

Міністерство освіти і науки України
Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Технології створення програмних продуктів»

за темою

«Study Posters»

Частина №3

Виконав(ла):
студентка 3-го курсу
групи AI-182
Волошина А.Д.
Перевірив:
Бабич М. І.

Одеса-2020

Анотація

В курсовій роботі розглядається процес створення програмного продукту «Study Posters». Робота виконувалась в команді з декількох учасників: Толмаченко Яна Вадимівна, Грибков Дмитрій Іванович, Волошина Анастасія Дмитрівна. Тому в пояснювальній записці у розділах «Проектування» та «Конструювання» детальніше описано лише одну частину з урахуванням планів проведених робіт з розділу «Планування» з описом особливостей конструювання:

- структур даних моделі MVP в системі керування базами даних PostgreSQL;

- програмних модулів в інструментальному середовищі Android Studio з використанням мов програмування Kotlin та Java.

Результати роботи розміщено на *github*-репозиторії за адресою:
<https://github.com/Anastasiia-Voloshyna/Study-Posters>

Перелік скорочень

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП – програмний продукт

UML – уніфікована мова моделювання

Зміст

1	Вимоги до програмного продукту	7
1.1	Визначення потреб споживача	7
1.1.1	Ієрархія потреб споживача	7
1.1.2	Деталізація матеріальної потреби	8
1.2	Бізнес-вимоги до програмного продукту	9
1.2.1	Опис проблеми споживача	9
1.2.1.1	Концептуальний опис проблеми споживача	9
1.2.1.2	Метричний опис проблеми споживача	9
1.2.2	Мета створення програмного продукту	10
1.2.2.1	Проблемний аналіз існуючих програмних продуктів	10
1.2.2.2	Мета створення програмного продукту	11
1.2.3	Назва програмного продукту	11
1.2.3.1	Гасло програмного продукту	11
1.2.3.2	Логотип програмного продукту	11
1.3	Вимоги користувача до програмного продукту	12
1.3.1	Історія користувача програмного продукту	12
1.3.2	Діаграма прецедентів програмного продукту	13
1.3.3	Сценаріїв використання прецедентів програмного продукту	13
1.4	Функціональні вимоги до програмного продукту	17
1.4.1	Багаторівнева класифікація функціональних вимог	17
1.4.2	Функціональний аналіз існуючих програмних продуктів	18
1.5	Нефункціональні вимоги до програмного продукту	18
1.5.1	Опис зовнішніх інтерфейсів	18
1.5.1.1	Опис інтерфейса користувача	18
1.5.1.1.1	Опис INPUT-інтерфейса користувача	18
1.5.1.1.2	Опис OUTPUT-інтерфейса користувача	19

1.5.1.2	Опис інтерфейсу із зовнішніми пристроями	19
1.5.1.3	Опис програмних інтерфейсів	23
1.5.1.4	Опис інтерфейсів передачі інформації	23
1.5.1.5	Опис атрибутів продуктивності	23
2	Планування процесу розробки програмного продукту	24
2.1	Планування ітерацій розробки програмного продукту	24
2.2	Концептуальний опис архітектури програмного продукту	24
2.3	План розробки програмного продукту	25
2.3.1	Оцінка трудомісткості розробки програмного продукту	26
2.3.2	Визначення дерева робіт з розробки програмного продукту	29
2.3.3	Графік робіт з розробки програмного продукту	30
2.3.3.1	Таблиця з графіком робіт	30
3	Проектування програмного продукту	31
3.1	Концептуальне та логічне проектування структур даних програмного продукту	31
3.1.1	Концептуальне проектування на основі UML-діаграми концептуальних класів	31
3.1.2	Логічне проектування структур даних	32
3.2	Проектування програмних класів	33
3.3	Проектування алгоритмів роботи методів програмних класів	34
3.4	Проектування тестових наборів методів програмних класів	38
4	Конструювання програмного продукту	43
4.1	Особливості конструювання структур даних	43
4.1.1	Особливості інсталяції та роботи з СУБД	44
4.1.2	Особливості створення структур даних	44
4.2	Особливості конструювання програмних модулів	45
4.2.1	Особливості роботи з інтегрованим середовищем розробки	45
4.2.3	Особливості створення програмних класів	46

4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій	48
4.3 Модульне тестування програмних класів	51
5 Розгортання та валідація програмного продукту	56
5.1 Інструкція з встановлення програмного продукту	56
5.2 Інструкція з використання програмного продукту	56
5.3 Результати валідації програмного продукту	61
Висновки до курсової роботи	63

1 Вимоги до програмного продукту

1.1 Визначення потреб споживача

1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологічні;
- потреба в безпеці;
- приналежність;
- повага;
- пізнання;
- естетичні потреби;
- самоактуалізація.

Програмний продукт під назвою «Study Posters» має на меті задовольнити потребу користувача у пізнанні (надати актуальну інформацію про афіші) та самоактуалізація (можливість реалізувати себе за допомогою знайденої інформації). Ці рівні відносяться до ієрархії самовираження.



Рисунок. 1.1.1 – Ієрархія потреби споживача. Рівень потреби – самовираження

1.1.2 Деталізація матеріальної потреби

Для деталізації матеріальної потреби можна скористатися ментальними картами (MindMap). При створенні ментальних карт матеріальна потреба розташовується в центрі карти. Асоціативні гілки можна швидко створити, припускаючи, що в загальному вигляді з об'єктом пов'язані три потоки даних / інформації: вхідний, внутрішній, вихідний. Кожен потік - це асоціативна група, що включає можливі п'ять гілок, що відповідають на п'ять питань: Хто? Що? Де? Коли? Як?

Відповідно до рекомендацій по створенню ментальних карт кожна гілка-асоціація може бути розділена на додаткові асоціативні гілки, які деталізують відповіді на поставлені питання.

Потреба, яка була визначена при аналізі матеріальних проблем споживача, основні та додаткові асоціативні гілки зображені на Рис 1.2

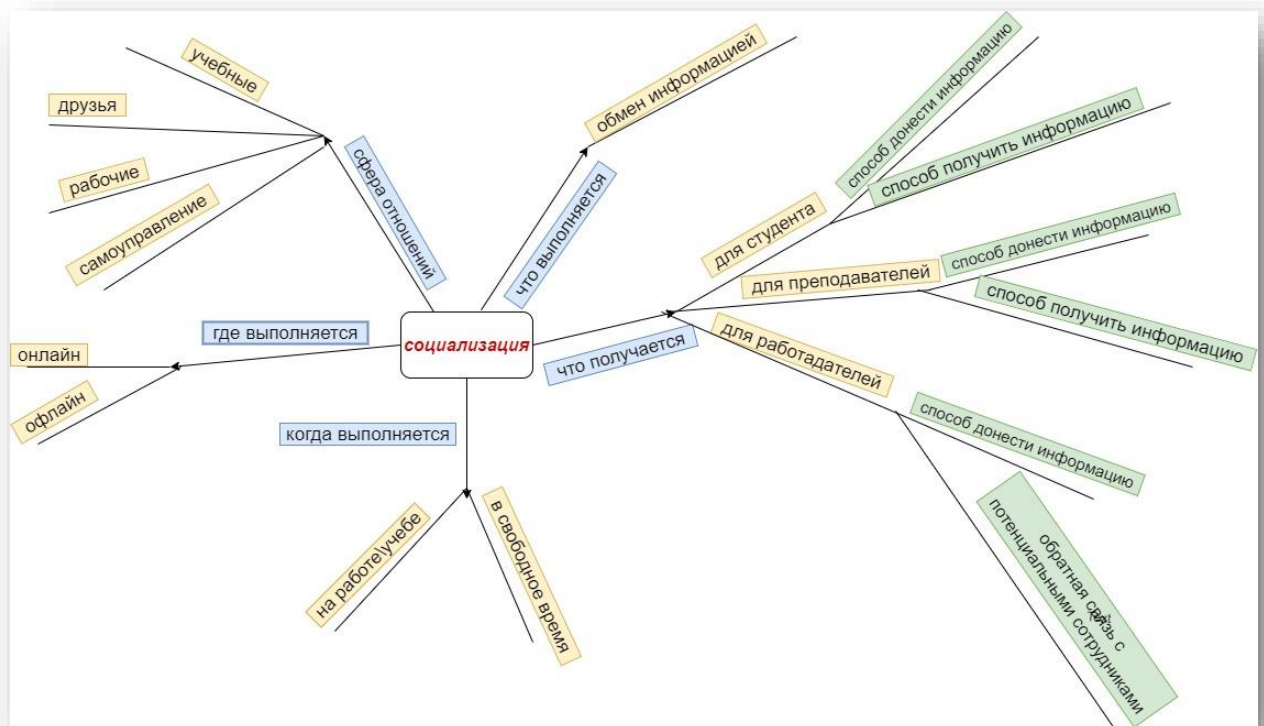


Рисунок - 1.1.2 – Деталізація матеріальної потреби

1.2 Бізнес-вимоги до програмного продукту

1.2.1 Опис проблеми споживача

1.2.1.1 Концептуальний опис проблеми споживача

Концептуальний опис проблеми споживача відображений у табл. 1.1

Таблиця – 1.2.1.1

№	Загальний опис проблеми
1	Складність доступу до інформації щодо новин та заходів
2	Відсутність її структурованості даних
3	Неможливість надійно зберігати новини щодо заходів університету

1.2.1.2 Метричний опис проблеми споживача

Метричний опис проблем споживача відображені у табл.1.2.1 – 1.2.5

Таблиця – 1.2.1.2.1

№	Загальний опис проблеми	Метричні показники незадоволеності споживача
1	Довгий пошук інформації про потрібний захід	Низький рівень інформативності існуючих джерел, можливість «загубити» необхідну інформацію.

Метричні показники визначаються рівнем доступності – AL (access level), який можна визначити як $AL = NA / N$;

де NA – кількість користувачів продукту, котрим треба інформація щодо існуючих заходів інституту;

N – загальна кількість користувачів, що зареєстровані в додатку.

