Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного додатку для планування робочого часу студента

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Іщук Д.І. (прізвище та ініціали)

Керівник (підпис) Матійчук Л.П. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Крамар О.І. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Іщук Дем`ян Ігорович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного додатку для планування робочого часу студента

Керівник роботи Матійчук Любомир Павлович, к.е.н, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від 02 » березня 2021 року № 4/7-171
2. Термін подання студентом завершеної роботи 23 червня 2021р.
3. Вихідні дані до роботи Літературні джерела за тематикою дослідження

4. Зміст роботи (перелік питань, які потрібно розробити)
1 Аналіз предметної області та проектування ANDROID-додатку 1.1. Огляд і аналіз існуючих аналогів, що реалізують функції предметної області. 1.2. Специфікація вимог до системи. 2 Програмна реалізація додатку для обліку робочого часу студента на базі ОС ANDROID, тестування та дослідна експлуатація. 2.1.. Розроблення архітектури програмної системи. 2.2. Тестування та розгортання програмного додатку. 2.3. Інструкція користувача. 3. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Мета роботи. Система планування робочого часу. Огляд відомих додатків для ОС Android. Діаграма варіантів використання системи. Архітектура системи. Діаграма станів. Схема функціонування додатку під ОС Android. Діаграма класів системи. Схема реалізації з'єднання з базою даних SQLite. Перевірка наявності введеної інформаці у БД. Створення нового проекту в середовищі Eclipse для ОС Android. Написання коду в Eclipse на Java для ОС Android. Блок-схема алгоритму програми. Тестування системи в Robotium. Встановлення додатку. Інструкція користувача. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 7 червня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.06.2021	*Виконано*
2.	Підбір джерел предметної області та проектування	08.06.2021-09.06.2021	*Виконано*
3.	Переклад та опрацювання джерел щодо розробки мобільного додатку для планування робочого часу студента.	10.06.2021-11.06.2021	*Виконано*
4.	Виконання дослідження щодо розробки мобільного додатку для планування робочого часу студента.	12.06.2021-13.06.2021	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та проектування ANDROID-додатку»	14.06.2021-15.06.2021	*Виконано*
6.	Оформлення розділу «Програмна реалізація додатку для обліку робочого часу студента на базі ОС ANDROID, тестування та дослідна експлуатація..»	16.06.2021-17.06.2021	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»		*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	17.06.2021	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	18.06.2021	*Виконано*
10.	Нормоконтроль	19.06.2021	*Виконано*
11.	Перевірка на плагіат	19.06.2021	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.06.2021	*Виконано*
13.	Захист кваліфікаційної роботи	23.06.2021	

Студент _____ Іщук Д.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка мобільного додатку для планування робочого часу студента // Кваліфікаційна робота освітнього рівня «Бакалавр»// Іщук Дем`ян Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2021 // сторінок – 43, рисунки – 31, таблиць –2 , джерел –37.

Ключові слова: мобільний додаток, ОС Android, розробка Android додатку, середовище Eclipse, робочий час студента.

У першому розділі здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих програмних систем. Здійснено аналіз вимог до програмної системи.

Запропоновано архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних.

У другому розділі здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. По результатам тестування сформовано підсумок тестування. Також в даному розділі було розкрито питання встановлення та налаштування програмного забезпечення, а також вказані вимоги, дотримання яких необхідно для користування програмою.

ANNOTATION

Mobile application development for student work time planning // Qualification work of the educational level "Bachelor" // Ishchuk Demyan Igorovich // Ivan PuliyuTernopil National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science,group. CHc-42 // Ternopil, 2021 // pages –43, figures –31, tables –2, sources –37.

Keywords: mobile application, Android OS, Android application development, Eclipse environment, student working hours.

In the first section the description of the subject area, directions of activity is carried out. The composition of the functions included in the business process is determined, on the basis of which the business process management scheme is developed. The analysis of known software systems is carried out. The analysis of requirements to the software system is carried out.

The architecture of the software application is offered, which will allow to better understand the functions of its main parts. The structural scheme is created and described, the main components of which are: the level of the client, the level of business logic and the level of data. The functional structure of the system and its main elements - data processing modules are described.

The second section describes the testing procedures and their results, describes the test requirements for the software, as well as the identified defects. Based on the test results, the test result is formed. This section also covered the issues of software installation and configuration, as well as the requirements that must be met to use the program.

ЗМІСТ

ВСТУП

Android - це операційна система, яка розроблена для мобільних пристроїв (КПК, смартфонів, планшетів). Основна перевага системи, що вона розроблена на основі вільного ядра Linux. Операційну систему розроблено фірмою Android Inc., яку потім придбав Google. Весь вихідний код ОС безкоштовно надається розробникам мобільних застосунків. Мобільні пристрої на ОС Android набули широкого застосунку та популярні і впевнено та лідирують на ринку операційних систем для смарфонів та планшетів.

Актуальність запропонованого програмного продукту полягає в можливості спланувати з користю час студента.

Метою даної кваліфікаційної роботи є допомога перш за все, студентам та зацікавленим користувачам в наданні інструменту, який надасть змогу відстежувати важливі події під час робочого дня.

Інтерфейс програми є зручним і розроблений так, щоб повній мірі задовольняти усі вимоги користувачів, а саме – доступність для сприйняття, простий у використанні, тощо.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ ANDROID-ДОДАТКУ

1.1. Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

Розглянемо основні програми-аналоги до розроблюваного програмного додатку доя обліку робочого часу студента.

Springpad (рисунок 1.1). Дуже потужна програма.

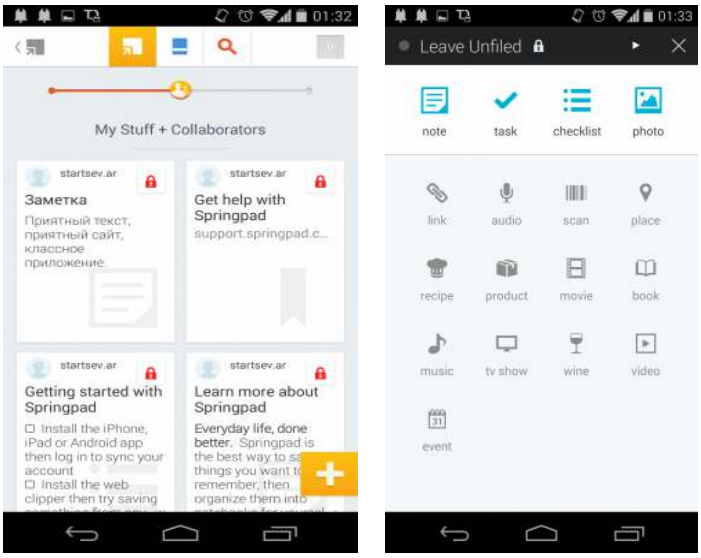

Рисунок 1.1 – Springpad

На відміну від Evernote, абсолютно безкоштовна, але дуже перевантажений інтерфейс. Просто губишся серед кнопок і вкладок. Функціонал величезний, більше навіть, ніж у Evernote. Не вистачає тільки розпізнавання тексту з аудіозапису.

ColorNote (рисунок 1.2). Досить-таки простенький додаток з малим набором функцій. З незвичайного тільки календар нотаток. Один з варіантів для мінімалістів.

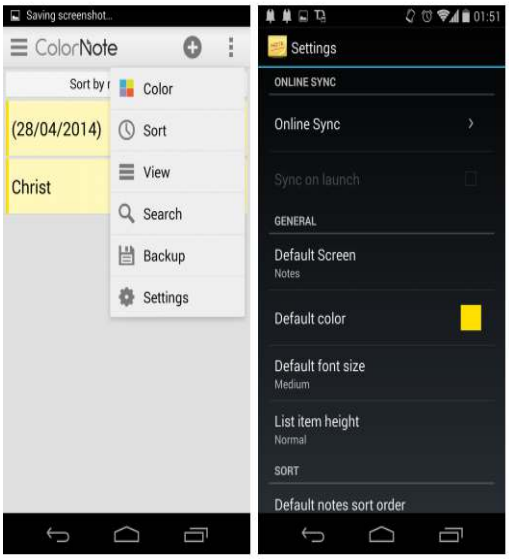

Рисунок 1.2 – ColorNote

Diigo (рисунок 1.3). Дуже простий додаток, орієнтований більше на роботу з сайтами. Дуже приємні враження залишив сайт програми.

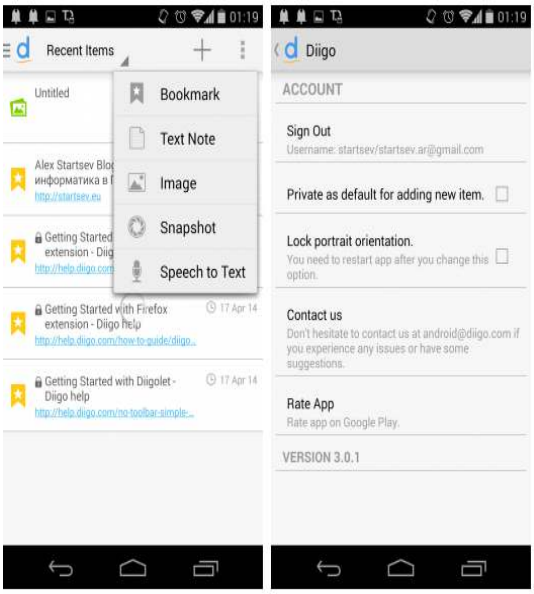

Рисунок 1.3 – Diigo

Thinkery (рисунок 1.4). Дуже мінімалістичний додаток. Функцій мізер, форматування тексту слабке. Ви можете зробити практично нічого. Мені абсолютно не сподобався додаток.

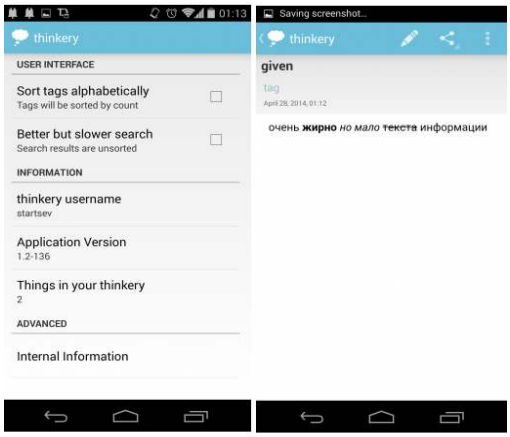

Рисунок 1.4 – Thinkery

GNotes. Досить-таки не переобтяжений зайвим додаток, але з усіма потрібними функціями для ведення простеньких списків. У цьому додатку є практично все, що потрібно. Особисто мені не вистачає нумерованих списків. Ще не зовсім зрозуміла настройка нагадувань. Я не знайшов, де можна поставити повідомлення для нотаток. Можна поставити пін або малюнок на певні блокноти. Це зручно.

Deepmemo (рисунок 1.5). Начебто прекрасний додаток з багатим набором функцій, але є відчуття, що все зроблено якось криво. Для розпізнання тексту чогось потрібен доступ до Gmail. Аудіозапис не може зробити, бо не знаходить чого-небудь для цього встановленого. На карті не показує нічого, але зате тут можна додавати відеонотатки. Не знаю, наскільки це корисно. Можна замітки за календарем дивитися. Є навіть статистика нотаток. жорстка прив'язка до Google. Можна робити репости в соціальні мережі.

