодії з веб-адміністратором та користувачами.

3.1.1 Додавання нових справ

Основним елементом застосунку, з яким працює користувач, є окрема справа, що створюється після заповнення необхідних даних про цю справу:

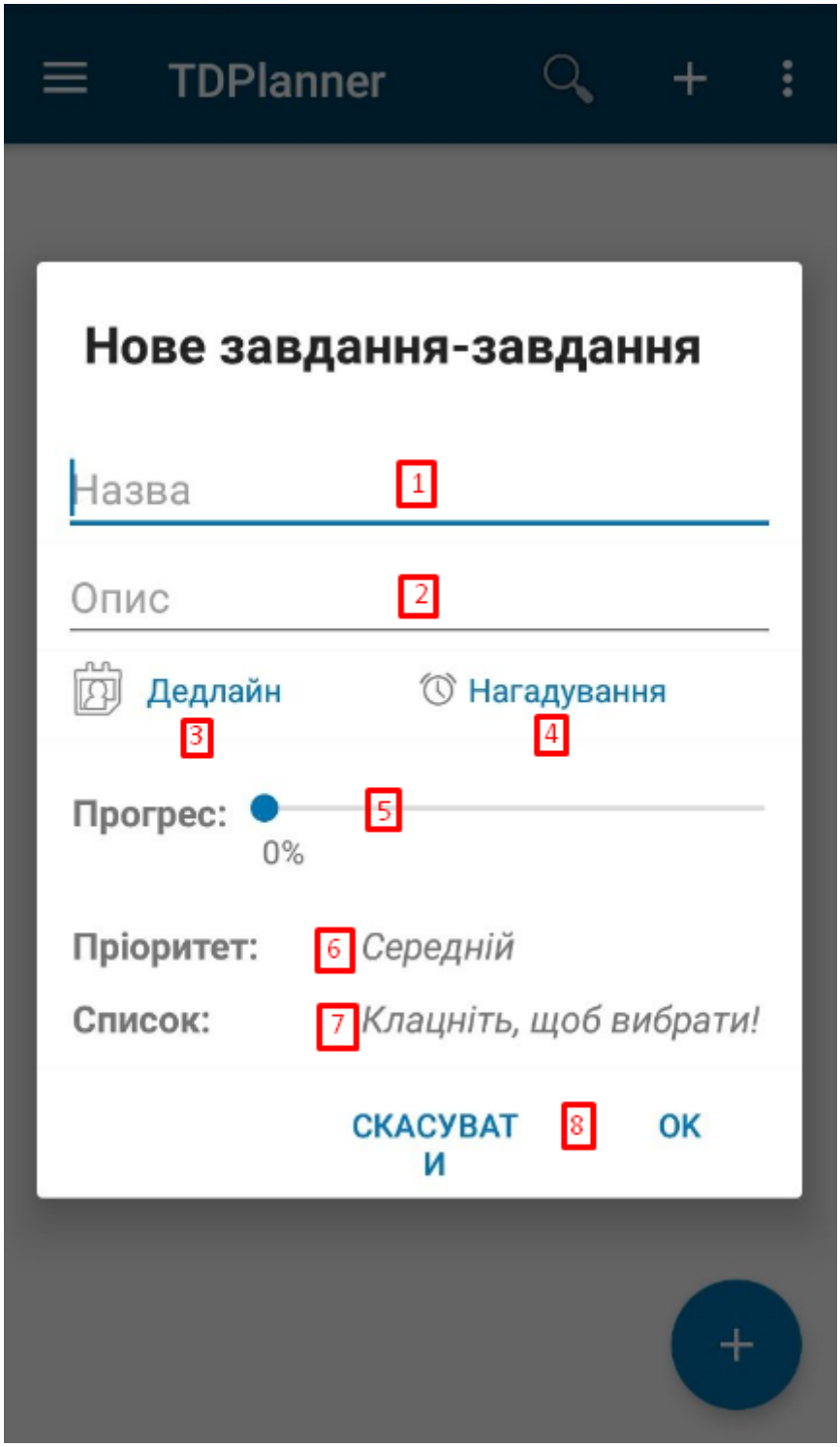

Рисунок 3.1 - Скріншот фрагменту створення задачі

1. назва справи – вмістить загальну назву справи, завдяки якому користувач має змогу швидко її знайти або власноруч, або за допомогою пошуку;

2. опис справи – більш докладний опис справи, що заплановано;