$AL \rightarrow 0$

1.2.2 Мета створення програмного продукту

1.2.2.1 Проблемний аналіз існуючих програмних продуктів

Проблемний аналіз існуючих програмних продуктів відображена у табл.2.1

Таблиця 1.2.2.1

№	Назва продукту	Вартість	Ступінь готовності	Примітка
1	ори.ua	Безкоштовно	1	Недостатня кількість інформації щодо афіш
2	Telegram-канали	Безкоштовно	1	Неструктурована інформація
3	today.od.ua	Безкоштовно	1	Достатньо афіш, але вони не стосуються навчальних закладів
4	Instagram-акаунти	Безкоштовно	1	Багато інформації, що не стосується тематики

1.2.2.2 Мета створення програмного продукту

Підвищення інформативності студентів та викладачів з урахуванням інтересів кожного, структуризація афіш та загальної інформації щодо заходів що будуть проводитись у навчальному закладі.

1.2.3.1 Гасло програмного продукту

Гасло — лаконічна фраза, що впадає в око, добре запам'ятовується та висловлює суть продукту.

На початковому етапі розробки було вигадано гасло, яке найкращим чином відображає мету та суть роботи веб-сервісу та пов'язано з логотипом та назвою.

Гасло: «Study Posters – усі афіши в твоїй кишені».

1.2.3.2 Логотип програмного продукту

Відмінним способом представлення назви програмного продукту є його логотип, що поєднує зорові образи.

На рис 1.2.3.2 можна побачити розроблений командою логотип, який відображає назву веб-сервісу.



Рисунок - 1.2.3.2 - Логотип

1.3.1 Історія користувача програмного продукту

1. Як користувач, я можу авторизуватися в додатку.
2. Як користувач, я можу переглядати афіші, щоб знайти необхідні.
3. Як користувач, я можу шукати афіші за назвою, щоб вибрати потрібні.
4. Як користувач, я можу додавати афіші в обране, щоб не втратити їх.
5. Як користувач, я можу змінювати інформацію про себе в своєму профілі.
6. Як користувач, я можу змінювати колірні теми в додатку для себе.
7. Як адміністратор, я можу виконувати ті самі дії, що і користувач.
8. Як адміністратор, я можу додавати нові афіші.
9. Як адміністратор, я можу видаляти користувачів.

1.3.2 Діаграма прецедентів програмного продукту

Діаграма прецедентів (Use Case UML-діаграма) включає:

- актори (зацікавлені особи і зовнішні системи зі своїм API);
- прецеденти як основні функції ПП;
- зв'язки між прецедентами і акторами як множиною зацікавлених осіб;
- можливі зв'язки-узагальнення між акторами.

Діаграма прецедентів програмного продукту зображена на рис. 1.3.1

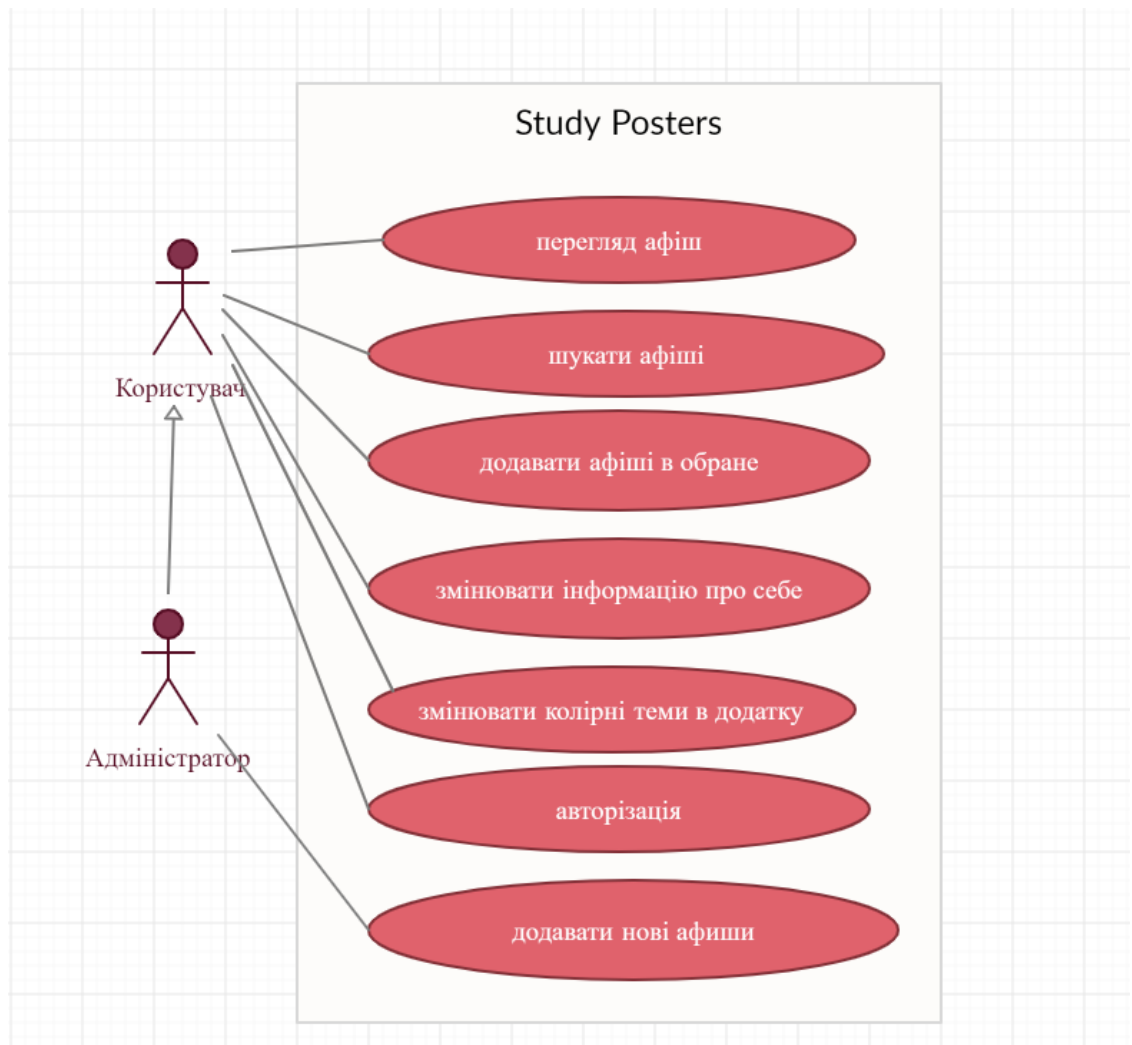


Рисунок - 1.3.2. Діаграма прецедентів програмного продукту

Зі сторони користувача програма повинна реалізувати можливість перегляду афіш, додавання афіш в обране, можливість шукати потрібні афіші по ключам, можливість редагування власну інформацію про користувача і можливість змінювати колір тем у додатку.

Зі сторони адміністратора повинні виконуватися ті ж умови, що й у користувача, але так само адміністратор повинен мати можливість додавання нових афіш.

1.3.3 Сценаріїв використання прецедентів програмного продукту

Для кожного прецедента описан сценарій використання з урахуванням пунктів:

- назва прецеденту;
- передумови початку виконання прецеденту;
- актори як зацікавлені особи у виконанні прецеденту;
- актор-основна зацікавлена особа як ініціатор початку прецеденту;
- гарантії успіху (що отримають актори у разі успішного завершення прецеденту);
- основний успішний сценарій;
- альтернативні сценарії, прив'язані до кроків основного успішного сценарію.

Основний успішний сценарій прецедента «авторизація»:

1. ПП запрошує в користувача параметри авторизації;
2. Користувач передає ПП свої параметри;
3. ПП надає користувачеві доступ до інших прецедентів.

Альтернативний сценарій основного успішного сценарія прецедента «авторизація»:

- 3.1 ПП виявляє, що користувач передав йому неправильні дані;
- 3.2 ПП видає повідомлення про помилку і переходить до кроку 1 основного успішного сценарія.

Основний успішний сценарій прецедента «додання афіш до списку обраного»:

Передумова: користувач має бути авторизованим та знаходитись на сторінці детальної інформації про дану афішу.

1. ПП показує користувачеві детальний опис обраної афіші;
2. Користувач натискає кнопку «додати до списку обраного»;
3. ПП додає дану афішу до списку обраного цього користувача.

Основний успішний сценарій прецедента «пошук афіш»:

Передумова: користувач має бути авторизованим та знаходитись на сторінці перегляду афіш.

1. ПП видає список афіш для перегляду;
2. Користувач натискає кнопку «пошук»;
3. ПП надає користувачеві доступ до пошукового рядку;
4. Користувач друкує якесь слово/хештег, які пов'язані з темою шуканої афіші;
5. ПП видає користувачеві список афіш, які мають слово, або хештег, які надрукував користувач;

Альтернативний сценарій основного успішного сценарія прецедента «пошук афіш»:

- 5.1 ПП виявляє, що жодна афіша не містить в собі такого слова/хештегу;
- 5.2 ПП видає повідомлення про помилку і пропонує користувачеві перехід до кроку 3 основного успішного сценарія.

Основний успішний сценарій прецедента «налаштування додатку»:

Передумова: користувач має бути авторизованим та знаходитись на сторінці свого аккаунту.

1. Користувач натискає кнопку «налаштування»;
2. ПП надає користувачеві можливість змінити інтерфейс програми(мова, тема) на даному пристрої та налаштувати оповіщення;
3. Користувач налаштовує під себе програму;
4. ПП зберігає зміни для даного пристрою.

Основний успішний сценарій прецедента «додавання нових афіш»:

Передумова: користувач має бути авторизованим та мати статус адміністратора.

1. Адміністратор натискає кнопку «додати афішу»;
2. ПП видає вікно створення нових афіш;
3. Адміністратор заповнює форму створення та натискає кнопку «створити»;
4. ПП надає можливість переглянути усі деталі афіші;
5. Якщо усі деталі афіші правильні, адміністратор натискає кнопку «так»;

ПП додає нову афішу;

Альтернативний сценарій основного успішного сценарія прецедента «додавання нових афіш»:

5.1 Якщо адміністратор бачить якусь помилку в деталях, він натискає кнопку «змінити»;

5.2 Перехід до кроку 3 основного успішного сценарія.