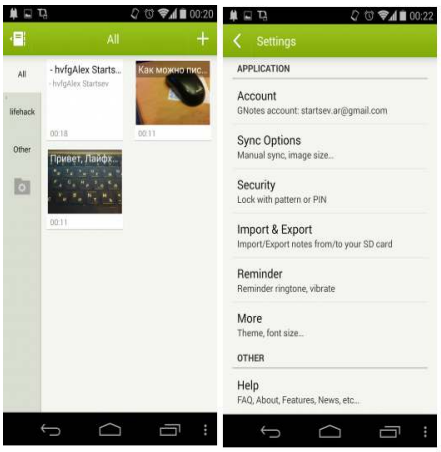

Рисунок 1.5 – Gnotes

Нагадування, фото, завдання, все круто. Все є. Але додаток не викликає бажання ним користуватися.

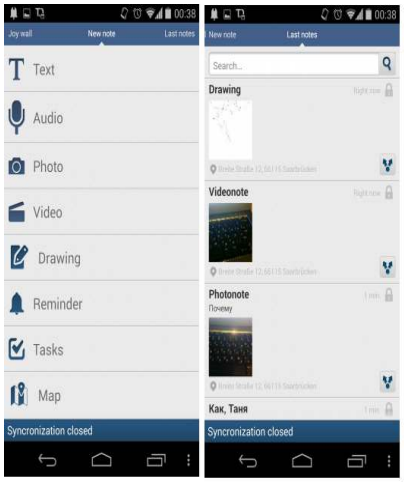

Рисунок 1.6 – Deepmemo

Wiz Note (рисунок 1.7). Дуже гарний екран блокування, та й сам додаток - один з найкрасивіших. Занадто довго завантажуються замітки з аудіозаписами. Але це тільки в перший раз. Додаток китайський, але це не завадило йому викликати у мене бажання, принаймні, спробувати.

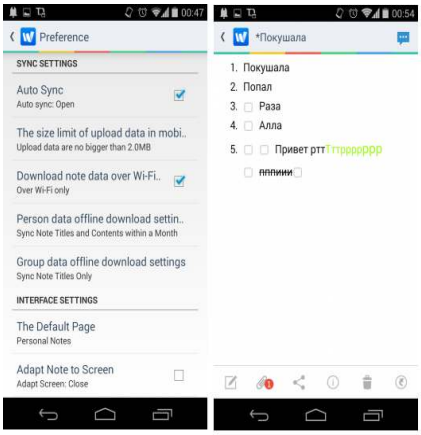

Рисунок 1.7 – Deepmemo

SimpleNote (рисунок 1.8). Додаток виправдовує свою назву. Він дійсно дуже корисний і дуже простий. У налаштуваннях ви можете вибрати більш компактне відображення заміток, принцип сортування та блокування PIN-кодом. Додаток створено для тих, хто хоче створювати найпростіші замітки.

Fetchnotes (рисунок 1.9). Досить-таки простенький додаток для найпростіших нотаток. Нагадує чимось твітер. Можна додавати хештеги (тільки англомовні). А ще є функція Контакти. Ви додаєте Nickname, і цей Nickname в список контактів програми. Можна вказати email або мобільний телефон, і людина отримає повідомлення.

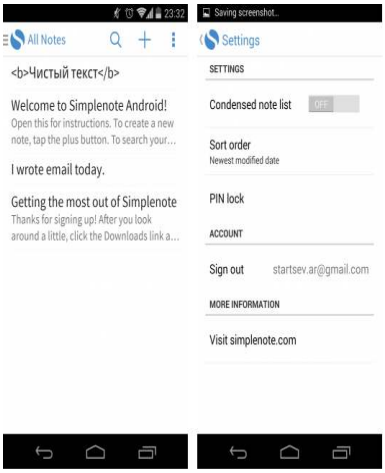

Рисунок 1.8 – SimpleNote

Зловив один баг програми: якщо ви перейдете до нотаток за хештегом або пов'язані з контактом, то повернутися до повного списку не вийде. Принаймні, це інтуїтивно незрозуміло.

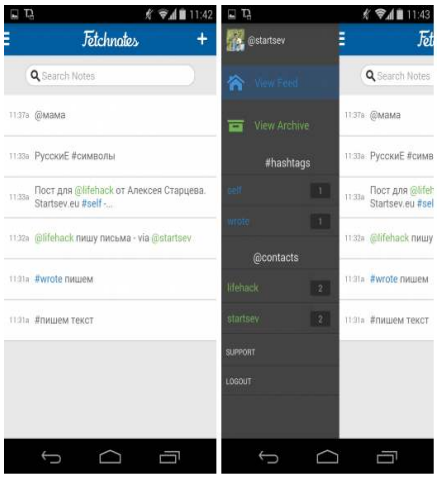

Рисунок 1.9 – Fetchnotes

WorkFlowy (рисунок 1.10). Ця програма для ведення нотаток вкладених списків. Ось серйозно, воно ідеально для тих, хто любить вкладені списки. Функціональності нуль, але вона і не потрібна тут. Ви неначе берете аркуш паперу пишіть на ньому список всіх ваших цілей на рік. Потім берете інший

аркуш і на ньому розписуєте детально що потрібно зробити за однією з цілей. А на першому аркуші створюєте посилання на цей другий. І так до нескінченності.

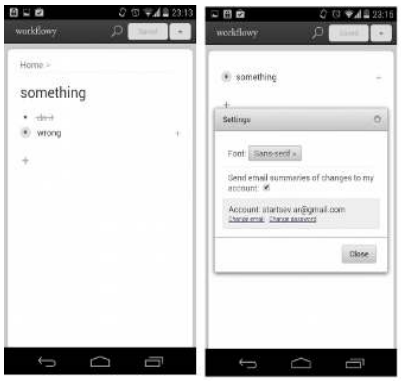

Рисунок 1.10 – WorkFlowy

Menote (рисунок 1.11). Додаток відразу при старті пропонує вам синхронізуватися з однією з облікових записів.

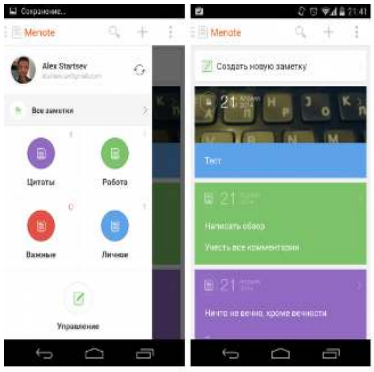

Рисунок 1.11 – Menote

У моєму випадку це був Google. Додаток виглядає дуже симпатично. Але от за функціональністю додаток підкачав. У налаштуваннях доступно тільки зміна інтервалу синхронізації з Google Drive.

Незважаючи на досить широкий спектр програм для обліку робочого часу, усі вони є досить нагромаджені по функціоналу і вимагають системних ресурсів, тому актуальним є розробка додатку, який би вирішував ці проблеми.

1.2. Специфікація вимог до системи

Специфікація вимог до програмної системи – це специфікація окремого програмного продукту, програми або набору програм, які виконують деякі дії в деякому середовищі. Тобто – це повний опис поведінки системи що розробляється [3].

В загальному випадку специфікація включає наступне:

- глосарій проекту;
- опис варіантів використання.

На рисунку 1.12 подано діаграму прецендентів взаємозвязків з програмою. Діаграма варіантів використання дозволяє швидко побачити основні функції, які буде виконувати програма після розробки.

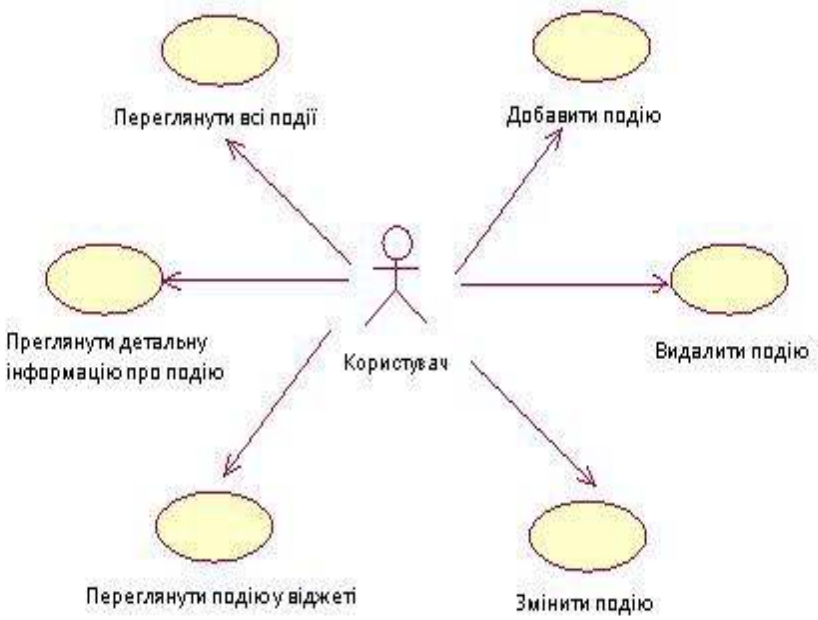

Рисунок 1.12 – Діаграма прецедентів взаємодії користувача з программою

Ця діаграма дозволяє виділити деякі об’єкти в системі та відобразити події основного сценарію програмного продукту, у вигляді послідовності повідомлень, якими обмінюються об’єкти.

Як видно з рисунку 1.13 користувач може виконати наступне:

а) добавити певну подію;

б) видалити якусь подію;

в) замінити вже створену подію;

г) перегляд усіх подій.

Розглянемо детальніше варіанти використання додатку, які описані у наступних таблицях (таблиця 1.1-1.2).

Таблиця 1.1

Варіант використання «Вхід в систему»

Контекст викор.	Вхід в систему
Дійові особи	Користувач
Передумова	Активне з’єднання з БД
Тригер	Немає
Сценарій	1. Запустити додаток 2. Дочекатися з’єднання з сервером
Постумова	Відображається вхід в систему

Таблиця 1.2

Варіант використання «Переглянути подію у віджеті»

Контекст використання	Переглянути подію у віджеті
Дійові особи	Користувач
Передумова	Активне з’єднання з БД
Тригер	Немає
Сценарій	1. Запустити додаток 2. Дочекатися з’єднання з сервером 3. Натиснути кнопку «Переглянути події» 4. Відображення списку подій
Постумова	Відображення списку подій

Висновки до розділу 1

У цьому розділі здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих програмних систем. Здійснено аналіз вимог до програмної системи.

РОЗДІЛ 2
ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ ДЛЯ ОБЛІКУ РОБОЧОГО ЧАСУ СТУДЕНТА НА БАЗІ ОС ANDROID, ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

2.1. Розроблення архітектури програмної системи

На рисунку 2.1 представлено архітектуру розроблюваного додатку, яка складається з декількох основних компонентів, які розглянемо детальніше.