3. дедлайн – визначає момент виконання справи за планом користувача;

4. нагадування – визначає момент нагадування про виконання справи за планом користувача, також може використовуватися для визначення початку справи;

5. прогрес – показує прогрес виконання справи та може буди змінено відповідно виконанню даної задачі;

6. пріорітет – визначає порядок, у якому будуть сортовані справи у загальному та окремих списках, а також надає можливість сортувати усі справи за пріорітетами;

7. Список – надає можливість вибрати список справ, до якого буде належати дана справа

8. Кнопки скасування та підтвердження – кнопки для скасування створення/редагування справи та для підтвердження створення/редагування справи;

Після заповнення відповідних полів та перевірки їх коректності потрібно натиснути на кнопку «Ок», щоб додати нову справу/редагувати стару або «Скасувати», щоб скасувати дану операцію.

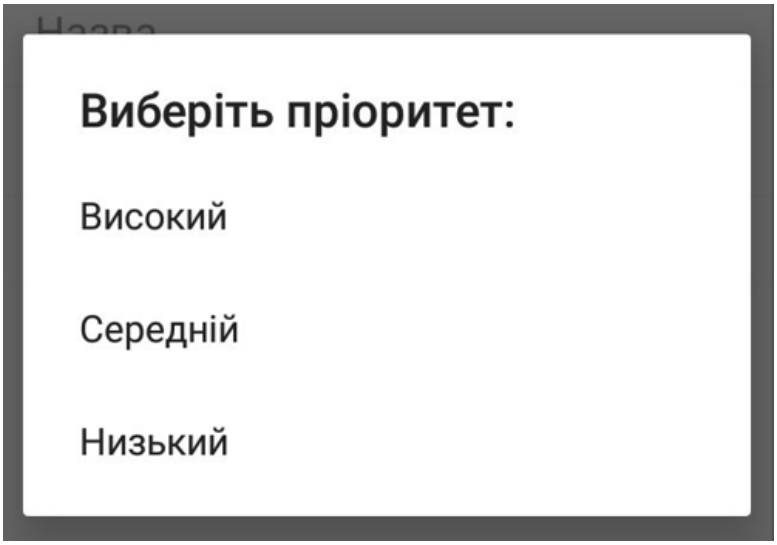

Рисунок 3.2 – Фрагмент вибору пріорітету справи

Оглянути додану справу можливо на головній сторінці застосунку або на сторінці списку справ, до якого її було додано.

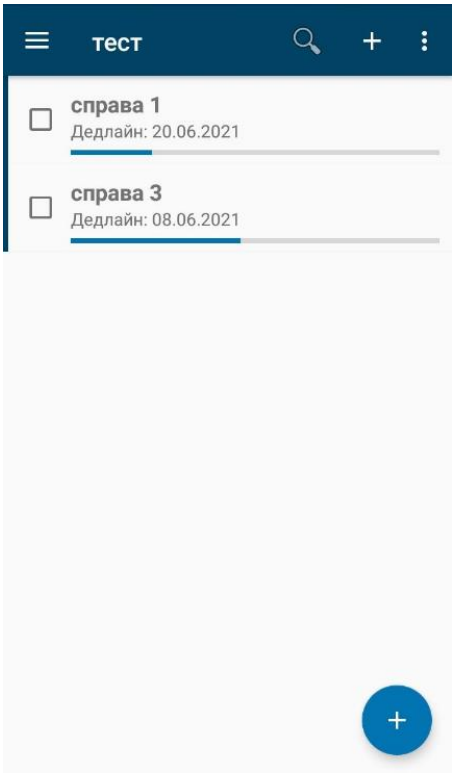

Рисунок 3.3 - Відображення списку справ із вже створеними справами

3.1.2 Додавання нових підзадач до справ

Додавання підзадач виконується за вже сворених справ, до яких ця підзадача додається.