1.4 Функціональні вимоги до програмного продукту

1.4.1. Багаторівнева класифікація функціональних вимог

Таблиця – 1.4.1. Багаторівнева класифікація функціональних вимог

Ідентифікатор функції	Назва функції
FR 1	Авторизація
FR 1.1	Запитання у користувача параметрів авторизації та передача користувачем своїх параметрів з подальшим наданням доступу
FR 1.2	Отримання неправильних параметрів від користувача та виведення повідомлення про помилку
FR 1.3	Введення нових параметрів
FR 2	Вхід до акаунту
FR 3	Перегляд стрічки афіш
FR 4	Пошук афіш за ключем
FR 4.1	Неможливість відшукати афіши за введеним ключем
FR 5	Додавання потрібної афіши до обраних
FR 6	Додавання хештегу до обраних
FR 7	Зміна інформації акаунту
FR 8	Додавання адміністратором нових афіш
FR 8.1	Отримання неправильних даних від адміністратора з подальшим виведенням повідомлення про помилку

1.4.2 Функціональний аналіз існуючих продуктів

Таблиця 1.1 Функціональний аналіз існуючих продуктів

Ідентифікатор функції	telegram	opu.ua
FR 1.1	+	+
FR 1.2	+	+
FR 1.3	+	+
FR 2	+	+
FR 3	-	+
FR 4	+	-
FR 4.1	+	-
FR 5	-	-
FR 6	-	-
FR 7	+	+
FR 8	+	+
FR 8.1	+	+

1.5 Нефункціональні вимоги до програмного продукту

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейса користувача

1.5.1.1.1 Опис INPUT-інтерфейса користувача

В якості INPUT-інтерфейсу користувача використовується сенсорний екран (Touchscreen, Touchpad, Multi-touch). Особливості використання сенсорного екрану:

- використання сенсору для закінчення введення інформації;
- використання сенсору для підтвердження ознайомлення з помилкою;
- використання сенсору для входу в акаунт;
- використання сенсору для ознайомлення із стрічкою;
- використання сенсору для введення ключа;

- використання сенсору для підтвердження ознайомлення з помилкою;
- використання сенсору для додавання потрібної афіші до обраних;
- використання сенсору для додавання хештегу до обраних;
- використання сенсору для зміни інформації акаунту;
- використання сенсору для додавання нових афіш.

1.5.1.1.2 Опис OUTPUT-інтерфейсу користувача

В якості OUTPUT-інтерфейсу користувача використовуватиметься графічний екран (дисплей смартфона). Вся інформація виводитиметься на нього (афіші, сповіщення і так далі

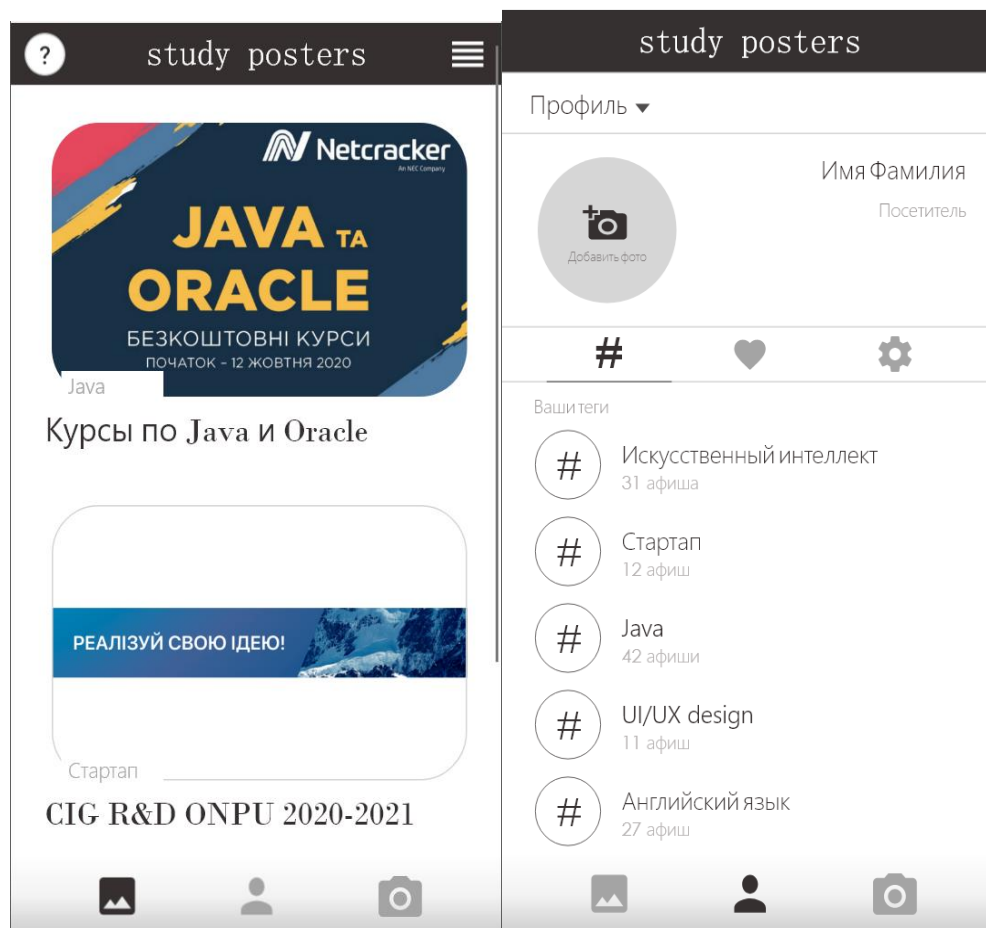


Рисунок 1.5.1.1.2.1 – Прототип та фінальний дизайн OUTPUT-інтерфейса користувача.

study posters

Регистрация | Вход

Профиль▼

ФИО

Ник

Пароль

Повторите пароль

Контактный телефон

Электронная почта

Зарегистрироваться

Отмена

Рисунок 1.5.1.1.2.2 – Прототип функції FR1

study posters

Вход | Регистрация

Профиль▼

Ник

Пароль

Войти

Отмена

Рисунок 1.5.1.1.2.4 – Прототип функції FR2



Курсы по Java и Oracle



CIG R&D ONPU 2020-2021



Творческий конкурс

Рисунок 1.5.1.1.2.4 – Прототип функції FR3

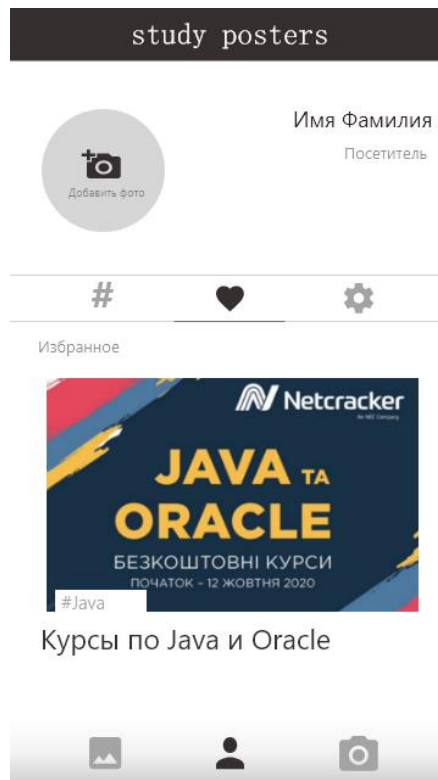


Рисунок – 1.5.1.1.2.10. Прототип функції FR5

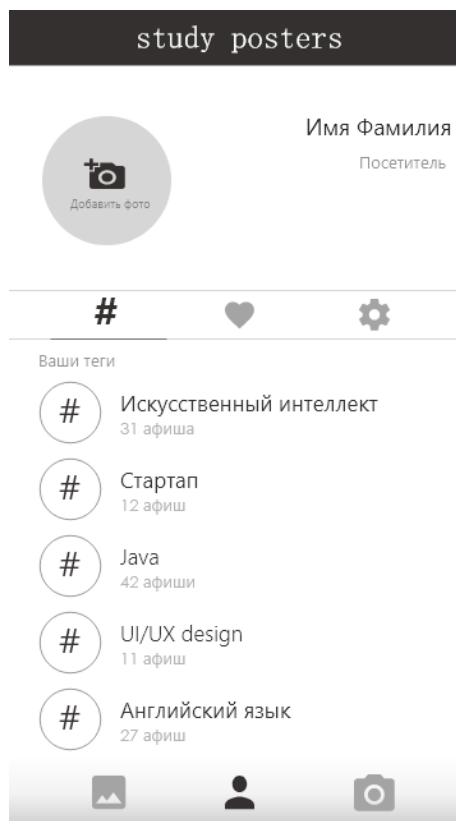


Рисунок – 1.5.1.1.2.10. Прототип функції FR6

1.5.1.3 Опис програмних інтерфейсів

В якості програмних інтерфейсів будуть використовуватися мови програмування Java (для Android з кінцевим розширенням .apk) Swift (для iOS з кінцевим розширенням .ipa), а також фреймворки Sencha Touch 2 та Dojo mobile.