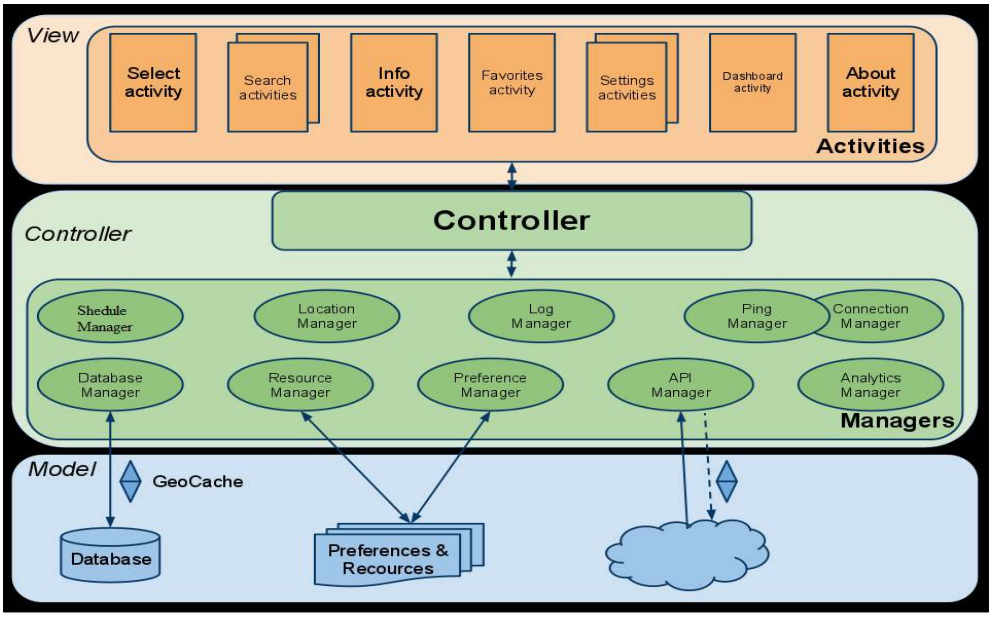

Рисунок 2.1. – Архітектура системи

Структурно можна розділити весь код програми на три компоненти:

- Model - представляє дані і реагує на запити від компонента Controller.
- Controller - інтерпретує отримані дані і повідомляє інших компонентів про різні події.
- View - відповідає за графічне представлення даних користувачеві

Model. Описує ключовий об'єкт докладання – Shedule, а також сюди можна віднести базу даних, яка зберігає відмічені користувачем події, що містить всі розклади сервісу, а також ресурси і налаштування програми.

Controller. Сюди відносяться 2 сутності - клас Controller і набір різних менеджерів. Клас Controller надає доступ компонента View до менеджерів, а

також виконує деякі функції інтепретації отриманих даних. Менеджери відповідають за обробку різних даних, отриманих ззовні. На даному етапі розробки є 10 різних менеджерів:

- Database manager;
- Location manager;
- Shedule manager;
- API manager;
- Connection & Ping manager;
- Analytics manager;
- Log manager;
- Resource manager;
- Preferences manager;
- Database manager;

Відповідає за з'єднання з базою даних, запис і читання події, які описують розподіл робочого часу студента.

Location manager. Зберігає останню виконану подію. Повідомляє про зміну розкладу, а також період наступних подій..

Shedule manager. Виводить розклад подій користувача і повідомляє про його зміну.

API manager. Відповідає за взаємодію додатку із зовнішніми сервісами goooogle, facebook.

Connection & Ping manager. Стежить за станом з'єднання з internet, а саме з goooogle, facebook, і оповіщає, коли зв'язок зникає або з'являється.

Analytics manager. Відповідає за роботу програми з сервісом оновлень.

Log manager. Організовує роботу системи логування програми.

Resource manager. Надає доступ до ресурсів програми таким як, малюнки, рядки, константи.

Preferences manager. Працює з настройками додатку, пише й читає їх.

На рисунку 2.2 представлено діаграму станів Android додатку, яка описує основні стани, в яких може перебувати додаток обліку робочого часу студента.

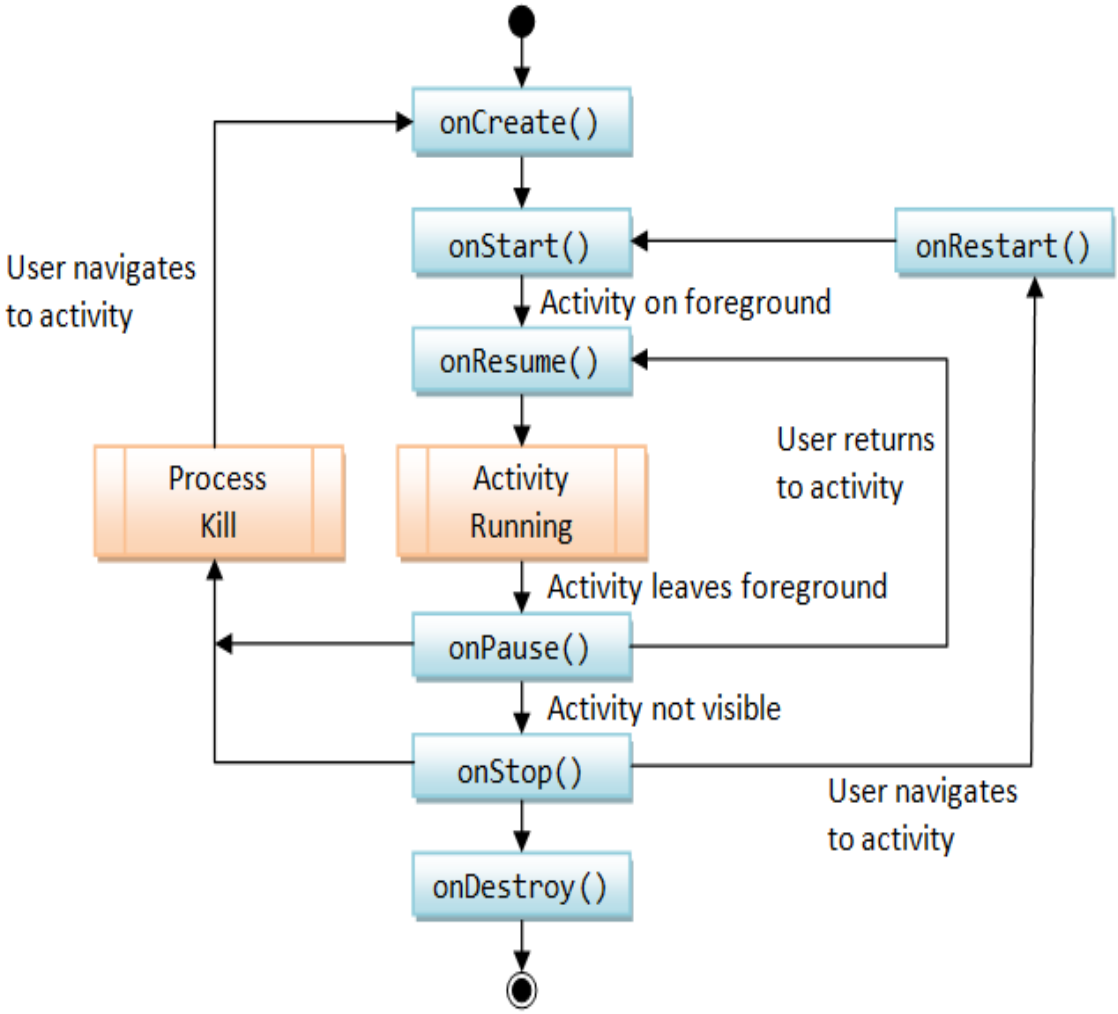

Рисунок 2.2 – Діаграма станів

Після запуску програми ми можемо згорнути і розгорнути її тоді, коли це буде потрібно, при цьому програма повністю збереже свій стан (звичайно, за умови, що не відбудеться системний збій). Якщо говорити про Android, то тут ситуація зовсім інша і ми в значно меншій мірі можемо контролювати перебіг використання додатку. Тому, перш ніж приступати до процесу розробки, ми повинні повністю знати і розуміти життєвий цикл Activity в Android додатках.

Діаграма життєвого циклу Android програми представлена на рисунку 2.3. Представлена діаграма, що демонструє те, як протікає життєвий цикл Activity в Android додатках. У сірих прямокутниках показані callback методи, які викликаються перш, ніж Activity перейде в яке-небудь стан.

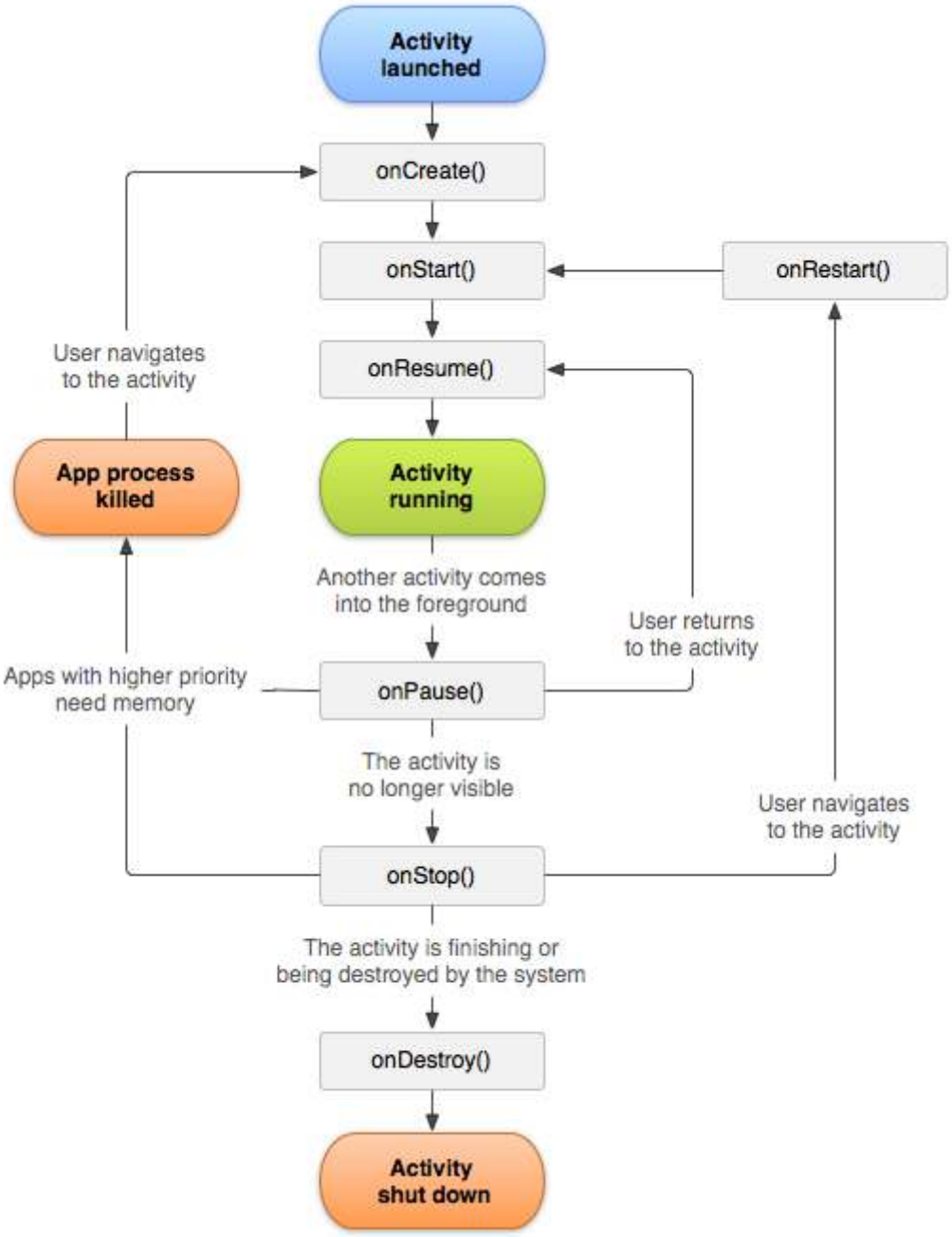

Рисунок 2.3. – Схема функціонування додатку під ОС Android

Дану діаграму демонструє протікання життєвого циклу Android програми, яку можна описати таким чином:

Коли ми запускаємо яку-небудь Activity, в першу чергу викликається метод onCreate (). Це той метод, в якому ми створюємо користувальницький інтерфейс і ініціалізуємо дані. Для відновлення стану UI даний метод приймає об'єкт типу Bundle як параметр.

Метод onStart() викликається перед тим, як користувач побачить Activity. В цей момент Activity ще неактивна.