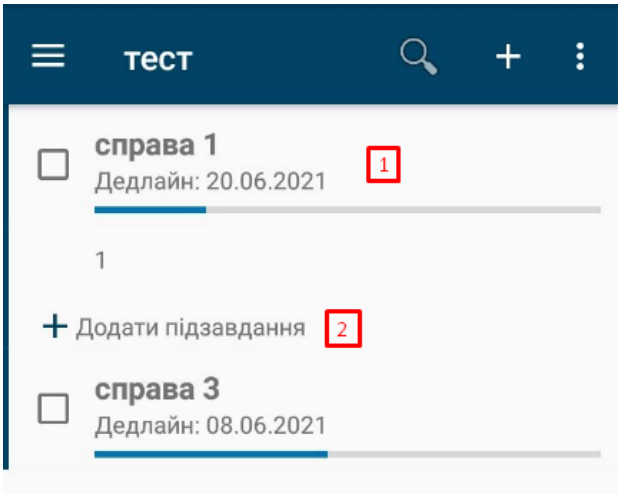

Рисунок 3.4 - Додавання підзадачі. Список справ

Сторінка додавання нової підзадачі містить наступні поля (рисунок 3.5):

1. Справа, до якої буде додано підзадачу;
2. Кнопка для додавання підзадачі;

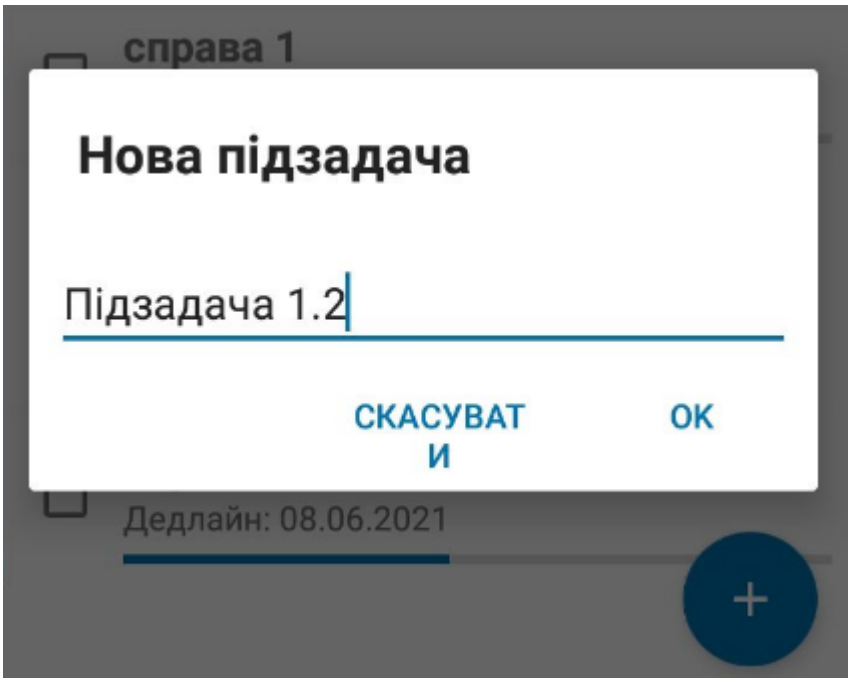

Рисунок 3.5 - Додавання підзадачі. Фрагмент створення

На даному скріншоті відображене вікно для введення назви нової підзадачі та її створення.

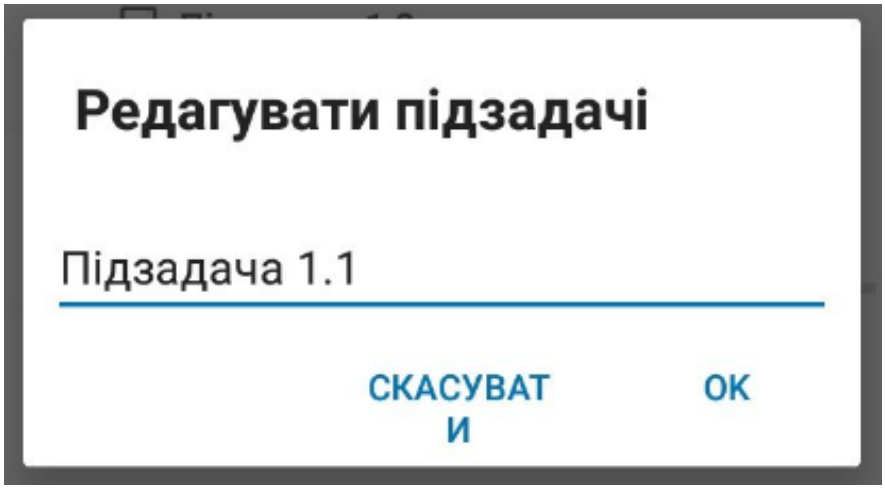

Рисунок 3.6 - Додавання підзадачі. Редагування

3.1.3 Додавання нових списків справ

Для зручного сортування існуючих справ було додано список справ, що містять справи, які було відмічено як єлемент даного списку.