1.5.1.4 Опис атрибутів продуктивності

Таблиця 1.1 Опис атрибутів продуктивності

Ідентифікатор функції	Максимальний час реакції ПП на дії користувачів, секунди
FR 1.1	3
FR 1.2	10
FR 1.3	6
FR 2	2
FR 3	2
FR 4	4
FR 4.1	2
FR 5	3
FR 6	3
FR 7	4
FR 8	10
FR 8.1	6

2 Планування процесу розробки програмного продукту

2.1 Планування ітерацій розробки програмного продукту

2.2 Концептуальний опис архітектури програмного продукту

Концептуальний опис архітектури програмного продукту зображено на рис. 2.2

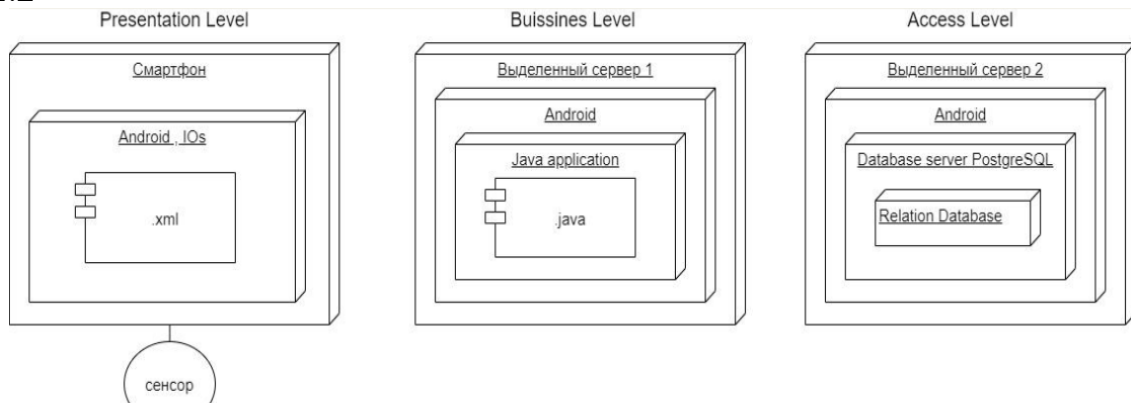


Рисунок – 2.2. Концептуальний опис архітектури програмного продукту

2.3 План розробки програмного продукту

З метою забезпечення для вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, було визначено функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП.

При створенні пріоритетів необхідно врахувати:

- сценарні залежності між прецедентами, до яких належать функції, на основі аналізу пунктів передумов початку роботи прецедентів, вказаних в описі сценаріїв роботи прецедентів;

- вплив роботи прецеденту, до якого належить функція, на досягнення мети ПП, наприклад у відсотках, на основі аналізу пунктів гарантій успіху, вказаних в описі сценаріїв роботи прецедентів.

Сценарні залежності будуть перетворені у відповідні функціональні

залежності. Вплив роботи прецеденту буде поширено на всі підлеглі функції ієрархії. При визначенні пріоритетів рекомендується використовувати наступні позначки:

- М (Must) – функція повинна бути реалізованою у перших ітераціях за будь-яких обставин;
- S (Should) – функція повинна бути реалізованою у перших ітераціях, якщо це взагалі можливо;
- C (Could) – функція може бути реалізованою, якщо це не вплине негативно на строки розробки;
- W (Want) – функція може бути реалізованою у наступних ітераціях.

Опису представлено в таблиці 2.3

Таблиця 2.3 – Опис функціональних пріоритетів

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет функції
FR1	-	0	M
FR1.1	-	0	M
FR1.2	-	0	M
FR1.3	-	0	M
FR2	-	0	M
FR3	-	10	S
FR4	-	10	M
FR4.1	FR4	0	M
FR5	FR4	75	M
FR6	-	5	W

FR7	-	0	W
FR8	-	0	M
FR8.1	-	0	M

2.3.1 Оцінка трудомісткості розробки програмного продукту

Визначення нескорегованого показника UUCP (Unadjusted Use Case Points)

Всі актори діляться на три типи: прості, середні і складні.

Простий актор представляє зовнішню систему з чітко визначеним програмним інтерфейсом.

Середній актор представляє або зовнішню систему, що взаємодіє з ПП за допомогою мережевих протоколів, або особистість, що користується текстовим інтерфейсом (наприклад, алфавітно-цифровим терміналом).

Складний актор представляє особистість, що користується графічним інтерфейсом. Загальна кількість акторів кожного типу помножується на відповідний ваговий коефіцієнт, потім обчислюється загальний ваговий показник (табл.2.3.1.1)

Таблиця - 2.3.1.1.Вагові коефіцієнти акторів

Тип актора	Ваговий коефіцієнт	Кількість
Простий	1	3
Середній	2	0
Складний	3	0

Тип прецедента	Кількість кроків сценарія	Ваговий коефіцієнт	Кількість
Простий	<4	5	5
Середній	4-7	10	2

Складний	>7	15	1
----------	----	----	---

Визначення UUCP

$$A = 3 \quad UC = 70 \quad UUCP = A + UC = 73$$

Технічна складність проекту (TCF - Technical Complexity Factor)

обчислюється з урахуванням показників технічної складності.

Кожному показнику присвоюється значення STi в діапазоні від 0 до 5:

- 1) 0 означає відсутність значимості показника для даного проекту;
- 2) 5 - високу значимість.

Показники технічної складності проекту TCF представлено в таблиці

2.3.1.2

Таблиця – 2.3.1.2. Показники технічної складності проекту

Показник	Опис показника	Вага	ST
T1	Распределенная система	2	2
T2	Высокая производительность (пропускная способность)	1	3
T3	Работа конечных пользователей в режиме онлайн	1	1
T4	Сложная обработка данных	-1	2
T5	Повторное использование кода	1	2
T6	Простота установки	0,5	0
T7	Простота использования	0,5	1
T8	Переносимость	2	0
T9	Простота внесения изменений	1	2
T10	Параллелизм	1	2
T11	Специальные требования к безопасности	1	2
T12	Непосредственный доступ к системе со стороны внешних пользователей	1	0

T13	Специальные требования к обучению пользователей	1	1
-----	---	---	---

TCF(Technical Complexity Factor) = 0,67

Рівень кваліфікації розробників (EF - Environmental Factor) обчислюється з урахуванням наступних показників (табл. 2.3.1.3).

Таблица – 2.3.1.3. Показники рівня кваліфікації розробників

Показник	Опис	Вага
F1	Знайомство з технологією	1
F2	Досвід розробки додатків	3
F3	Досвід застосування об'єктно-орієнтованого підходу	3
F4	Наявність ведучого аналітика	1
F5	Мотивація	5
F6	Стабільність вимог	1
F7	Часткова зайнятість	2
F8	Складні мови програмування	2

EF (Environmental Factor) = 0,94

Остаточне значення UCP (Use Case Points)

UUCP(Unadjusted Use Case Points) = 94

2.3.3 Графік робіт з розробки програмного продукту

2.3.3.1 Таблиця з графіком робіт

Для кожної підзадачі визначається виконавець, що фіксується у вигляді таблиці, яка представлена в таблиці 2.3.3.1

Таблиця – 2.3.3.1

WST	Дата початку	Дні	Дата завершення	Виконавець
WST 1.1.1	1. 10. 2020	5	5. 10. 2020	Толмаченко Я.В.
WST 1.1.2	5. 10. 2020	5	10. 10. 2020	Толмаченко Я.В.
WST 1.1.3	10. 10. 2020	2	12. 10. 2020	Толмаченко Я.В.
WST 1.2.1	12. 10. 2020	5	17. 10. 2020	Грибков Д.И.
WST 1.2.2	17. 10. 2020	3	20. 10. 2020	Грибков Д.И.
WST 1.2.3	20. 10. 2020	5	25. 10. 2020	Грибков Д.И.
WST 1.3.1	25. 10. 2020	2	27. 10. 2020	Волошина А.Д.
WST 1.3.2	27. 10. 2020	3	30. 10. 2020	Волошина А.Д.
WST 1.3.3	30. 10. 2020	3	2. 11. 2020	Волошина А.Д.

3 Проектування програмного продукту

3.1 Концептуальне та логічне проектування структур даних програмного продукту

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

Використовуючи кроки основного успішного та альтернативного сценаріїв роботи прецедентів ПП, було спроектовано UML-діаграми концептуальних класів. Моделювання проведено з урахуванням наступної послідовності кроків.

1. Визначити імена класів.
2. Визначити імена атрибутів класів.
3. Визначити зв'язку між класами (узагальнення, іменовані асоціації, агреговані асоціації).
4. Для зв'язків-асоціацій визначити назви, кратність і можливість агрегації.
5. Створити UML-діаграму класів в будь-якому графічному редакторі.

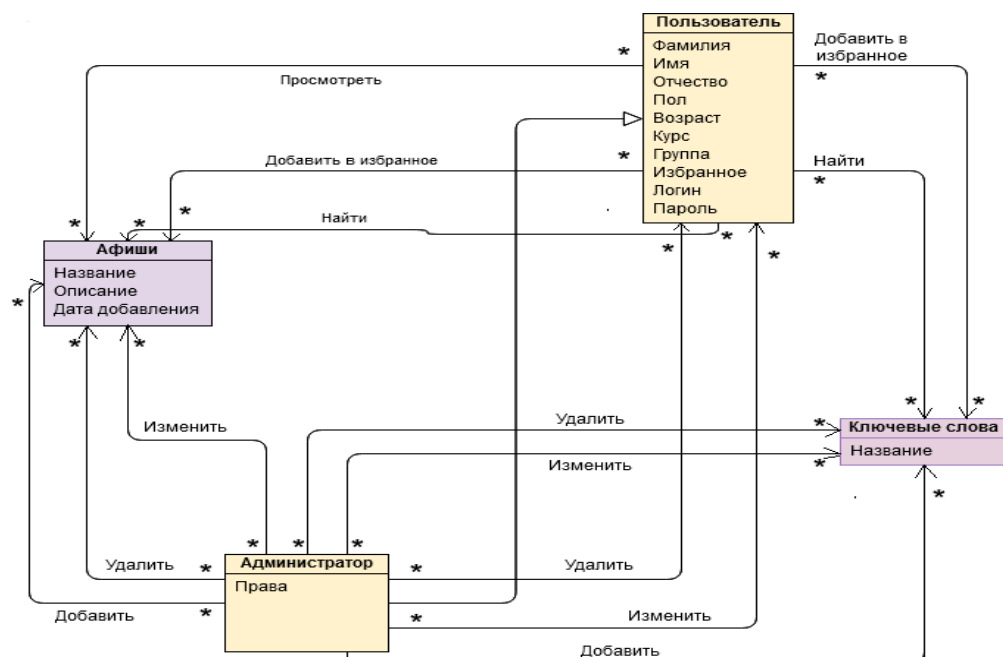


Рисунок - 3.1.1. UML-діаграма класів

3.1.2 Логічне проектування структур даних

UML-діаграма концептуальних класів була перетворена в опис структур даних з використанням моделі, яка була обрана в концептуальному описі архітектури ПП,