При виклику методу onResume() Activity стає видимою для користувача і готова до взаємодії з ним. В даний момент Activity запущена, активна і знаходиться в самому верху стека.

За замовчуванням Activity може залишатися в стані паузи якщо:

Користувач натиснув кнопку «Home».

Інша Activity або повідомлення знаходяться зверху, не повністю закривають Activity, що знаходиться під ними.

Пристрій перейшов в режим сну.

Є три можливих варіанти виходу Activity зі стану паузи:

Користувач закриває нову Activity або повідомлення і Activity, яка перебуває у стані паузи, переходить у статус запущеної/активної за допомогою виклику методу onResume().

Activity вивантажене системою зважаючи на гостру нестачі пам'яті. В даному випадку, перед тим як Activity буде знищена, ніякі методи не викликаються. Необхідно повністю перезапустити її, викликавши метод onCreate() і відновити попередній стан за допомогою Bundle об'єкта.

У всіх інших випадках Activity переходить у стан зупинки з викликом методу onStop(). Дана подія виникає за замовчуванням у тих випадках, коли користувач натискає кнопку «Back» або нова Activity повністю закриває поточну.

Є три можливих сценарії виходу Activity з стан зупинки:

Система вивантажує програму з пам'яті для звільнення ресурсів. Activity в стані зупинки має вищі шанси бути знищеною системою, ніж у стані паузи. У тому випадку, якщо це відбудеться, буде потрібно почати цикл з виклику onCreate().

Activity перезапущено викликом onRestart (), або викликом onStart () і onResume() методів, що в свою чергу означає, що користувач знову повернувся

до Activity. В останньому випадку користувальницький інтерфейс зберігає свій стан і не вимагає відновлення.

Викликаний метод onDestroy(), після чого Activity знищується. Це останній метод, який ми можемо викликати до того, як Activity буде знищена. Також даний метод може бути викликаний в тому випадку, якщо Activity завершила свою роботу або була знищена системою.

На рисунку 2.4. представлено діаграму класів розроблюваного додатку.

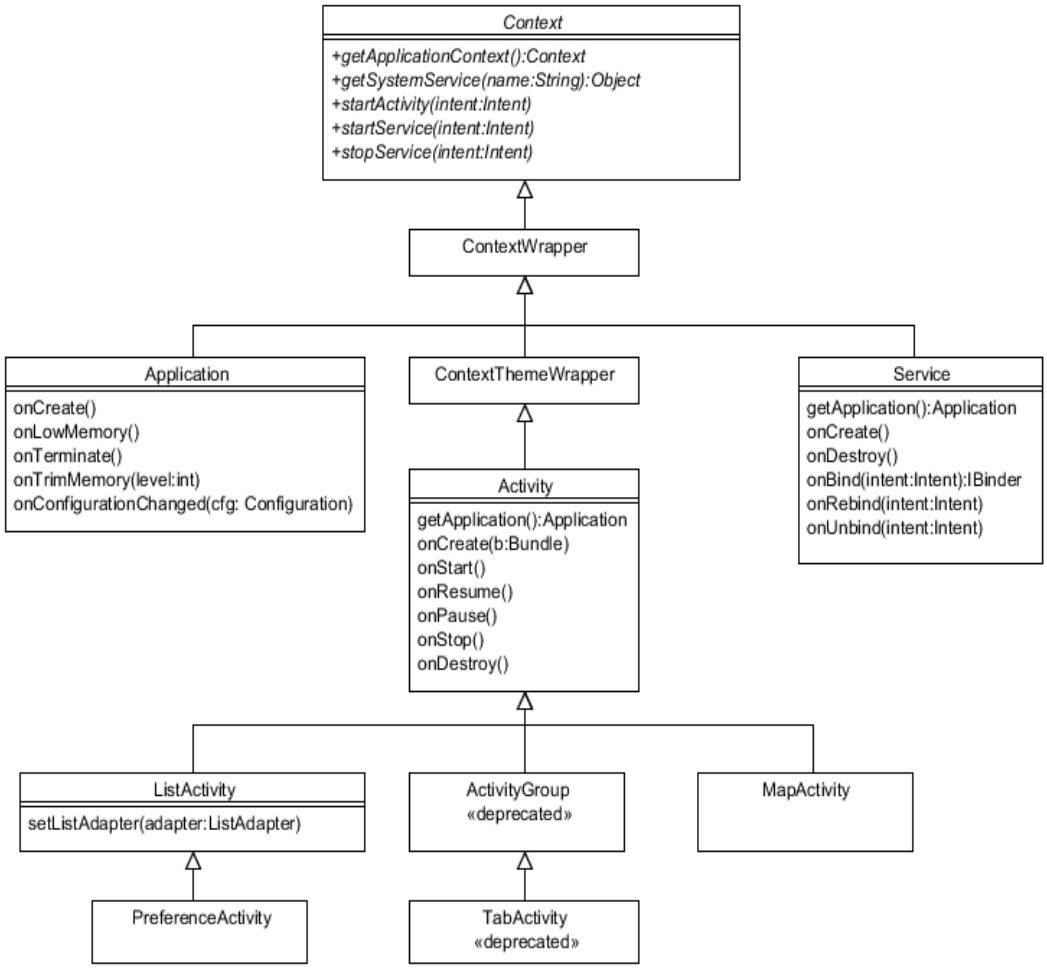

Рисунок 2.4. – Діаграма класів системи

2.2. Тестування та розгортання програмного додатку

Тестування є важливою частиною процесу розробки додатків. Для Android воно має особливу важливість, оскільки використовувані пристрої досить сильно відрізняються один від одного за такими параметрами:

- розмір і розширення екрана;
- версія Android;
- типорозмір пристрою;
- набір інструкцій процессора;
- наявність передньої камери, підтримка NFC, зовнішня клавіатура та ін.

Програми для Android слід тестувати на багатьох пристроях. Процес тестування додатків може включати тести різних типів. Розглянемо ручне функціональне тестування. Тестувальнику необхідно уважно перевірити всі функції програми, а потім скинути пристрій в початковий стан. Ці дії потрібно повторити для кожної програми і для кожного пристрою. При ручній роботі цей процес займає дуже багато часу.

Можна регулярно проводити автоматичне тестування функціональності без додаткових витрат. Наприклад, можна на ніч запускати тестування збірки на всіх пристроях, а вранці аналізувати результати і виправляти помилки.

Розглянемо декілька засобів для автоматичного тестування функціональності нашого додатку. При цьому мова йде тільки про засоби, що входять до складу Android SDK або поширюються з відкритим вихідним кодом.

Принцип автоматичного тестування. Наше завдання - з найбільшою точністю автоматизувати дії, що виконуються вручну. Розглянемо ці дії. Будемо використовувати кілька додатків і кілька пристроїв з Android.

Для кожної програми і кожного пристрою потрібно виконати наступні дії:

- встановити додаток на пристрій;
- запустити программу;
- протестувати додаток, використовуючи вибраний метод;
- видалити додаток;
- скинути пристрій в початковий стан;

На кожному етапі потрібно збирати та аналізувати дані (журнали та знімки екрану). Нижче описуються засоби для автоматизації цих дій.

У складі Android SDK передбачено дві програми для управління пристроями з Android: ADB і monkeyrunner. Розглянемо автоматизацію дій, застосовуваних при тестуванні вручну.

Управління пристроями Android за допомогою ADB. ADB (Android Debug Bridge) - це програма командного рядка для управління пристроями з Android. Програма ADB знаходиться в папці <android_sdk> / platform-tools /. Слід вказати цю папку у змінній середовища PATH (рисунок 2.5).

Встановлюємо і налаштовуємо Android SDK, потім підключаємо до комп'ютера пристрої з Android і виконуємо команду:

adb devices

За цією командою буде показаний список всіх підключених пристроїв з Android. Якщо список пристроїв не порожній, то ADB працює вірно.

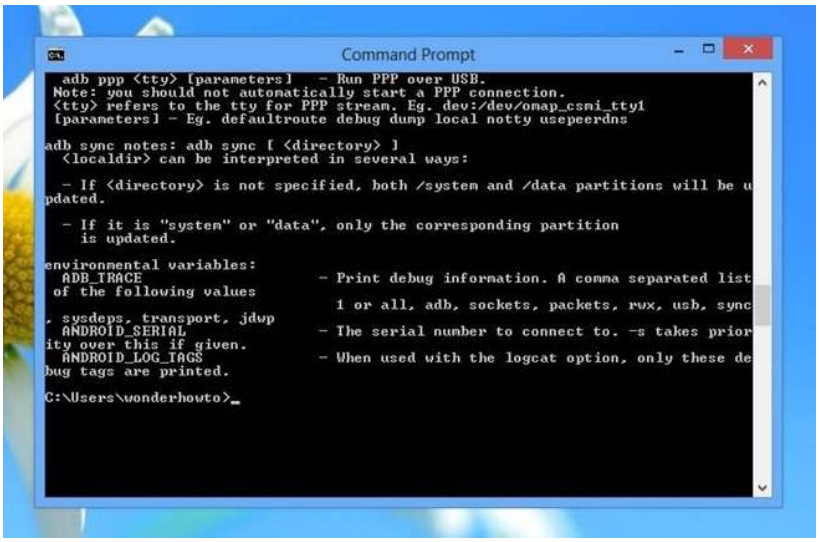

Рисунок 2.5– Встановлення Android Debug Bridge

Серійний номер пристрою можна отримати у вихідних даних команди «adb devices». Параметр -s дозволяє працювати одночасно з декількома підключеними пристроями.

До більшості папок на пристрої з Android можливий доступ тільки для читання. Доступ на запис доступний тільки до папок / sdcard (але з цієї папки неможливо запускати програми) та /data/local/tmp.

Logcat - це засіб командного рядка для читання журналів з пристроїв Android. Читати журнали з пристрою (блокування до натискання клавіш Ctrl-C): adb logcat. Очистити буфер журналу на пристрої: adb logcat –c. Записати буфер журналу на пристрої (відображення поточного вмісту буфера, без блокування):adb logcat –d. Програма monkeyrunner надає інтерфейси API для скриптів, керуючих пристроями Android (рисунок 2.6). Можна написати скрипт Python, що встановлює додаток Android, що запускає його, що отримує знімки екрану і зберігає їх на комп'ютері. Для запуску скриптів Monkeyrunner використовує Jython.