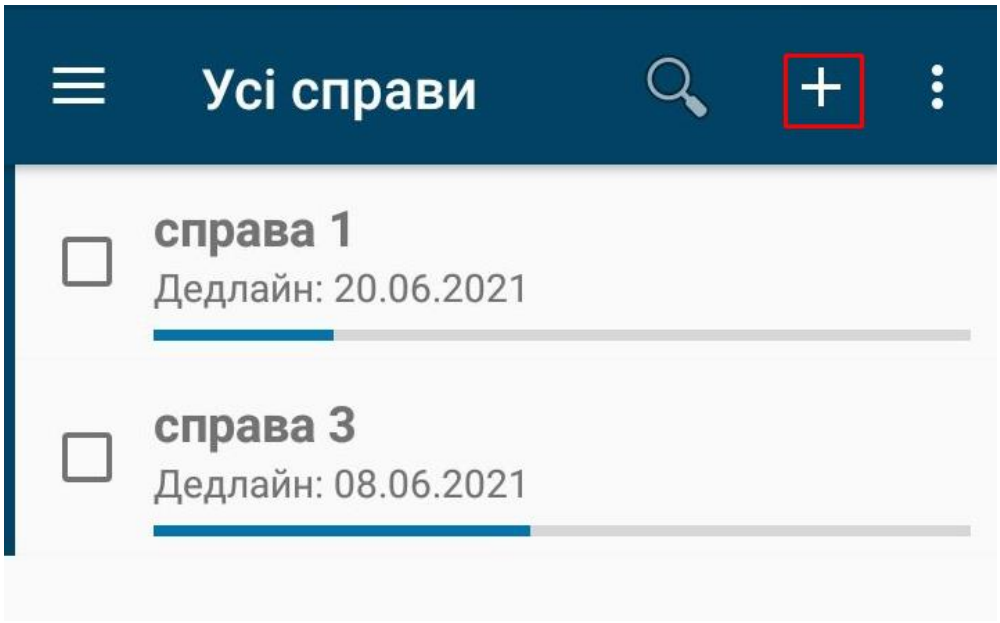

Рисунок 3.7 - Кнопка для додавання новог списку справ

Натиснувши відповідну кнопку, ми побачимо фрагмент створення нового списку, у якому можемо ввести необхідну користувачу назву списку.

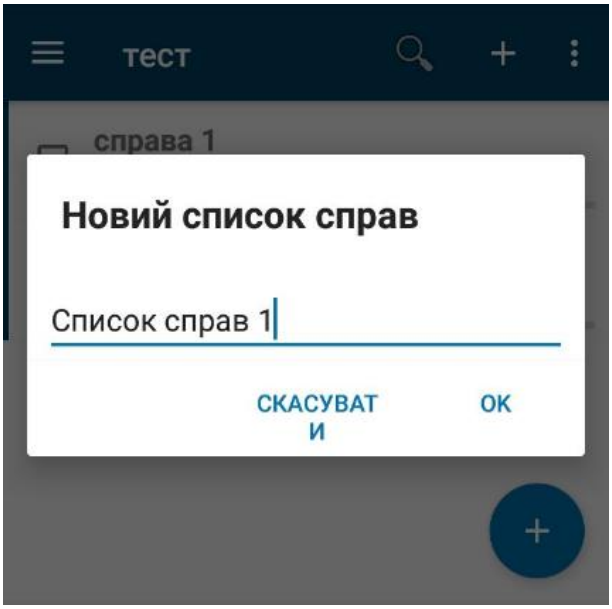

Рисунок 3.8 - Фрагмент створення нового списку

Після вводу назви та натискання на кнопку «Ок» ми зможемо перейти до цього списку за допомогою меню застосунку, що було реалізовано за допомогою NavigationDrawer.