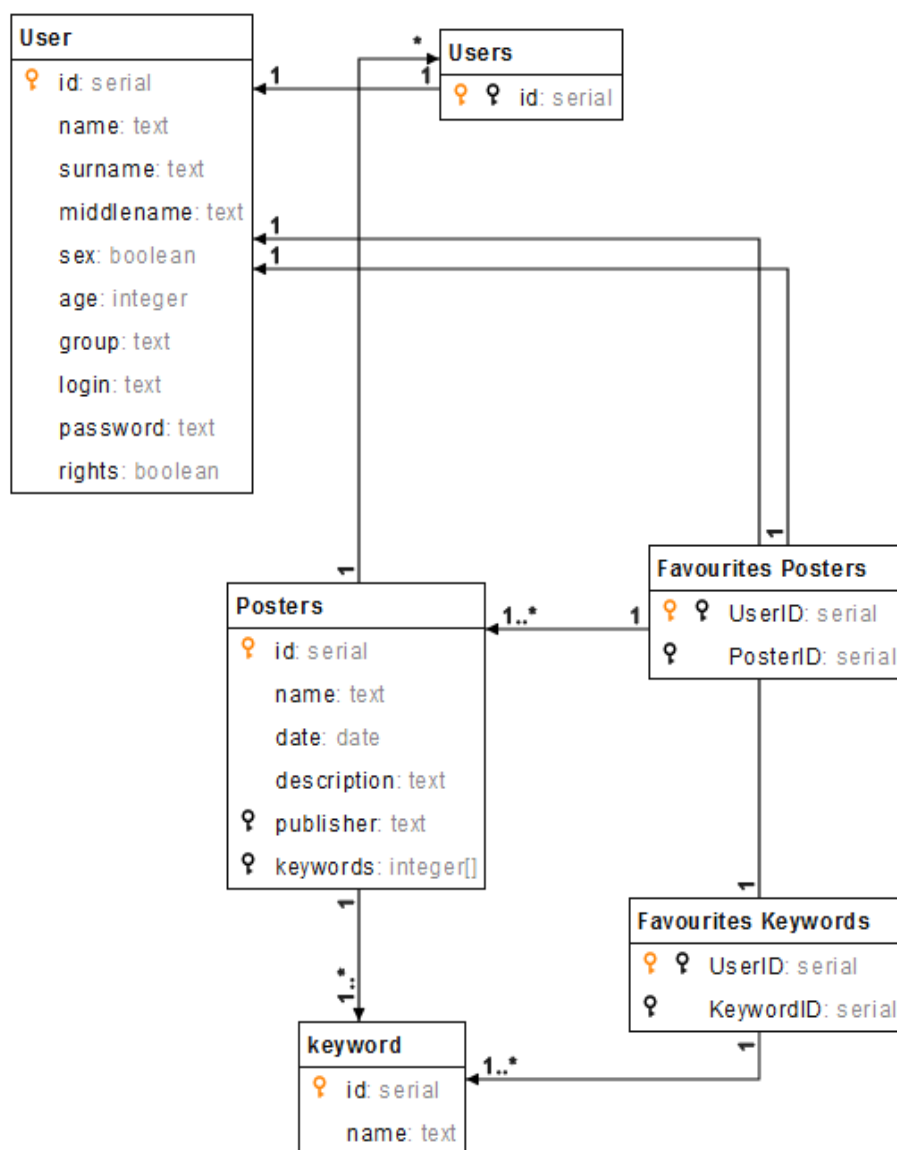
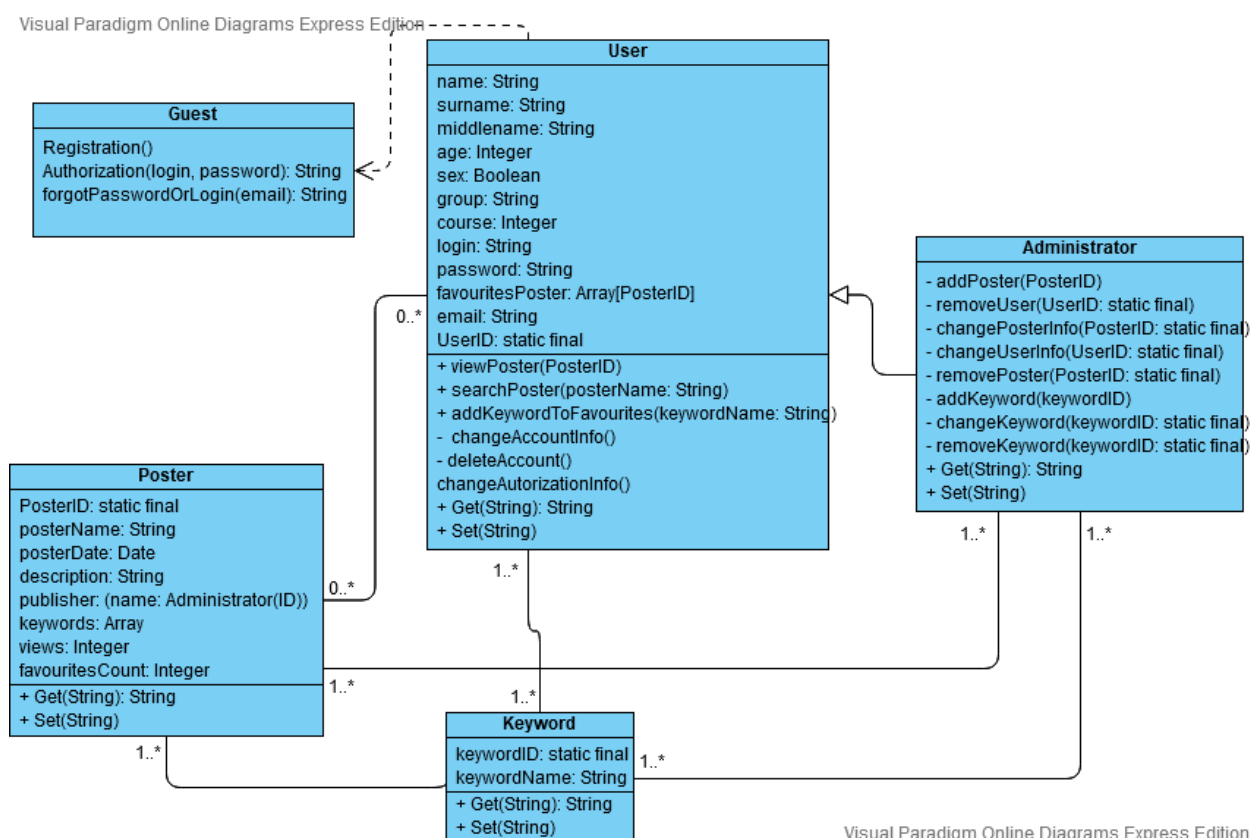


Рисунок – 3.1.2. Схема БД

3.2 Проектування програмних класів

1. На основі UML-діаграми концептуальних класів були спроектовані програмні класи:

- англійські назви класів та їх атрибутів;
- абстрактні класи, їх класи-нащадки та інші класи;
- зв'язки між класами (наслідування, іменована асоціація, агрегатна асоціація, або агрегація, композитна асоціація або композиція) та їх кратності;
- атрибути класів с типами даних (цілий, дійсний, логічний, перелічуваний, символьний з урахуванням розміру), та типом видимості (публічний, захищений, приватний);
- методи-конструктори ініціалізації екземплярів об'єктів класу, set-методи та get-методи для доступу до атрибутів класу



3.3 Проектування алгоритмів роботи методів програмних класів

Процес розробки алгоритмів:

1. З усіх методів виділили ті, що мають операції доступу до БД або містять керуючі умови (if-then-else, while).

2. Опишіть алгоритм роботи у вигляді UML-діаграми активності:

– кожний опис алгоритму представили в текстовому файлі на мові PlantUML, використовуючи веб-редактор.

Алгоритм роботи методу «Authorization» класу «User» мовою PlantUML:

```
@startuml
```

```
start
```

```
repeat
```

```
repeat
```

```
:Создание запроса у пользователя на получение его параметров аутентификации;
```

```
repeat while(Запрос не подтвержден);
```

```
:Передача от пользователя его параметров аутентификации;
```

```
:Сравнение параметров аутентификации пользователя с ранее сохранёнными в БД параметрами
```

```
note right
```

```
SELECT username FROM user
```

```
HAVING password = пароль, введённый пользователем при аутентификации
```

```
end note
```

```
backward: Информирование пользователя об ошибке;
```

```
repeat while(Неверный логин или пароль!)
```

:Получение параметров аутентификации пользователя;
:Информирование об успешной авторизации;
stop
@enduml