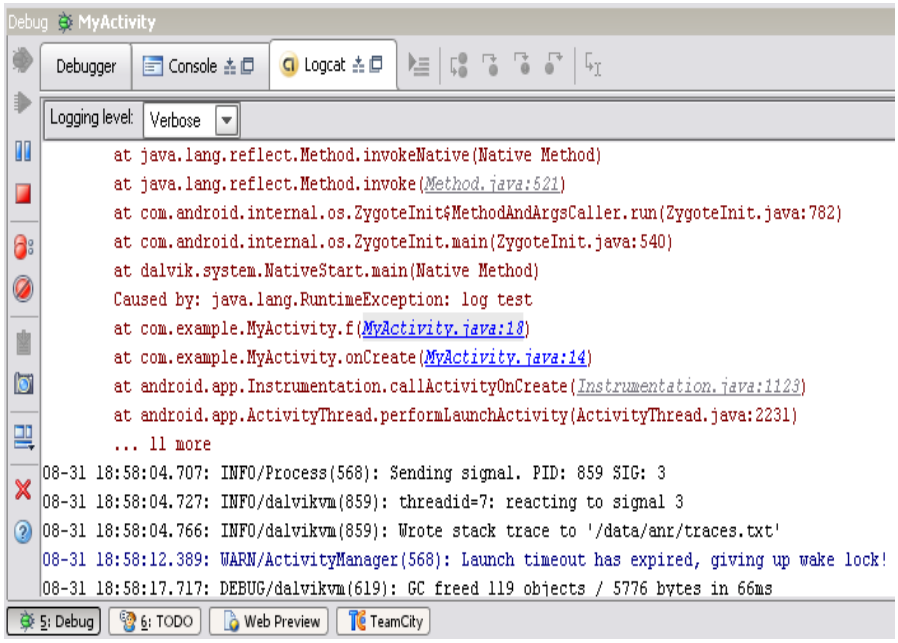

Рисунок 2.6 – Перевірка журналу за допомогою Android Logcat

Програма screencap зберігає знімок поточного екрану в графічний файл (рисунок 2.7):

```
1 adb shell screencap /sdcard/screen.png
2 adb pull /sdcard/screen.png screen.png
3 adb shell rm /sdcard /screen.png
```

Рисунок 2.7 – Запис результатів тестування в графічний файл

Скрипт для запуску і читання журналів розробленого додатку представлено на рисунку 2.8

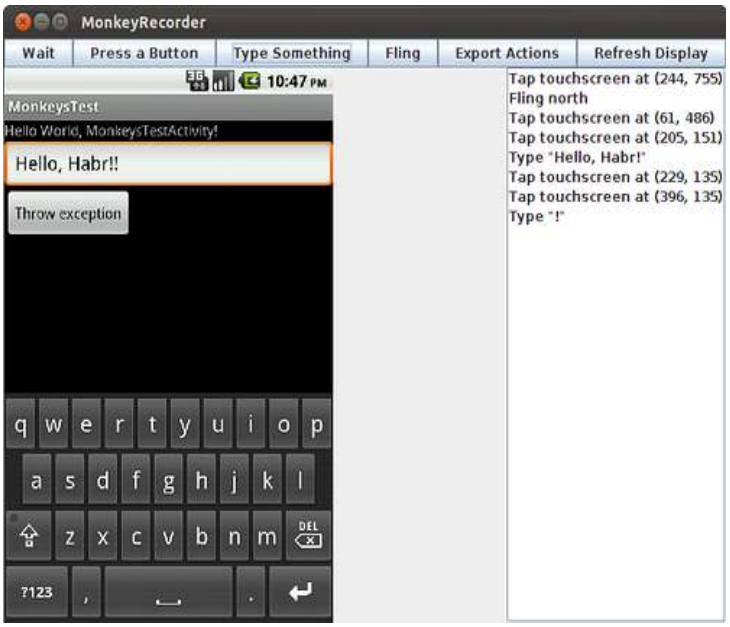

Рисунок 2.8 – Запис результатів тестування в Monkeyrunner

```
01 # coding: utf-8
02 from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
03
04 def log(fn, device):
05     msg = device.shell('logcat -d')
06     f_log = open(fn, 'at')
07     if msg is None:
08         msg = 'None'
09     f_log.write(msg.encode('utf-8'))
10     f_log.close()
11     device.shell('logcat -c')
12
13 if __name__ == '__main__':
14     device = MonkeyRunner.waitForConnection()
15     device.shell('logcat -c') # Clear logs buffer
16     # ...
17     log('example.log', device) # Write logs
```

Рисунок 2.9 – Скрипт зчитування журналів тестування

Тестування за допомогою Robotium (рисунок 2.10). Robotium не входить до складу Android SDK, ця програма поширюється з відкритим вихідним кодом. Сценарії Robotium визначають дії на рівні користувача інтерфейсу додатків, а не на рівні пристрої введення.

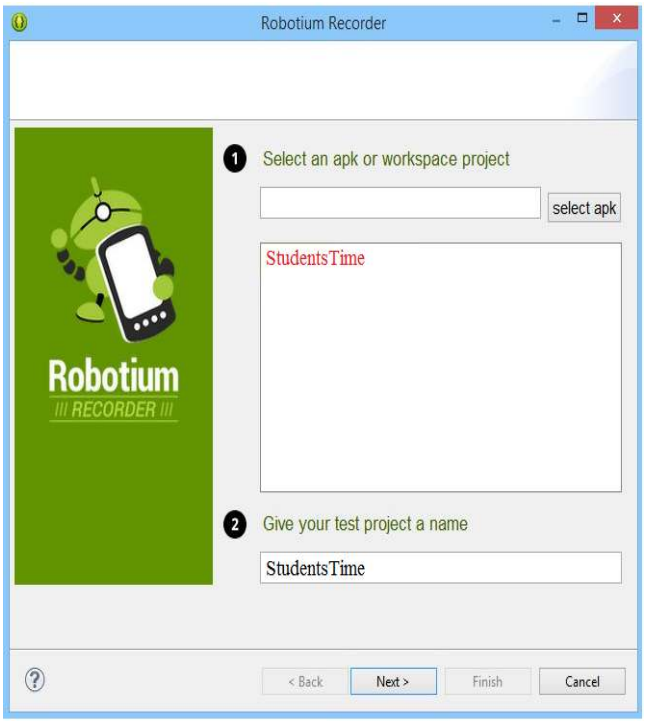

Рисунок 2.10 – Тестування за допомогою Robotium

Наприклад, в сценарії потрібно торкнутися кнопки «ОК». У цьому випадку скрипт monkeyrunner буде побудований таким чином: «імітувати дотик екрану в точці з координатами (x0, y0)». Скрипт Robotium буде побудований інакше: «натиснути кнопку з текстом« ОК ».

Опис дій на рівні інтерфейсу дозволяє зробити тестовий скрипт незалежним від розташування і розмірів елементів інтерфейсу, дозволи та орієнтації екрану.

Крім того, в Robotium можна перевіряти реакцію програми на дії. Наприклад, припустимо, що після натискання кнопки «ОК» повинен з'явитися список з елементом «Елемент 1».

Автоматичне тестування не замінює інші типи тестування. Продуманий процес тестування, що поєднує різні методи, в тому числі і автоматичне тестування, є найважливішою частиною процесу розробки високоякісних додатків.

Для роботи програмного додатку «Система планування робочого часу студента на базі ОС Android» на першому етапі спочатку встановим її на мобільний пристрій.

Інсталювати програми на ОС Android можна наступними способами:

1) скориставшись програмою «AppsInstaller» що подано (рисунок 2.11);

2) а також скориставшись файловим менеджером (ES File, ASTRO File Manager (рисунок 2.12), Glance, Explorer, X-plore (рисунок 2.13) та ін.).

Рисунок 2.11 – Встановлення додатків за допомогою AppsInstaller

Рисунок 2.12 – Встановлення додатків за допомогою ASTRO File Manager

Щоб можна було здійснити інсталювння програми першим способом потрібно виконати такі дії (рисунок 2.13):

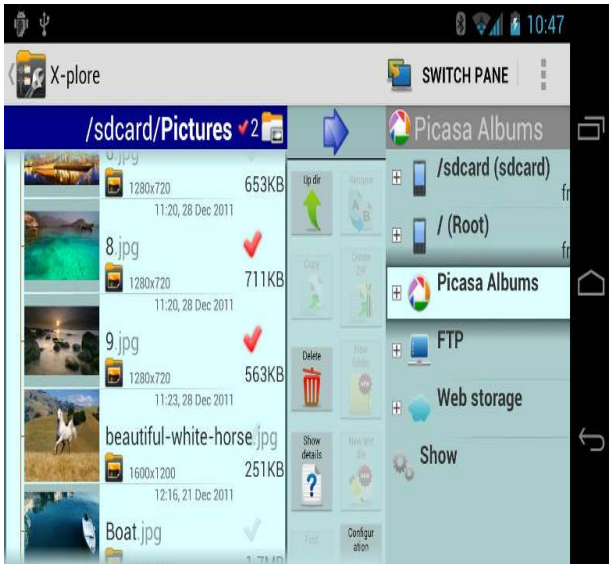

Рисунок 2.13 – Встановлення додатків за допомогою X-plore

– здійснити процедуру завантаження програмного додатку «AppsInstaller»;

– наступний крок скопіювати «*.apk» даний файл на карту памяті;

– заключний етап скористатись пошуком, щоб файл за допомогою «AppsInstaller» віднайти та інсталювати.

Для того, щоб ми могли скористатись другим способом інсталяції потрібно виконати наступні кроки:

– спочатку завантажити файловий менеджер;

– потім скопіювати «*.apk» файл на картку пам’яті;

– за допомогою файлового менеджера відшукати файл;

– запустити файл для інсталяції програми.

Програма поширюється шляхом копіювання файлу і вимагає попередньої інсталяції на мобільний пристрій користувача.

Щоб програма почала працювати потрібно встановити віджет на робочий стіл. Здійснити дану процедуру можна наступним чином:

– вибрати робочий стіл на якому потрібно встановити віджет;

– здійснити тривале натискання на місці в якому буде знаходитися віджет;

– в меню вибрати пункт «Widgets» (рисунок 2.14);

– з слідуючого списка вибрати віджет.

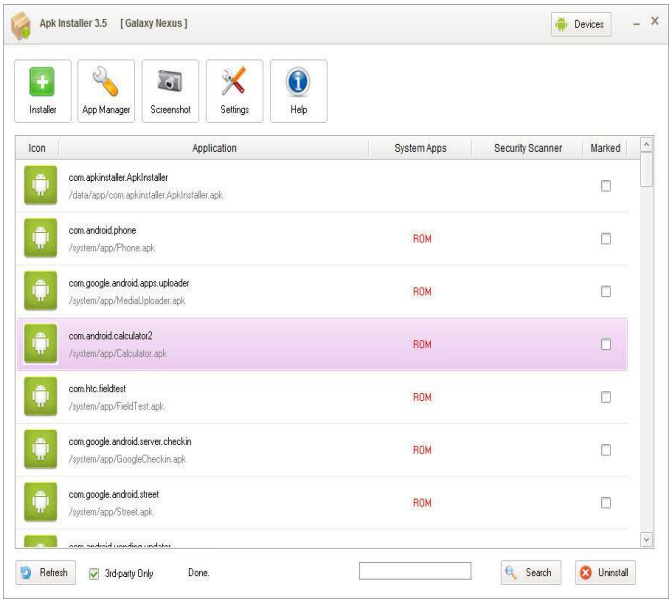

Рисунок 2.14 – Встановлення додатків за допомогою AppsInstaller

Розроблений програмний продукт орієнтований на роботу в ОС Android, тому для коректної роботи програми необхідне стабільне функціонування ОС.

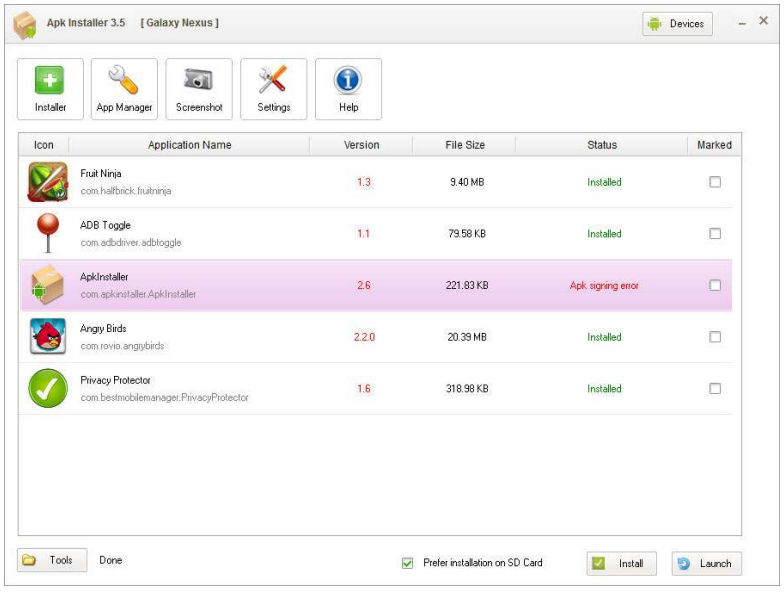

Рисунок 2.15 – Встановлення додатків за допомогою AppsInstaller

Програма створена в середовищі Eclipse. До технічних засобів відносимо самартфони, які працюють під ОС Android, на яких буде виконуватись програма. Мінімальними вимогами до апаратної частини самартфона, за яких програма працюватиме, можна вважати:

– процесор ARM 11 – 800 МГц;
– об’єм оперативної пам’яті 256 Мб;
– графічний процесор Adreno 200;
– об’єм внутрішньої пам’яті 158 МБ.