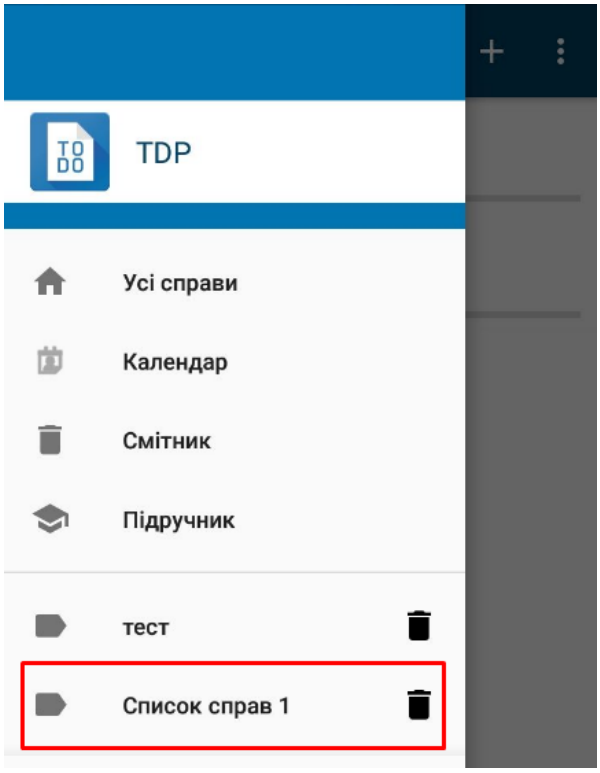

Рисунок 3.9 - Новий список справ

Перейшовши до цього списку, ми зможемо побачити список справ у ньому а також додати нові.

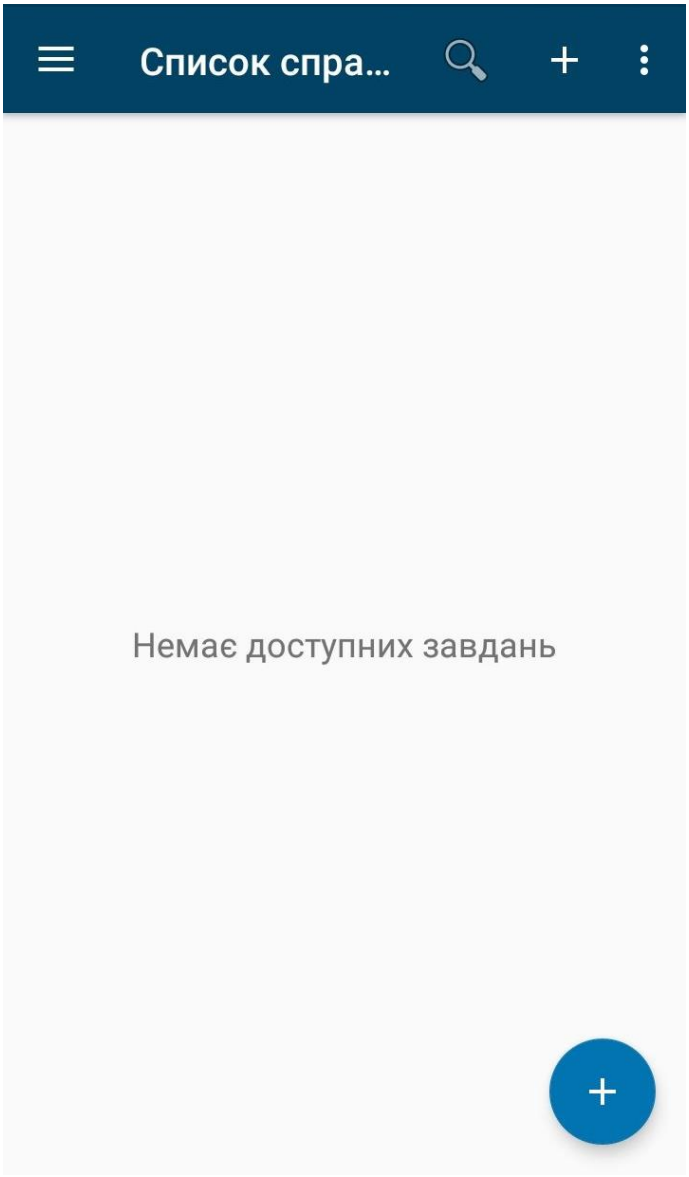

Рисунок 3.10 - Щойно створений список справ

При додаванні нової справи у пункті «Списки» буде автоматично вибрано список, у якому ця справа створюється.