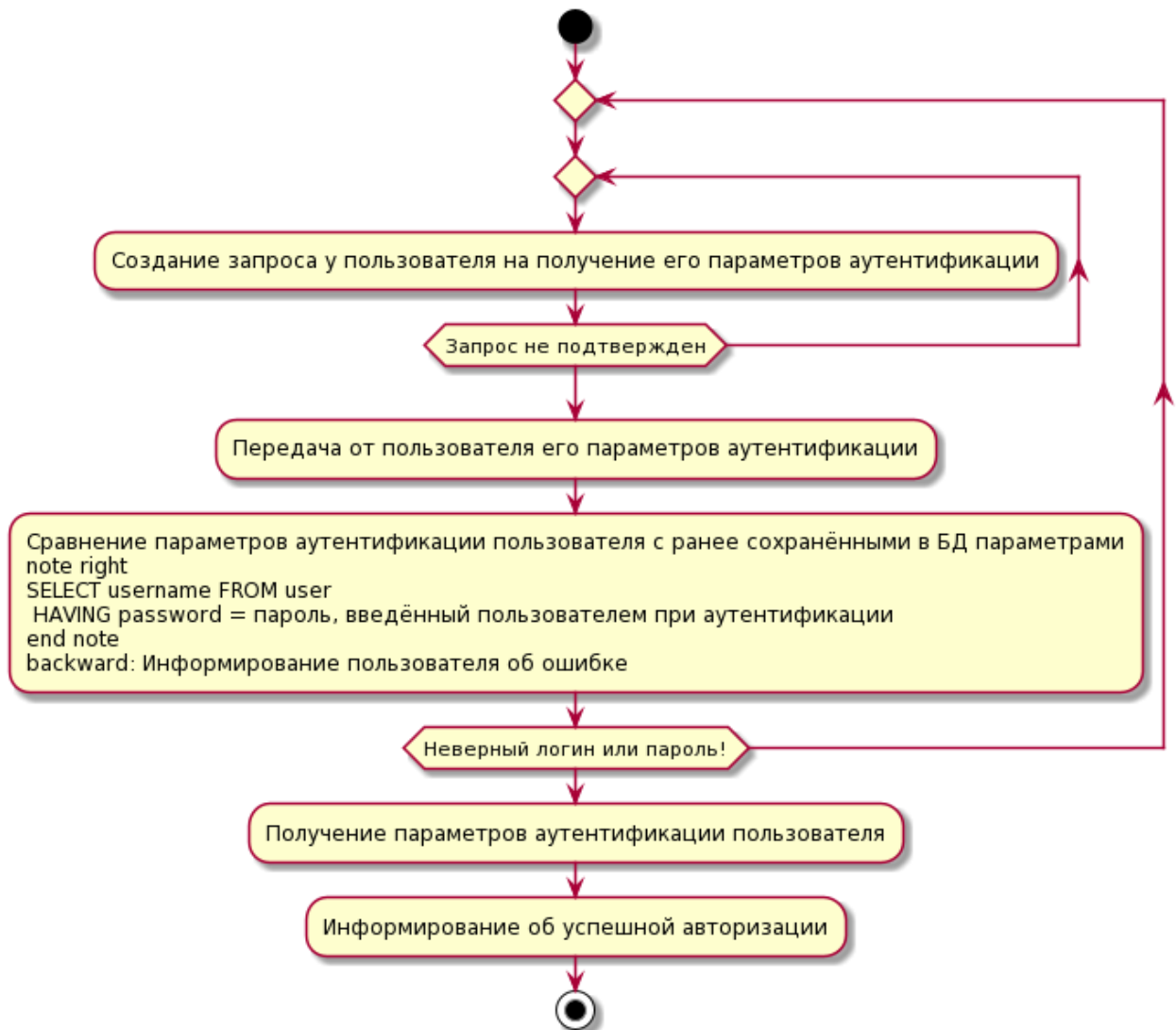


Рисунок 3.3.1 – Алгоритм методу «Authorization»

Алгоритм работы методу «searchPoster» мовою PlantUML:

```

@startuml
title searchPoster()

(*) --> "Enter parameters of poster"
  
```

```

    note right
    Select * from Poster exist Poster"
    end note
    --> if "Poster is exist"
    -->[Success] "Choose one Event of existed Poster"
    --> "Display success choosing"
    --> "Choose output Posters"
    --> (*)
    else
    --> [Failure] "Display Poster not found"
    endif
    --> (*)
    @enduml

```

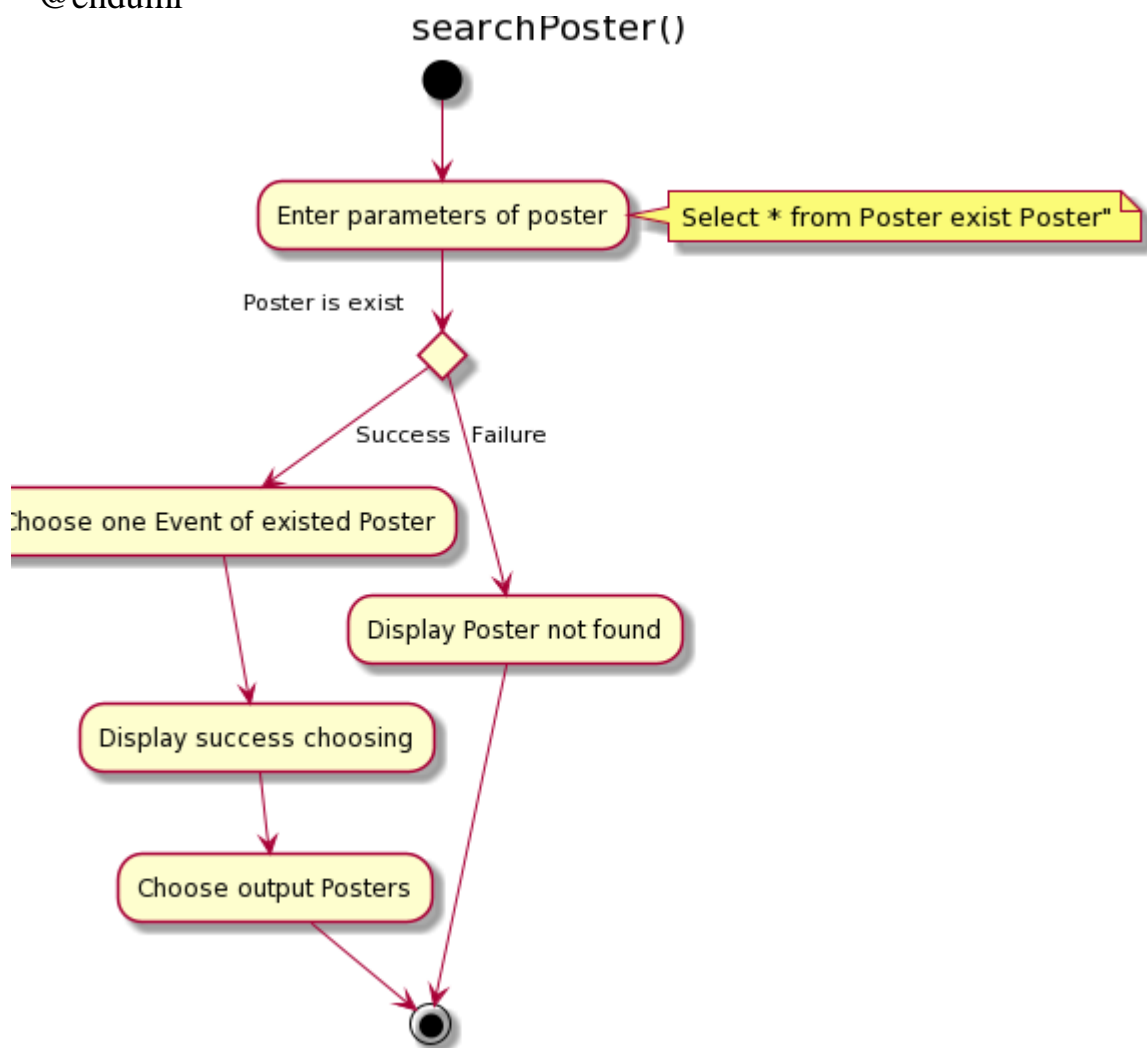


Рисунок 3.3.2 – Алгоритм методу «searchPoster»

Алгоритм роботи методу «deleteAccount» мовою PlantUML:

```
@startuml
```

```

title deleteAccount()
(*) --> "Delete Account"
note right
Select* from User WHERE userID
end note
end note
--> if "Are you sure?"
note right
Delete FROM Users where userID
end note
--> [Success] "Удалить аккаунт"
--> (*)
else
--> [Failure] "Аккаунт не удален."
endif
--> "Возвращение на главную страницу аккаунта"

```

@enduml

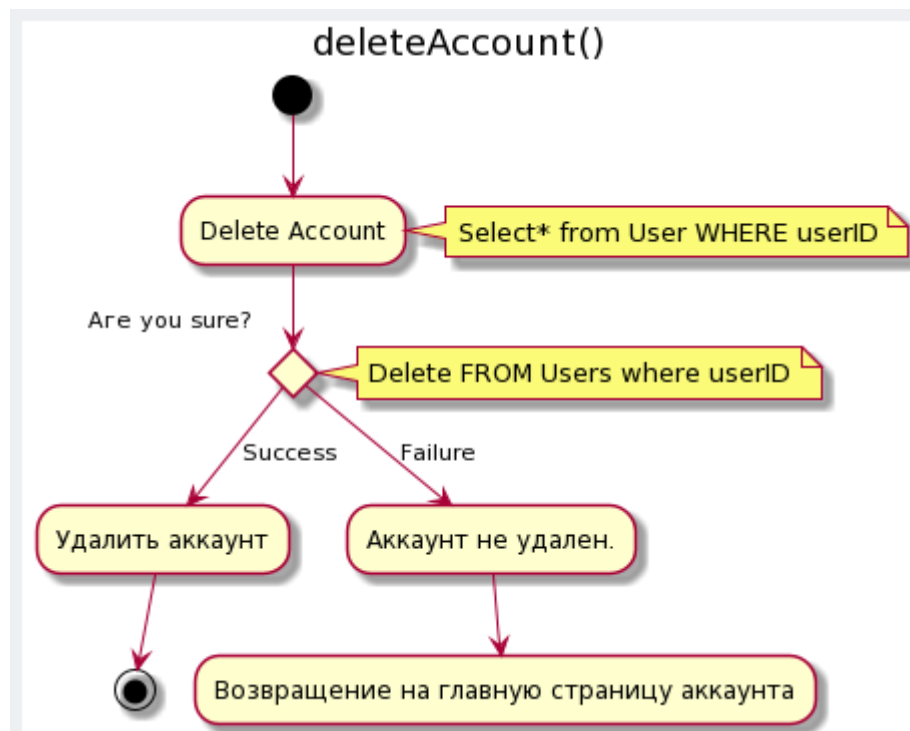


Рисунок 3.3.3 – Алгоритм метода «deleteAccount»