2.3. Інструкція користувача

Функціональні можливості програми наступні:

- це перш за все можливість відображення у віджеті;
- можливість перегляду усіх подій;
- можливість перегляду усієї інформації про заплановану подію;
- можливість добавити будь-яку подію;
- внесення змін в уже створену подію.

Детальний вигляд програми наведено на рисунку 2.16.

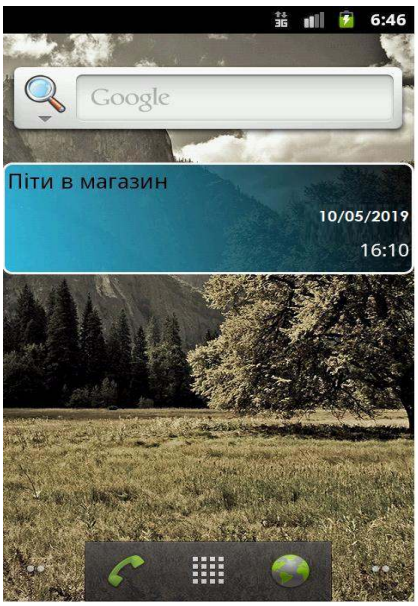

Рисунок 2.16 – Вигляд віджета на робочому столі емулятора

Для здійснення процедури перевірки роботи програми було використано персональний комп’ютер з попередньо встановленим програмним

забезпеченням.

Віджет дає можливість відображення подій, яка в найблищому часі повинна відбутися рисунок 2.17.

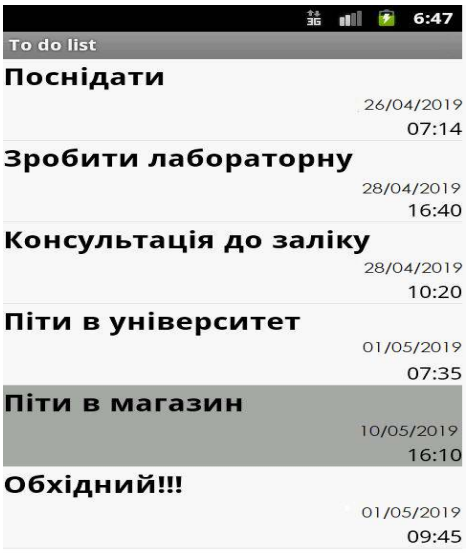

Рисунок 2.17 – Вигляд вікна усіх подій

В даному вікні можна скористатись вибором і обрати будь-яку подію, а також переглянути детальну інформацію про неї наведено на рисунку 2.14.

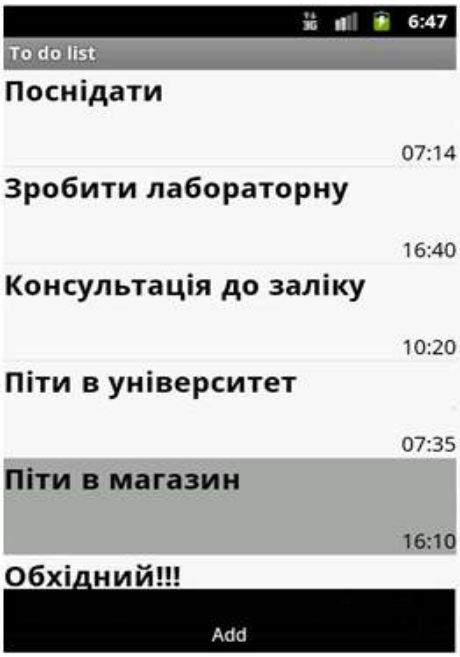

Рисунок 2.18 – Пункт меню для створення нової події

Щоб запланувати нову подію необхідно натиснути на кнопку «Menu» на емуляторі. З’явиться нове вікно із формою для внесення інформації про подію,

наведено на рисунку 2.19.

Рисунок 2.19 – Вікно для внесення інформації про нову подію

Розроблений додаток є зручним і практичним у використані, а також дозволяє ефективно працювати, не вимагаючи значних системних ресурсів та швидкісного Інтернету.

Висновки до розділу 2

У даному розділі запропоновано архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних.

Обґрунтовано технологію, мову програмування та розроблено програмну систему. Обґрунтовано засоби розробки додатку та здійснено опис основних програмних модулів системи.

А також здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. По результатам тестування сформовано підсумок тестування. Також в даному розділі було розкрито питання встановлення та налаштування програмного

забезпечення, а також вказані вимоги, дотримання яких необхідно для користування програмою, також описана інструкція користувача для роботи із додатком.

РОЗДІЛ 3
БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Інженерний захист персоналу об’єкту та населення. Правила застосування.

Функціонування на території нашої країни численних об’єктів підвищеної небезпеки, переважно в зонах з підвищеною концентрацією населення, різко посилює небезпеку великих техногенних катастроф, провокує та збільшує негативну дію особливо небезпечних стихійних явищ. Щороку втрати від таких надзвичайних ситуацій вимірюються тисячами людських життів, мільярдними збитками та невиправною шкодою для природного середовища.

Масштабність і багатогранність завдань щодо протидії сучасним природним і техногенним загрозам вимагають висококваліфікованої, технічно оснащеної, мобільної державної системи цивільного захисту.

Така система визнана складовою національної безпеки, а виконання її завдань - важливим обов’язком органів виконавчої влади всіх рівнів. Відповідно, з метою наближення до світових стандартів, від назви основного інструмента державної політики у сфері протидії наслідкам катастроф - цивільна оборона ми переходимо до назви - цивільний захист. І це не випадково. Сукупність завдань, що стоять перед службами цивільної оборони багатьох країн, більше пов’язані сьогодні з проблемами мирного часу, що дозволяє говорити про цивільний захист населення і територій, а не про цивільну оборону у воєнний час.

До захисних споруд цивільного захисту належать:

1) сховище – герметична споруда для захисту людей, в якій протягом певного часу створюються умови, що виключають вплив на них небезпечних факторів, які виникають внаслідок надзвичайної ситуації, воєнних (бойових) дій та терористичних актів;

2) протирадіаційне укриття – негерметична споруда для захисту людей, в якій створюються умови, що виключають вплив на них іонізуючого опромінення у разі радіоактивного забруднення місцевості;

3) швидкоспоруджувана захисна споруда цивільного захисту – захисна споруда, що зводиться із спеціальних конструкцій за короткий час для захисту людей від дії засобів ураження в особливий період.

Для захисту людей від деяких факторів небезпеки, що виникають внаслідок надзвичайних ситуацій у мирний час, та дії засобів ураження в особливий період також використовуються споруди подвійного призначення та найпростіші укриття.

Споруда подвійного призначення – це наземна або підземна споруда, що може бути використана за основним функціональним призначенням і для захисту населення.

Найпростіше укриття – це фортифікаційна споруда, цокольне або підвальне приміщення, що знижує комбіноване ураження людей від небезпечних наслідків надзвичайних ситуацій, а також від дії засобів ураження в особливий період.

За місцем розташування сховища можуть бути вбудовані і окремо розташовані. До вбудованих відносяться сховища, які розташовані в підвальних приміщеннях будинків, а до окремо розташованих – сховища, які розташовані за межами будинків і споруд.

Для вирішення питань щодо укриття населення в захисних спорудах цивільного захисту центральні органи виконавчої влади, місцеві державні адміністрації, органи місцевого самоврядування та суб’єкти господарювання завчасно створюють фонд таких споруд.

Порядок створення, утримання фонду захисних споруд цивільного захисту та ведення його обліку визначається Кабінетом Міністрів України. Проектування, будівництво, пристосування і розміщення захисних споруд та об’єктів подвійного призначення здійснюються згідно з нормами, які розробляються відповідно до Закону України "Про будівельні норми".

Вимоги щодо утримання та експлуатації захисних споруд визначаються центральним органом виконавчої влади, який забезпечує формування та реалізує державну політику у сфері цивільного захисту.

Утримання захисних споруд цивільного захисту у готовності до використання за призначенням здійснюється суб'єктами господарювання, на балансі яких вони перебувають (у тому числі споруд, що не увійшли до їх статутних капіталів у процесі приватизації (корпоратизації), за рахунок власних коштів.

У разі використання однієї захисної споруди кількома суб'єктами господарювання вони беруть участь в утриманні споруди відповідно до укладених між ними договорів.

Захисні споруди цивільного захисту можуть використовуватися у мирний час для господарських, культурних і побутових потреб у порядку, що визначається Кабінетом Міністрів України. З моменту виключення захисної споруди із фонду споруд цивільного захисту вона втрачає статус захисної споруди цивільного захисту.

Володіння, користування та розпорядження спорудами, які втратили статус захисних споруд цивільного захисту, здійснюється відповідно до закону. Захисні споруди цивільного захисту державної та комунальної власності не підлягають приватизації (відчуженню).

3.2. Запобігання наслідкам аварії на виробництвах із застосуванням аміаку.

Для отримання низьких температур технологічними схемами компресорного цеху багатьох промислових підприємств харчової та переробної промисловості передбачено застосування токсичної речовини – аміаку.

Потенційна небезпека таких технологічних схем полягає у порушенні герметичності обладнання і трубопроводів, що містять аміак. Найбільшу небезпеку з цієї точки зору являють собою руйнування автоцистерн з рідким

аміаком; руйнування напірних трубопроводів компресорів; порушення герметичності відокремлювачів рідини, лінійних та циркуляційних ресиверів, запірної арматури, батарей холодильних камер.

Наслідком таких аварій є виникнення загазованості виробничого приміщення, відкритого майданчика цеху і підприємства в цілому, а також прилеглих житлових районів; утворення вибухонебезпечної суміші аміаку з повітрям в приміщеннях, внаслідок чого можливі вибухи і пожежі.

Джерелами локальних викидів аміаку можуть служити процеси стиснення газоподібного і нагнітання рідкого аміаку, а також зливно-наливні операції.

Аварії (катастрофи) на підприємствах, транспорті та продуктопроводах можуть супроводжуватися викидом (виливом) в атмосферу і на прилеглу територію небезпечних хімічних речовин (НХР), таких як хлор, аміак, синильна кислота, фосген, сірчаний ангідрид та інші. Це являє серйозну небезпеку для населення, заражене повітря уражає органи дихання, а також очі, шкіру та інші органи.

Фактори небезпеки викиду (розливу) хімічно небезпечних речовин: забруднення навколишнього середовища, небезпека для всього живого, що опинилося на забрудненій місцевості (загибель людей, тварин, знищення посівів та ін.), крім того, внаслідок можливого хімічного вибуху виникнення сильних руйнувань на значній території.

Аміак – безбарвний газ з характерним різким запахом і їдким смаком. Він майже у два рази легший від повітря.

За звичайних умов аміак легко зріджується під тиском, а при випаровуванні поглинає тепло – сильно охолоджується. Ця властивість використовується у промислових та побутових холодильниках на м'ясокомбінатах, молокозаводах, овочевих базах, тобто там, де є необхідність в охолодженій продукції. Крім того, він є сировиною багатьох хімічних виробництв. Аміак зберігається і транспортується у зрідженому стані.