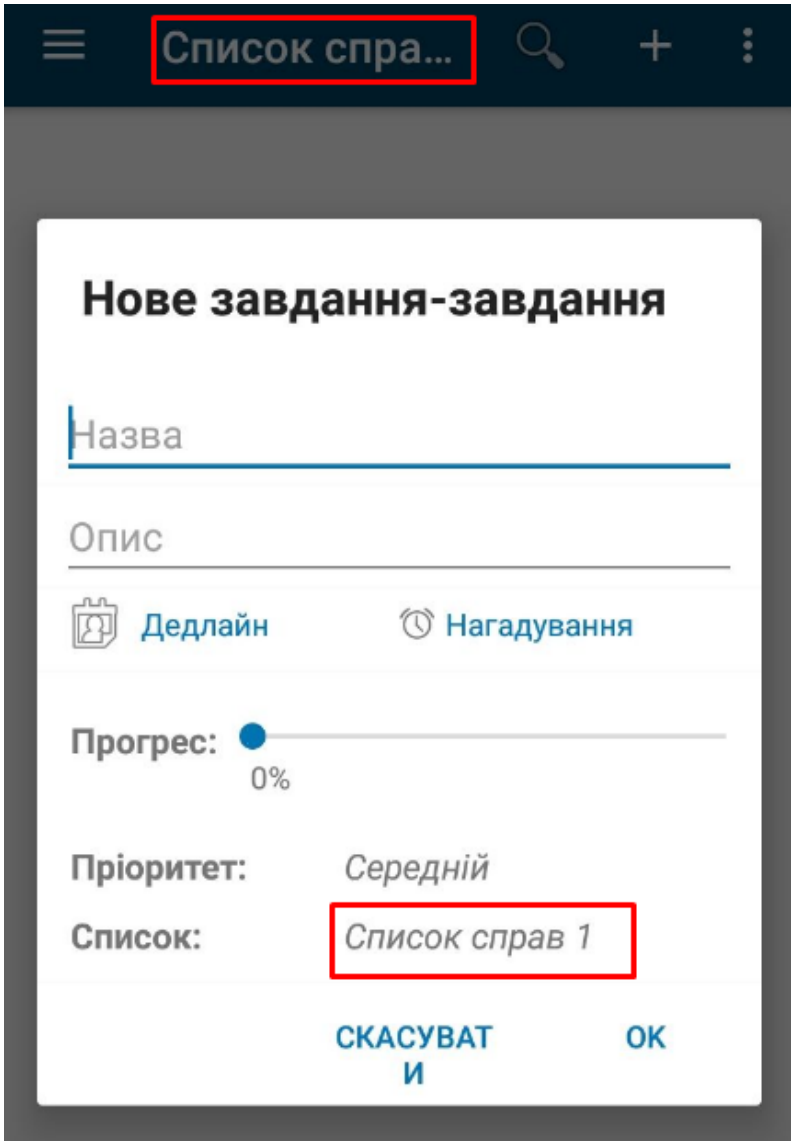

Рисунок 3.11 - Створення нової справи у існуючому списку справ

ВИСНОВКИ

В даній дипломній роботі було проаналізовано та розглянуто різноманітні способи та технології для створення застосунку. Було описано основні переваги та недоліки та порівняно трьох найбільш популярних мобільних операційних систем.

Також було наведено наочні та актуальні приклади найпопулярніших та розповсюджених інструментів для планування справ, у тому числі командного.

Було реалізовано окрім основних записів справи, списки справ та підзадачі, що дозволяэ точно групувати та налаштовувати власні справи, а також сортувати їх за пріорітетом або виконанням.

Робота виконана у IDE Android Studio за допомогою мови програмування Java. Для збереження даних використана реляційна база даних SQLite, що дозволяє зберігати дані на пристрої користувача.

Після проведення оптимізації застосунку було здійснено тестування на працездатність в різноманітних пристроях, як то: ZTE Blade V8 mini, Xiaomi Redmi Note 9. Всі фрагменти та блоки застосунку знаходились на відповідних позиціях, та відображалися коректно.

СПИСОК ЛІТЕРАТУРИ

1. Корпоративный планировщик Мегаплан

[URL]: https://megaplan.ru/facilities/
(дата звернення: 22.02.2021).

2. Персональный web-планировщик Миниплан

[URL]: http://miniplan.ru
(дата звернення: 03.03.2021).

3. Статистика мобільних операційних систем в Україні
[URL]: https://gs.statcounter.com/os-market-share/mobile/ukraine
(дата звернення: 05.03.2021).

4. Работа с XML в Android [Электронный ресурс]
[URL]: https://www.ibm.com/developerworks/ru/library/x-android/
(дата звернення: 09.03.2021).

5. Платформа для разработки мобильных приложений Microsoft Xamarin
[URL]: https://vc.ru/n/xamarin-free
(дата звернення: 09.03.2021).