3.4 Проектування тестових наборів методів програмних класів

Назва функції	№ тесту	Опис значень вхідних даних	Опис очікуваних значень результату
Registration	1	Гость вводит логин, пароль, ещё раз пароль для его подтверждения и адрес электронной почты для регистрации	<p>Правильный класс: в системе появляется новый пользователь, которому присваивается ID.</p> <p>Неправильный класс: логин уже существует в системе. Выводится сообщение об ошибке.</p> <p>Неправильный класс: логин превышает допустимую длину в 50 символов. Выводится сообщение об ошибке.</p> <p>Неправильный класс: логин не соответствует нормам ввода. Выводится сообщение об ошибке.</p> <p>Неправильный класс: в системе уже есть пользователь с такой электронной почтой. Выводится сообщение об ошибке.</p> <p>Неправильный класс: в системе уже есть пользователь с таким номером телефона. Выводится сообщение об ошибке.</p>
Authorization	2	Гость вводит логин и пароль для входа в систему	<p>Правильный класс: успешный вход в систему как пользователь.</p> <p>Неправильный класс: логин не совпадает.</p>

			Выводится сообщение об ошибке. Неправильный класс: пароль не совпадает. Выводится сообщение об ошибке.
forgotPasswordOrLogin	3	Гость вводит адрес электронной почты, которая использовалась во время регистрации и выбирает, что хочет сбросить	Правильный класс: на указанный адрес электронной почты приходит ссылка, при переходе на которую можно сбросить логин или пароль и установить новый. Неправильный класс: нет пользователей с такой электронной почтой. Выводится сообщение об ошибке.
viewPoster	4	Пользователь или администратор нажимает на кнопку «Посмотреть детально» под постером	Переход на страницу детального описания постера
searchPoster	5	Имя постера или ключевое слово	Правильный класс: постеры, которые соответствуют введенному имени или у которых есть введенное ключевое слово. Неправильный класс: нет постера с таким именем или ключевым словом. Выводится сообщение, что постеров с таким именем или ключевым словом не существует.
addKeywordToFavourites	6	Ключевое слово	Ключевое слово добавляется в Избранные и можно быстро совершить поиск при нажатии на него.

changeAccountInfo	7	Пользователь нажимает на кнопку «Изменить данные аккаунта».	Данные аккаунта, которые пользователь изменил, изменяются в системе.
deleteAccount	8	Пользователь нажимает на кнопку «Удалить аккаунт»	Аккаунт пользователя удаляется из системы.
changeAuthorizationInfo	9	Пользователь нажимает на кнопку «Сменить логин» или «Сменить пароль»	<p>Правильный класс: логин или пароль пользователя меняются на введенные им.</p> <p>Неправильный класс: логин уже существует. Выводится сообщение об ошибке.</p> <p>Неправильный класс: логин превышает допустимую длину в 50 символов. Выводится сообщение об ошибке.</p> <p>Неправильный класс: логин не соответствует нормам ввода. Выводится сообщение об ошибке.</p>
addPoster	10	Необходимая информация про постер	<p>Правильный класс: постер добавляется и его можно посмотреть.</p> <p>Неправильный класс: имя постера превышает допустимую длину в 100 символов. Выводится сообщение об ошибке.</p> <p>Неправильный класс: ключевого слова не существует. Выводится сообщение об ошибке.</p>
removeUser	11	Администратор нажимает на кнопку «Удалить аккаунт пользователя» и вводит его ID	Аккаунт пользователя удаляется из системы

changePosterInfo	12	Администратор нажимает на кнопку «Изменить данные постера», вводит его ID и изменяет его атрибуты	Правильный класс: атрибуты постера изменяются. Неправильный класс: имя постера превышает допустимую длину в 100 символов. Выводится сообщение об ошибке. Неправильный класс: ключевого слова не существует. Выводится сообщение об ошибке.
changeUserInfo	13	Администратор нажимает на кнопку «Изменить данные аккаунта пользователя» и вводит ID пользователя, затем изменяет его данные	Правильный класс: данные аккаунта пользователя изменяются. Неправильный класс: указанная электронная почта уже закреплена за другим пользователем. Выводится сообщение об ошибке. Неправильный класс: указанный номер телефона уже закреплён за другим пользователем. Выводится сообщение об ошибке.
removePoster	14	Администратор нажимает на кнопку «Удалить постер» и вводит его ID	Постер удаляется из системы
addKeyword	15	Администратор нажимает на кнопку «Добавить ключевое слово» и вводит его	Правильный класс: ключевое слово добавляется в систему. Неправильный класс: ключевое слово превышает максимальную длину в 100 символов. Выводится сообщение об ошибке.
changeKeyword	16	Администратор нажимает на кнопку «Изменить	Правильный класс: ключевое слово изменяется.

		ключевое слово» и изменяет его	Неправильный класс: ключевое слово превышает максимальную длину в 100 символов. Выводится сообщение об ошибке.
removeKeyw ord	17	Администратор нажимает на кнопку «Удалить ключевое слово» и вводит его	Ключевое слово удаляется из системы
addPosterToF avourites	18	Пользователь нажимает на кнопку «Добавить в Избранное»	Постер добавляется в избранное

4 Конструювання програмного продукту

4.1 Особливості конструювання структур даних

Архітектура програмного забезпечення – це представлення системи програмного забезпечення, яке дає інформацію про складові компоненти системи, про взаємозв'язки між цими компонентами і правила, що регламентують ці взаємозв'язки. Це процес, що передбачає послідовність дій для створення або зміни архітектури системи, і проекту системи по цій архітектурі, з врахуванням безлічі обмежень.

4.1.2 Особливості створення структур даних

До створеної бази даних були створені тригери та функції для маніпулювання даними, занесена демонстраційна інформація. Також були розроблені запити для демонстрації можливостей створеної бази даних. Архітектура бази даних передбачає підключення її до комп'ютерного додатку чи веб-сайту через використання користувачів. Така система дозволяє обмежувати привілеї для захисту даних.

Розглянемо SQL-запити для створення таблиць:

1. Таблиця “Користувач”

```
CREATE SEQUENCE s_user;
```

```
CREATE TABLE user
```

```
(id_user INT PRIMARY KEY DEFAULT NEXTVAL('s_user'),
```

```
name VARCHAR,
```

```
surname VARCHAR,
```

```
middlename VARCHAR,
```

```
group VARCHAR,
```

```

age INT,

sex BOOLEAN,

login VARCHAR,

password VARCHAR,

rights BOOLEAN );

```

2. Таблиця “Афіша”

```

CREATE TABLE posters

(id_poster INT PRIMARY KEY DEFAULT NEXTVAL('s_user'),

name VARCHAR,

date DATE,

description VARCHAR,

publisher VARCHAR,

id_keyword INT REFERENCES keyword(id),

id_user INT REFERENCES user(id) ));

```

3. Таблиця “Кейворд”

```

CREATE TABLE keyword

(id_keyword INT PRIMARY KEY DEFAULT NEXTVAL('s_keyword'),

name VARCHAR,

);

```

4. Таблиця «Обрані кейворд»

```

CREATE TABLE favposter

(

id_poster INT,

id_user INT,

PRIMARY KEY (id_user,id_poster)

);

```

5. Таблиця «Обрані афіши»

```
CREATE TABLE favkeyword  
(  
    id_user INT,  
    id_keyword INT,  
    PRIMARY KEY (id_user, id_keyword)  
);
```

Для зберігання первинного ключа події використовується тип даних INT, бо операції порівняння та пошуку швидше за все працюють з цифровими типами.

База даних складається з наступних таблиць: Користувачі, Афіша, Кейворд, Обрані афіши, Обрані кейворд.

Таблиця «Користувач» містить інформацію про усіх користувачів системи. У ній присутні такі поля, як унікальний ідентифікатор (первинний ключ), ім'я, прізвище, по батькові, групу, вік, стать, пароль та логін для входу у систему, права (звичайний користувач чи адміністратор).

Таблиця «Афіша» містить інформацію про афіши. У ній присутні такі поля, як унікальний ідентифікатор події, назва афіши, її дата та час створення.

Таблиця «Обрані кейворд» містить інформацію про усі ключові слова, що були обрані користувачем. Це також таблиця, яка зв'язує таблиці «Кейворд» и «Користувач» у зв'язок багато-до-багатьох. У ній присутні такі поля, як унікальні ідентифікатори (первинні ключі) таблиць «Кейворд» и «Користувач»

Таблиця «Обрані афіши» містить інформацію про усі афіші, що були обрані користувачем. Це також таблиця, яка зв'язує таблиці «Афіші» и «Користувач» у зв'язок багато-до-багатьох. У ній присутні такі поля, як унікальні ідентифікатори (первинні ключі) таблиць «Афіші» и «Користувач».

4.2 Особливості конструювання програмних модулів

4.2.1 Особливості роботи з інтегрованим середовищем розробки

Інформаційна система «Study Posters» представляє собою андроїд-додаток. Розробка системи виконується в Android Studio. WebStorm знаходить застосування всіх можливостей сучасної андроїд-розробки. Включає в себе розумне автодоповнення коду, перевірку помилок на льоту, швидку навігацію по коду і рефакторинг для Java та Kotlin.