Він один з найважливіших продуктів сучасної хімічної промисловості. Головною галуззю його застосування є виробництво нітратної кислоти і

азотних добрив. Крім того, аміак використовують для виробництва багатьох інших хімічних продуктів. Останнім часом зріджений аміак і водний розчин аміаку стали широко застосовувати безпосередньо як азотне добриво.

Як рідина, аміак легший за воду, має меншу густину і при виході на повітря утворює слабкий дим. Вогненебезпечний, створює вибухові суміші з повітрям, отруйний. Особливо небезпечний для очей.

У випадку розливу рідкого аміаку і його концентрованих розчинів не можна доторкатися до розлитої рідини.

Ознаки отруєння аміаком:

- нежить, кашель, важке дихання, задуха;

- підвищене серцебиття, порушена частота пульсу;

- при контакті з рідким аміаком виникає обмороження, можливий опік з пухирями, виразки.

Перша допомога при отруєнні аміаком:

- одягніть протигаз і виведіть ураженого на свіже повітря;

- дайте подихати зволоженим повітрям (теплими водяними парами 10%-ного розчину ментолу в хлороформі);

- дайте йому теплого молока з «Боржомі» або харчовою содою;

- при задусі необхідний кисень;

- при спазмі голосових щілин забезпечте тепло на ділянку шиї, теплі ванночки, інгаляцію;

- при зупинці дихання проведіть серцево-легеневу реанімацію;

- при потраплянні в очі − промийте водою або 0,5-1%-ним розчином квасців, вазеліновою або оливковою олією;

- при уражені шкіри − обмийте чистою водою, зробіть примочки з 5%-ного розчину оцтової, лимонної або соляної кислоти.

При отруєнні аміаком винести потерпілого із зони зараження, шкіру, рот, ніс промити водою. В очі закапати по дві-три краплі 30% альбуциду, в ніс - оливкове масло. При необхідності відправити потерпілого до медичного закладу.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було створено програмне забезпечення “Розробка мобільного додатку для планування робочого часу студента”. Головною метою створення програмного продукту було використання даного ресурсу студентом, який потребує допоміжного програмного забезпечення для планування свого робочого дня.

Запропонована розробка реалізована в середовищі Eclipse. Основний принцип, що користувачі зможуть відслідковувати усі важливі події. Для усіх подій відображаються поля для заповнення: заголовок події, короткий опис і час події. Програма реалізована на основі операційної системи Android.

В ході виконання кваліфікаційної роботи отримано наступні практичні результати:

1. Розроблено прикладне програмне забезпечення, що дає змогу добавляти, видаляти чи корегувати події, які за допомогою програми відслідковуються та відображаються у віджеті.

2. Тестування програмного забезпечення проілюструвало, що програма працює коректно та виконує усі поставлені перед нею задачі.

ПЕРЕЛІК ДЖЕРЕЛ

1. A.V. Burdakov, U.A. Grigorev, A.D. Ploutenko. Comparison of table joinexecution time for parallel DBMS and MapReduce, Software Engineering / 811:Parallel and Distributed Computing and Networks / 816: Artificial Intelligence andApplications Proceedings (March 18 – 18, 2014, Innsbruck, Austria), ACTA Press,2014.

2. Aleksey Burdakov, Uriy Grigorev, Andrey Ploutenko, Eugene Ttsviashchenko"Estimation Models for NoSQL Database Consistency Characteristics", 24thEuromicro International Conference on Parallel, Distributed, andNetwork-BasedProcessing (PDP), 2016, pp. 35-42, doi: 10.1109/PDP.2016.23.

3. Аткинсон Л., Сураски З. PHP5: Библиотека профессионала, 3–е издание: Пер. з англ. – М.: Вильямс, 2006. – 944 с.

4. Bettina Kemme. Gustavo Alonso. Database Replication: a Tale of Research acrossCommunities. Proceedings of the VLDB Endowment, Vol. 3, No. 1. P. 5-12.

5. Buasilovsky, P. Adaptive and intelligent Web-based educational systems / P. Buasilovsky, CPeylo// International Journal of Artificial Intelligence in Education. Special Issue on Adaptive and Intelligent Web-based Educational Systems -2003. -№13 (2-4). -P. 159-172.

6. Codd, Edgar F.: A Relational Model of Data for Large Shared Data Banks. In:Communications of the ACM 13 (1970), June, No. 6, p. 377–387.

7. Converse T., Park J., Morgan C. PHP5 and MySQL Bible. – Indianapolis, Canada: Wiley Publishing Inc., 2004. – 1083 p.

8. Ситник В.Ф. Системи підтримки прийняття рішень. – К.: Техніка, 2005. –164с.

9. Ситник В.Ф. Системи підтримки прийняття рішень. – К.: Техніка, 2005. –164с.

10.F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T.Chandra, A. Fikes, and R. E. Gruber, Bigtable: a distributed storage system forstructured data, Proceedings of the 7th Conference on USENIX Symposium

onOperating Systems Design and Implementation - Volume 7 (Seattle, WA,November 06 - 08, 2006), USENIX Association, Berkeley, CA, 15-15, 2006.

11.FreshKnowledge CMS - онтологічно-орієнтована система керування контентом, розроблена здобувачем [Електронний ресурс]. - Режим доступу : http://www.freshknowledge.net.

12.FreshKnowledge CMS - онтологічно-орієнтована система керування контентом, розроблена здобувачем [Електронний ресурс]. - Режим доступу : http://www.freshknowledge.net.

13.Marshall, B. Convergence of Knowledge Management and E-Learning: the GetSmart Experience [Електронний ресурс] / Marshall, B., et al. // JCDL. -Houston, 2003. - Режим доступу: http://ai.bpa.arizona.edu/go/intranet/ Publication/JCDL-2003-Marshall.pdf.

14. Murray, T. AuthoringIntelligentTutoringSystems: AnAnalysisoftheStateoftheArt / T. Murray. // InternationalJournalofArtificialIntelligenceinEducation. -1999.-№10-P. 98-129

15.Y. Sheffer, R. Holz, P. Saint-Andre Summarizing Known Attacks on Transport Layer Security ÇTLS) and Datagram TLS ÇDTLS).

16.Андон, Ф. И. Логические модели интеллектуальных информационных систем / Ф. И. Андон, А. Е. Яшунин, В. А. Резниченко. - К: Наукова думка.-1999.-397 с.

17. Андон, Ф. И. Логические модели интеллектуальных информационных систем / Ф. И. Андон, А. Е. Яшунин, В. А. Резниченко. - К: Наукова думка.-1999.-397 с.

18.Дейт К. Дж. Введение в системы баз данных: Пер. с англ. – 6–е изд. – К.: Диалектика, 2007. – 784 с.

19. Елизаренко, Г. Н. Проектирование компьютерных курсов обучення: концепция, язык, структура / Г. Н. Елизаренко. - К.: НТУУ «КПИ», 2001.

20.Елизаренко, Г. Н. Проектирование компьютерных курсов обучення: концепция, язык, структура / Г. Н. Елизаренко. - К.: НТУУ «КПИ», 2001.

21.Іщук В. І.Сетифікація програмного забезпечення на основі моделі якості/ В. І. Іщук, І. О. Боднарчук // Збірник тез доповідей Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 16-17 листопада 2017 року. — Т. : ТНТУ, 2017. — Том 2. — С. 73–74. — (Комп’ютерно-інформаційні технології та системи зв’язку).

22.Комп’ютерна система аутентифікації осіб/ В. А. Марків, Г. М. Осухівська, Ю. З. Лещишин, А. М. Луцків // Матеріали наукової конференції ТНТУ ім. І. Пулюя, 17-18 травня 2017 року. — Т. : ТНТУ, 2017. — С. 90–91. — (Інформаційні технології).

23.Краковецкий А. Кластеризация: алгоритмы k-means и c-means [Електронний ресурс] / Александр Краковецкий // Habrahabr. – 2009. – Режим доступу до ресурсу: http://habrahabr.ua/post/67078/.

24.Лапінський В. В., Габрусєв В. Ю. Основи операційних систем: Посібник для студентів. – К.: Вища школа, 2007. – 96 с.

25.Лаврищева, Е. М. Методы программирования: теория, инженерия, практика / Е. М. Лаврищева. - К.: Наукова думка. - 2006. - 451 с.

26. Лаврищева, Е. М. Методы программирования: теория, инженерия, практика / Е. М. Лаврищева. - К.: Наукова думка. - 2006. - 451 с.

27.Луцків А. М.Архітектури комп'ютерних систем опрацювання великих даних/ А. Луцків, В. Діденко // Матеріали науково-технічної конференції „Інформаційні моделі, системи та технології“, 12-13 грудня 2018 року. — Т. : ТНТУ, 2018. — С. 75. — (Комп’ютерні системи та мережі).

28. Люгер, Джордж, Ф. Искусственный интеллект: стратегии и методы решення сложных проблем / Люгер, Джордж, Ф. - 4-е издание.: Пер. с англ.- М.: Издательский дом «Вильяме», 2005. - 864 с.

29.Маєвський О. В. Будова та експлуатація ПК : Конспект лекцій / Маєвський О.В., Мацюк О.В., Смакула І.З. — Тернопіль : ПМП ”РОМС-К” , 2010 — 368 с. — ISBN 9665670786.

30.Маклаков С.В. BPwin и ERwin: CASE-средства для разработки информационных систем. – М.: Диалог-Мифи, 1999. - 295 с.

31.Марценко С. В.Математичне моделювання та статистичні методи обробки даних вимірювань в задачах моніторингу електронавантаження/ Марценко С.В. — Тернопіль , 2011 — 20 с.

32.Митчелл М., Оулдем Д., Самьюэл А. Программирование для Linux. Профессиональный поход. – М.: Вильямс, 2002. – 288 с.

33.Назаревич О.Комп'ютерні технології І CAD-програми в навчальному процесі: проблеми і методика/ Назаревич О., Назаревич Б. // Вісник Тернопільського державного технічного університету. — том 14. — с.176-178

34. Олецький О. В. Застосування формальних моделей онтологій для формалізації інформаційних потоків у системах управління контентом / О. В. Олецький //Теоретичні та прикладні аспекти побудови програмних систем. Матеріали міжнародної конференції TAAPSD’2005. Київ, 7-9 грудня 2005 р. - С. 26-29.

35.Основи програмування. Курс лекцій для студентів першого рівня вищої освіти за спеціальністю No 121 Інженерія програмного забезпечення/ Уклад.: М.Р. Петрик, О.Ю.Петрик - Тернопіль: ТНТУ 2018- 64 с.

36.Самойлов, В. Д. Модельное конструирование компьютерных приложений / В. Д. Самойлов. - К.: Наукова думка. - 2007. - 198 с.

37. Самойлов, В. Д. Модельное конструирование компьютерных приложений / В. Д. Самойлов. - К.: Наукова думка. - 2007. - 198 с.