6. Илья Бубнов. Популярные среды разработки и их недостатки
[URL]: https://gb.ru/posts/ide_negative
(дата звернення: 09.03.2021).

7. Введение в разработку Android-приложений
[URL]: http://www.intuit.ru/studies/courses/4462/988/lecture/14988
(дата звернення: 14.03.2021).

8. Брюс Эккель. Философия Java. СПб.: Питер, 2014. 640 с.
(дата звернення: 16.03.2021)

9. Google. Build a UI with Layout Editor
[URL]: https://developer.android.com/studio/write/layout-editor
(дата звернення: 16.03.2021).

10. Пол Дейтел, Харви Дейтел, Александер Уолд. Android для разработчиков. 3-е издание. СПб.: Питер, 2016. 512 с.
(дата звернення: 02.04.2021).

11. Діаграма варіантів використання мови UML
[URL]: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/
(дата звернення: 08.04.2021).

12. Android Studio
[URL]: https://android-developers.googleblog.com/2018/03/android-studio-3-1.html
(дата звернення: 15.04.2021).

13. Інтерфейс Android Studio
[URL]: https://developer.android.com/studio
(дата звернення: 12.05.2021).

14. Александра Кукуть. 12 топовых языков мобильной разработки по версии IEEE.
[URL]: https://dev.by/news/12-topovyh-yazykov-mobilnoi-razrabotki-po-versii-ieee
(дата звернення: 16.05.2021).