ДОДАТКИ

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

app.js

```
const http = require('http');
const path = require('path');
const express = require('express');
const mongoose = require('mongoose');
const socketIO = require('socket.io');
const config = require('./config.json');
const router = require('./domain/router');
const {createRoom, joinRoom, leftRoom, getRooms, updateSlide} =
require('./domain/events');
const port = process.env.PORT || config.port;
50
const app = express();
const server = http.createServer(app);
const io = socketIO(server);
app.use(express.static(path.join(__dirname, 'public')));
// підключенння маршрутів
app.use('/', router);
// підключенння обобника подій

io.on('connection', (socket) => {
socket.on('createRoom', createRoom(socket));
socket.on('joinRoom', joinRoom(socket));
socket.on('getRooms', getRooms);
socket.on('updateSlide', updateSlide(socket));
socket.on('leftRoom', leftRoom(socket));
socket.on('disconnect', leftRoom(socket));
});
// старт сервера
server.listen(port, () => {
mongoose.connect('mongodb://localhost:27017/test');
console.log(`Server is up on port ${port}`)
});
```

router.js

```
const path = require('path');
const router = require('express').Router();
const axios = require('axios');
const formidable = require('formidable');
const PDFImage = require('pdf-image').PDFImage;
const token = require('../helpers.js');
const Room = require('../models/room.js');
// обробник маршруту завантаження презентації
```

```
router.post('/upload', function (req, res) {
const form = new formidable.IncomingForm();
form.uploadDir = path.join(__dirname, 'uploads');
form.parse(req, (err, fields, files) => {
if (err) return res.sendStatus(500);
const fieldName = Object.keys(files)[0];
const filePath = files[fieldName].path;
const pdfImage = new PDFImage(filePath);
pdfImage.convertPage(0).then((imagePath) => {
res.json({
attachment: filePath,
preview: imagePath
});
});
});
});
// обробник маршруту скачування презентаціями
router.get('/download/:filename', function (req, res) {
const filename = req.params.filename;
res.sendFile(path.join(__dirname, 'uploads', filename))
});
// обробник маршруту отримання архівів
router.get('/archives', function (req, res) {
const url =
`https://api.opentok.com/v2/project/${config.api_key}/archive`;
axios.get(url, {headers: {'X-OPENTOK-AUTH': token()}})
.then(response => {
const obj = {};
51
response.data.items
.filter(archive => archive.url !== null)
.forEach(archive => obj[archive.sessionId] = archive);
Room.find().then(rooms => {
rooms.forEach(room => {
if (obj[room.roomId]) {
obj[room.roomId].name = room.roomName;
}
});
res.json(Object.values(obj));
}
);
})
.catch(error => res.sendStatus(500));
});
module.exports = router;
```

```
events.js
const fs = require('fs');
const path = require('path');
const Room = require('../models/room');
const openTok = require('./opentok');
// функція обробки події створення кімнати

const createRoom = socket => {

// roomInfo - об'єкт з полями roomName - ім'я кімнати, attachment -
url

// презентації на сервері, preview - url зображення 1 слайда на
сервері
return (roomInfo, callback) => {
openTok.createSession({mediaMode: 'routed', archiveMode: 'always'},
(err, session) => {
if (err) return callback(err);
const {roomName, attachment, preview} = JSON.parse(roomInfo);
fs.renameSync(attachment, path.join(__dirname, 'uploads',
`${session.sessionId}.pdf`));
fs.renameSync(preview, path.join(__dirname, 'uploads',
`${session.sessionId}.png`));
const room = new Room({
roomId: session.sessionId,
roomName: roomName,
presenter: {
userId: openTok.generateToken(session.sessionId),
socketId: socket.id
},
viewers: [],
createdAt: new Date().getTime()
});
room.save().then(room => {
socket.join(room.roomId);
callback(room);
});
});
};
};
// функція обробки входу в кімнату
const joinRoom = socket => {
// roomId – ідентифікатор кімнати
return (roomId, callback) => {
Room.findOneAndUpdate({roomId: roomId}, {
$push: {
'viewers': {
```

```
userId: openTok.generateToken(roomId),
socketId: socket.id
}
}
52
}).then(room => {
socket.join(roomId);
callback(room);
});
}
};
// функція обробки виходу з кімнати
const leftRoom = socket => {
return () => {
Room.findOne({
$or: [
{
'presenter.socketId': socket.id
}, {
viewers: {
$elemMatch: {
socketId: socket.id
}
}
}
]
}).then(room => {
if (room) {
if (room.presenter.socketId === socket.id) {
room.presenter = {};
room.save();
socket.broadcast.to(room.roomId).emit('forceDisconnect');
} else {
room.viewers = room.viewers.filter(viewer => viewer.socketId !==
socket.id);
}
socket.leave(room.roomId, null);
}
});
};
};
// функція обробки отримання кімнат

const getRooms = callback => Room.find ({presenter: {$ ne: {}}}).
then (rooms => callback (rooms));
```

```
// функція обробки зміни поточного слайда
const updateSlide = socket => {
// slide - номер слайда
return slide => {
Room.findOne({
'presenter.socketId': socket.id
}).then(room =>
socket.broadcast.to(room.roomId).emit('updateSlide', slide));
}
};
module.exports = {
createRoom, joinRoom, leftRoom, getRooms, updateSlide
};
config.json
{
«api_key»: «46112232»,
«api_secret»: «7c1ef51e404b2f7a299ccadc4792bc6bb9c01048»,
«port»: 4567
}
build.gradle
53
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'kotlin-kapt'
android {
compileSdkVersion 'android-P'
defaultConfig {
applicationId «com.pvasiliev.rtcpresentation»
minSdkVersion 21
targetSdkVersion 26
versionCode 1
versionName «1.0»
testInstrumentationRunner
«android.support.test.runner.AndroidJUnitRunner»
}
buildTypes {
debug {
buildConfigField «String», «API_KEY», «\»46112232\»»
buildConfigField «String», «NODE_ADDRESS»,
«\»http://192.168.43.196:4567\»»
}
release {
buildConfigField «String», «NODE_ADDRESS», «\»https://evening-tor-
```

```
33683.herokuapp.com\»»
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
}
}
compileOptions {
sourceCompatibility 1.8
targetCompatibility 1.8
}
}
dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation «org.jetbrains.kotlin:kotlin-stdlib-
jre7:$kotlin_version»
implementation 'com.android.support:appcompat-v7:28.0.0-alpha1'
implementation 'com.android.support:design:28.0.0-alpha1'
implementation 'com.android.support.constraint:constraint-
layout:1.1.0'
implementation 'com.squareup.retrofit2:retrofit:2.4.0'
implementation 'com.squareup.retrofit2:adapter-rxjava2:2.4.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation('io.socket:socket.io-client:1.0.0') {
exclude group: 'org.json', module: 'json'
}
implementation 'com.opentok.android:opentok-android-sdk:2.13.0'
implementation 'com.github.stephanenicolas.toothpick:toothpick-
runtime:1.1.3'
kapt 'com.github.stephanenicolas.toothpick:toothpick-
compiler:1.1.3'
implementation 'com.arello-mobile:moxy:1.5.3'
implementation 'com.arello-mobile:moxy-app-compat:1.5.3'
kapt 'com.arello-mobile:moxy-compiler:1.5.3'
implementation 'ru.terrakok.cicerone:cicerone:3.0.0'
implementation 'io.reactivex.rxjava2:rxandroid:2.0.2'
implementation «io.reactivex.rxjava2:rxjava:2.1.13»
implementation 'io.reactivex.rxjava2:rxkotlin:2.2.0'
implementation 'com.github.bumptech.glide:glide:4.7.1'
implementation 'com.afollestad.material-dialogs:core:0.9.6.0'
implementation 'com.afollestad.material-dialogs:commons:0.9.6.0'
implementation 'com.github.marlonlom:timeago:3.0.2'
54
implementation 'com.github.chrisbanes:PhotoView:2.1.3'
implementation 'com.jakewharton.timber:timber:4.7.0'
testImplementation 'junit:junit:4.12'
```

```
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
}
RtcApi.kt
interface RtcApi {
@POST(«/upload»)
@Multipart
fun upload(@Part presentation: MultipartBody.Part):
Single<JsonObject>
@GET(«/download/{filename}»)
fun download(@Path(«filename») filename: String): Single<File>
@GET(«/archives»)
fun listArchives(): Single<List<Archive>>
}
FileConverterFactory.kt
// клас для десеріалізації файлу
class FileConverterFactory : Converter.Factory() {
override fun responseBodyConverter(type: Type, annotations:
Array<out Annotation>?, retrofit:
Retrofit?): Converter<ResponseBody, *>? {
return if (type == File::class.java) {
FileConverterFactory.FileConverter
} else {
null
}
}
companion object FileConverter : Converter<ResponseBody, File> {
override fun convert(responseBody: ResponseBody): File {
val inputStream = responseBody.byteStream()
val file = File.createTempFile("tmp", FORMAT_PDF)
val outputStream = FileOutputStream(file)
inputStream.copyTo(outputStream)
return file
}
}
}
RoomRepository.kt
class RoomRepository @Inject constructor(private val socket:
Socket, private val api: RtcApi) {
companion object {
private const val KEY_FILENAME = "filename"
private const val KEY_ROOM_NAME = "roomName"
}
// функція для створення кімнати
```

```
// file - файл презентації

fun createRoom (file: File): Single <Room> =

uploadFile (file) .flatMap {uploadInfo (it)} .doOnSuccess
{createPresentationScope (it, file)}

// функція для входу в кімнату

// room - інформацією про кімнату
fun joinRoom(room: Room): Single<Pair<Room, File>> =
downloadFile(room).flatMap { downloadInfo(room, it) }.doOnSuccess {
createPresentationScope(it.first, it.second) }
// функція для отримання списку кімнат
fun getRooms(): Single<List<Room>> =
55
Single.create { emitter ->
socket.emit(GET_ROOMS, Ack {
val rooms: List<Room> = Gson().fromJson(it.joinToString(), object :
TypeToken<List<Room>>() {}.type)
emitter.onSuccess(rooms)
})
}
// функція для виходу з кімнати
fun leaveRoom() = socket.emit(LEFT_ROOM)
fun getRoomConnection(): Single<Boolean> =
Single.create { emitter ->
socket.on(EVENT_DISCONNECT, {
emitter.onSuccess(true)
socket.off(EVENT_DISCONNECT)
socket.off(FORCE_DISCONNECT)
})
socket.on(FORCE_DISCONNECT, {
emitter.onSuccess(true)
socket.off(EVENT_DISCONNECT)
socket.off(FORCE_DISCONNECT)
})
}
// функція для отримання списку архівуються лекцій

fun getArchives (): Single <List <Archive >> = api.listArchives ()

// функція для скачування файлу

// room - інформація про кімнату
private fun downloadFile(room: Room): Single<File> =
api.download("${room.roomId}.pdf")
private fun downloadInfo(room: Room, file: File): Single<Pair<Room,
File>> =
```

```
Single.create { emitter ->
socket.emit(JOIN_ROOM, room.roomId, Ack {
val room = Gson().fromJson(it.joinToString(), Room::class.java)
emitter.onSuccess(room to file)
})
}
// функція для завантаження файлів

// file - файл презентації
private fun uploadFile(file: File): Single<JsonObject> {
val filePart = RequestBody.create(MediaType.parse(MIME_TYPE_PDF),
file)
val body = MultipartBody.Part.createFormData(KEY_FILENAME,
file.name, filePart)
return api.upload(body).doOnSuccess { it.addProperty(KEY_ROOM_NAME,
file.name) }
}
private fun uploadInfo(roomInfo: JsonObject): Single<Room> =
Single.create { emitter ->
socket.emit(CREATE_ROOM, roomInfo, Ack {
val room = Gson().fromJson(it.joinToString(), Room::class.java)
emitter.onSuccess(room)
})
}
private fun createPresentationScope(room: Room, file: File) {
val user = room.viewers.find { it.socketId == socket.id() } ?:
room.presenter
val scope = Toothpick.openScopes(Scopes.APP_SCOPE,
Scopes.PRESENTATION_SCOPE)
scope.installModules(object : Module() {
init {
bind(Room::class.java).toInstance(room)
bind(User::class.java).toInstance(user)
bind(File::class.java).toInstance(file)
}
})
}
}
```