ДОДАТКИ

Додаток А. Код MainActivity

```
public class MainActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {


    public static final String COMMAND = "command";
    public static final int COMMAND_UPDATE = 3;
    public static final int COMMAND_RUN_TODO = 2;


    private static final String TAG = MainActivity.class.getSimpleName();

    // Keys
    private static final String KEY_TODO_LISTS = "restore_todo_list_key_with_savedinstancestate";
    private static final String KEY_CLICKED_LIST = "restore_clicked_list_with_savedinstancestate";
    private static final String KEY_DUMMY_LIST = "restore_dummy_list_with_savedinstancestate";
    private static final String KEY_IS_UNLOCKED = "restore_is_unlocked_key_with_savedinstancestate";
    private static final String KEY_UNLOCK_UNTIL = "restore_unlock_until_key_with_savedinstancestate";
    public static final String KEY_SELECTED_FRAGMENT_BY_NOTIFICATION = "fragment_choice";
```

```
    private static final String KEY_FRAGMENT_CONFIG_CHANGE_SAVE =
"current_fragment";
    private static final String KEY_ACTIVE_LIST = "KEY_ACTIVE_LIST";
    private static final String POMODORO_ACTION =
"org.wanderspool.privacyfriendlytodolist.TODO_ACTION";



    // Fragment administration
    private Fragment currentFragment;
    private FragmentManager fragmentManager = getSupportFragmentManager();

    //TodoTask administration
    private RelativeLayout rl;
    private ExpandableListView exLv;
    private TextView tv;
    private ExpandableTodoTaskAdapter expandableTodoTaskAdapter;
    private TextView initialAlert;
    private TextView secondAlert;
    private FloatingActionButton optionFab;



    // Database administration
    private DatabaseHelper dbHelper;

    private SharedPreferences mPref;

    // TodoList administration
    private ArrayList<TodoList> todoLists = new ArrayList<>();
```

```
    private TodoList dummyList; // use this list if you need a container for tasks that does not exist in the database (e.g. to show all tasks, tasks of today etc.)
    private TodoList clickedList; // reference of last clicked list for fragment
    private TodoRecyclerView mRecyclerView;
    private TodoListAdapter adapter;
    private MainActivity containerActivity;

    // Service that triggers notifications for upcoming tasks
    private ReminderService reminderService;

    // GUI
    private NavigationView navigationView;
    private NavigationView navigationBottomView;
    private Toolbar toolbar;
    private DrawerLayout drawer;

    // Others
    private boolean inList;
    boolean isInitialized = false;
    boolean isUnlocked = false;
    long unlockUntil = -1;
    private static final long UnlockPeriod = 30000; // keep the app unlocked for 30 seconds after switching to another activity (settings/help/about)
    int affectedRows;
    int notificationDone;
    private int activeList = -1;

    //Pomodoro
    private boolean pomodoroInstalled = false;
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        getMenuInflater().inflate(R.menu.search, menu);
        getMenuInflater().inflate(R.menu.add_list, menu);

        MenuItem searchItem = menu.findItem(R.id.ac_search);
        SearchView searchView = (SearchView)
MenuItemCompat.getActionView(searchItem);
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener()
{

            @Override
            public boolean onQueryTextSubmit(String query) {
                collapseAll();
                expandableTodoTaskAdapter.setQueryString(query);
                expandableTodoTaskAdapter.notifyDataSetChanged();
                return false;
            }

            @Override
            public boolean onQueryTextChange(String query) {
                collapseAll();
                expandableTodoTaskAdapter.setQueryString(query);
                expandableTodoTaskAdapter.notifyDataSetChanged();
                return false;
            }
```

```
    });

    MenuItem priotiryGroup = menu.findItem(R.id.ac_group_by_prio);
    priotiryGroup.setChecked(mPref.getBoolean("PRIORITY", false));

    MenuItem deadlineGroup = menu.findItem(R.id.ac_sort_by_deadline);
    deadlineGroup.setChecked(mPref.getBoolean("DEADLINE", false));

    return super.onCreateOptionsMenu(menu);
}



private void collapseAll()
{
    // collapse all elements on view change.
    // the expandable list view keeps the expanded indices, so other items
    // get expanded, when they get the old expanded index
    int groupCount = expandableTodoTaskAdapter.getGroupCount();
    for(int i = 0; i < groupCount; i++)
        exLv.collapseGroup(i);
}



@Override
public boolean onOptionsItemSelected(MenuItem item) {
    boolean checked = false;
    ExpandableTodoTaskAdapter.SortTypes sortType;
```

```
        sortType = ExpandableTodoTaskAdapter.SortTypes.DEADLINE;

        collapseAll();

        switch (item.getItemId()) {
            case R.id.ac_add:
                startListDialog();
                addListToNav();
                break;
            case R.id.ac_show_all_tasks:

expandableTodoTaskAdapter.setFilter(ExpandableTodoTaskAdapter.Filter.ALL_T
ASKS);
                expandableTodoTaskAdapter.notifyDataSetChanged();
                mPref.edit().putString("FILTER", "ALL_TASKS").commit();
                return true;
            case R.id.ac_show_open_tasks:

expandableTodoTaskAdapter.setFilter(ExpandableTodoTaskAdapter.Filter.OPEN_
TASKS);
                expandableTodoTaskAdapter.notifyDataSetChanged();
                mPref.edit().putString("FILTER", "OPEN_TASKS").commit();
                return true;
            case R.id.ac_show_completed_tasks:

expandableTodoTaskAdapter.setFilter(ExpandableTodoTaskAdapter.Filter.COMPL
ETED_TASKS);
                expandableTodoTaskAdapter.notifyDataSetChanged();
                mPref.edit().putString("FILTER", "COMPLETED_TASKS").commit();
```

```
            return true;
        case R.id.ac_group_by_prio:
            checked = !item.isChecked();
            item.setChecked(checked);
            sortType = ExpandableTodoTaskAdapter.SortTypes.PRIORITY;
            mPref.edit().putBoolean("PRIORITY", checked).commit();
            break;
        case R.id.ac_sort_by_deadline:
            checked = !item.isChecked();
            item.setChecked(checked);
            sortType = ExpandableTodoTaskAdapter.SortTypes.DEADLINE;
            mPref.edit().putBoolean("DEADLINE", checked).commit();
            break;
        default:
            return super.onOptionsItemSelected(item);
    }
```

Додаток Б. Код SplashActivity

```
package com.wanderspool.tdplanner.view;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;


   public class SplashActivity extends AppCompatActivity {

       @Override
       protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);

          Intent mainIntent = new Intent(SplashActivity.this, MainActivity.class);
          SplashActivity.this.startActivity(mainIntent);
          SplashActivity.this.finish();

       }

}
```

Додаток Б. Код TTodoList

```
package com.wanderspool.tdplanner.model.database.tables;

/*
 * This class is responsible to define sql table of To-Do lists.
 */

public final class TTodoList {

	private static final String TAG = TTodoList.class.getSimpleName();

	// columns + tablename
	public static final String TABLE_NAME = "todo_list";
	public static final String COLUMN_ID = "_id";
	public static final String COLUMN_NAME = "name";

	// sql table creation
	public static final String TABLE_CREATE = "CREATE TABLE " +
TABLE_NAME + "(" + COLUMN_ID +
			" INTEGER PRIMARY KEY AUTOINCREMENT, " +
COLUMN_NAME + " TEXT NOT NULL);";

}
```