4.2.2 Особливості створення програмної структури

Для реалізації системи була обрана мови об'єктно-орієнтованого програмування Kotlin та Java. Наведемо як приклад клас створення створення користувача на сайті. Використовуємо мову програмування Kotlin.

```
override fun onScroll(scrollPosition: Float, currentPosition: Int, newPosition: Int,
currentHolder: RecyclerView.ViewHolder?, newCurrent: RecyclerView.ViewHolder?) {
    if (Math.abs(scrollPosition) > 0.5) {
        observer.onNext(newPosition)
    } else {
        observer.onNext(currentPosition)
    }
}

override fun onDispose() {
    view.removeScrollListener(this)
}
}

class RangeChangedObservable(private val view: RangeBar) : Observable<Pair<Int, Int>>() {

    override fun subscribeActual(observer: Observer<in Pair<Int, Int>>) {
        val listener = Listener(view, observer)
        observer.onSubscribe(listener)
        view.setOnRangeBarChangeListener(listener)
    }
}
```

Створення сторінок у андроїд-додатку відбувається за допомогою мови розмітки XML. Приклад використання XML-файлів:

Файл activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v7.widget.Toolbar
        android:id="@+id/main_toolbar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:text="@string/app_name"/>

        <ImageView
            android:id="@+id/main_filterIcon"
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="right|center_vertical"
            android:layout_marginRight="16dp"
            android:src="@drawable/ic_filter"/>

    </android.support.v7.widget.Toolbar>

    <fragment
        android:id="@+id/fragment"
        android:name="com.sumera.argallery.ui.feature.picturelist.PictureListFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/main_tabs"/>

    <android.support.design.widget.TabLayout
        android:id="@+id/main_tabs"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent">
```

```
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/main_toolbar">
```

```
<android.support.design.widget.TabItem
    android:id="@+id/main_allTab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/main_all_tab"/>
```

```
<android.support.design.widget.TabItem
    android:id="@+id/main_favouriteTab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/main_favourites_tab"/>
```

```
<android.support.design.widget.TabItem
    android:id="@+id/main_filteredTab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/main_filter_tab"/>
```

```
</android.support.design.widget.TabLayout>
```

```
<FrameLayout
    android:id="@+id/main_filterTabContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
</android.support.constraint.ConstraintLayout>
```

4.2.4. Особливості розробки алгоритмів методів програмних класів або процедур/функцій

У програмному середовищі були розроблені алгоритми методів програмних класів. Як приклад, наведемо алгоритм методу авторизації.

```
import android.app.Dialog
import android.content.Intent
import android.databinding.DataBindingUtil
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.widget.ListView
import android.widget.Toast
import com.android.volley.toolbox.Volley
import com.auth0.android.Auth0
```



```

import com.auth0.android.authentication.AuthenticationException
import com.auth0.android.provider.AuthCallback
import com.auth0.android.provider.WebAuthProvider
import com.auth0.android.result.Credentials
import com.auth0.samples.kotlinapp.databinding.ActivityMainBinding

class Authorization : Guest() {

    var binding: ActivityMainBinding? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        val queue = Volley.newRequestQueue(this)

        binding = DataBindingUtil.setContentView(this, R.layout.activity_main)
        binding?.loggedIn = false

        val listToDo = findViewById(R.id.list_todo) as ListView

        val loginButton = findViewById(R.id.login_button)
        loginButton.setOnClickListener { Authorization() }
    }

    override fun onNewIntent(intent: Intent) {
        if (WebAuthProvider.resume(intent)) {
            return
        }
        super.onNewIntent(intent)
    }

    private fun Authorization() {
        val account = Auth0(getString(R.string.auth0_client_id),
        getString(R.string.auth0_domain))
        account.isOIDCConformant = true

        WebAuthProvider.init(account)
            .withScheme("demo")
            .withAudience("StudyPosters")
            .start(this, object : AuthCallback {
                override fun onFailure(dialog: Dialog) {
                    runOnUiThread { dialog.show() }
                }

                override fun onFailure(exception: AuthenticationException) {
                    runOnUiThread {
                        Toast.AuthorizationPassword(
                            regexCheck = ^{?.{1,50}}
                            if (!regexCheck && password.length > 50)
                                this@MainActivity, "Пароль не совпадает!"
                                Toast.LENGTH_SHORT).show()
                        Toast.AuthorizationEmail(
                            if (!requireNotNull(nickname).exist()){
                                this@MainActivity, "Пользователя с таким
никон не существует!"
                            }
                    }
                }
            })
    }
}

```

```

        }

        override fun onSuccess(credentials: Credentials) {
            CredentialsManager.saveCredentials(this@MainActivity,
credentials)
            binding?.loggedIn = true
        }
    })
}
}

```

4.2.5 Особливості використання спеціалізованих бібліотек та API

Для спрощення розробки програмного продукту було використано готове API календаря – React Calendar. Оскільки API має відкритий вихідний код, календар було кастомізовано та налагоджено відповідно до потреб програмного продукту.

RxKotlin - бібліотека до Kotlin, яка може допомогти вам створювати високореактивні додатки в меншій кількості кодів.

Android KTX - набір розширень Kotlin для розробки додатків під Android. Додає переваги в розробці: менше коду, більше задоволення і полегшення розуміння коду проекту.

4.3 Тестування програмних модулів

4.3.1 Тестування методу AddUser()

Guest.Registration() представляє собою метод, що перевіряє введені користувачем дані під час реєстрації та проводить їх валідацію. Реалізація цього методу:

```

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

```

```

import android.widget.Toast;

public class Registration : Guest() {

    EditText firstLastName;
    EditText email;
    EditText password;
    EditText confirmPassword;
    EditText phone;
    EditText nickname;
    Button register;
    Button cancel;
    val fields = listOf(EditText);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        firstLastName = findViewById(R.id.firstLastName);
        nickname = findViewById(R.id.address);
        email = findViewById(R.id.email);
        password = findViewById(R.id.password);
        confirmPassword = findViewById(R.id.confirmPassword);
        phoneNumber = findViewById(R.id.phoneNumber);

        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                checkDataEntered();
            }
        });
    }

    boolean isEmail(EditText text) {
        CharSequence email = text.getText().toString();
        return
            (!TextUtils.isEmpty(email)
Patterns.EMAIL_ADDRESS.matcher(email).matches());
    }
}

```

```

        boolean isEmpty(EditText text) {
            CharSequence str = text.getText().toString();
            return TextUtils.isEmpty(str);
        }

        void checkDataEntered() {
            if (firstName > 80) {
                Toast t = Toast.makeText(this, "ФИО слишком длинное!",
Toast.LENGTH_SHORT);
                t.show();
            }

            if (nickname.exist()) {
                nickname.setError("Пользователь с таким никнеймом уже
существует!");
            }

            if (requireNotNull(fields).forEach{it.keys}) {
                fields.setError("Одно или несколько полей не
заполнено!");
            }
            if (!confirmPassword.equals(password)) {
                fields.setError("Пароли не совпадают!");
            }
        }
    }
}

```

study posters

Регистрация | Вход

Профиль ▼

Галёнкин Андрей Степа

Ник

Пароль

Повторите пароль

Контактный телефон

Электронная почта

Одно или несколько полей не заполнено!

Зарегистрироваться

Отмена

Рисунок 4.3.1.1 – Проверка на заполнение всех полей

study posters

Регистрация | Вход

Профиль ▼

Галёнкин Андрей Степа

Merlin

● ● ● ● ● ● ● ●

● ● ● ● ● ● ● ●

+380954524189

alter@gmail.com

Пользователь с таким никнеймом
уже существует!

Зарегистрироваться

Отмена

Рисунок 4.3.1.2 – Перевірка на існування користувача

Таблиця 4.3.1.

№	Тестові набори	Очікувані результати
1	firstLastName = Галєнкін Андрей Степанович nickname = null password = null confirmPassword = null email = null phone = null	Одно или несколько полей не заполнено!
2	firstLastName = Галєнкін Андрей Степанович nickname = Merlin password = 12345678 confirmPassword = 12345678 email = alter@gmail.com phone = +380954524189	Пользователь с таким никнеймом уже существует!

5 Розгортання та валідація програмного продукту пояснювальної записки курсової роботи

5.1 Інструкція з встановлення програмного продукту

Розроблене програмне забезпечення підтримується на будь-якому смартфоні з будь-якою версією Android.

Здійснювати дії у додатку та користуватися ним користувач може за допомогою сенсорного дисплею.

У подальшому планується зробити адаптацію для мобільних пристроїв версії iOS.

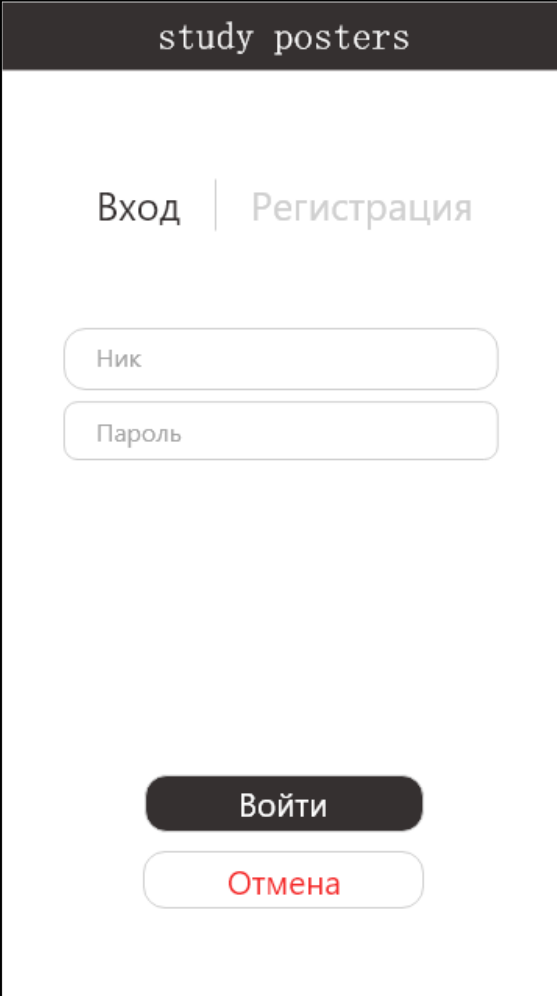
5.2 Інструкція з використання програмного продукту

5.2.1 Реєстрація користувача

ПП надає можливість користувачу «гість» вести параметри реєстрації (ПІБ, логін, пароль, повторити пароль, телефон, електронна пошта, як показано на рисунку 5.2.1.1.

Рис 5.2.1.1

ПП надає можливість користувачу «гість» вести параметри авторизації (ПІБ, логін, пароль, повторити пароль, телефон, електронна пошта, як показано на рисунку 5.2.1.2.



study posters

Вход | Регистрация

Ник

Пароль

Войти

Отмена

Рис 5.2.1.2

ПП надає можливість користувачу додатку передивлятися стрічку новин та подробну інформацію щодо афіши, яка їх зацікавила, як показано на рисунках 5.2.1.3 та 5.2.1.4

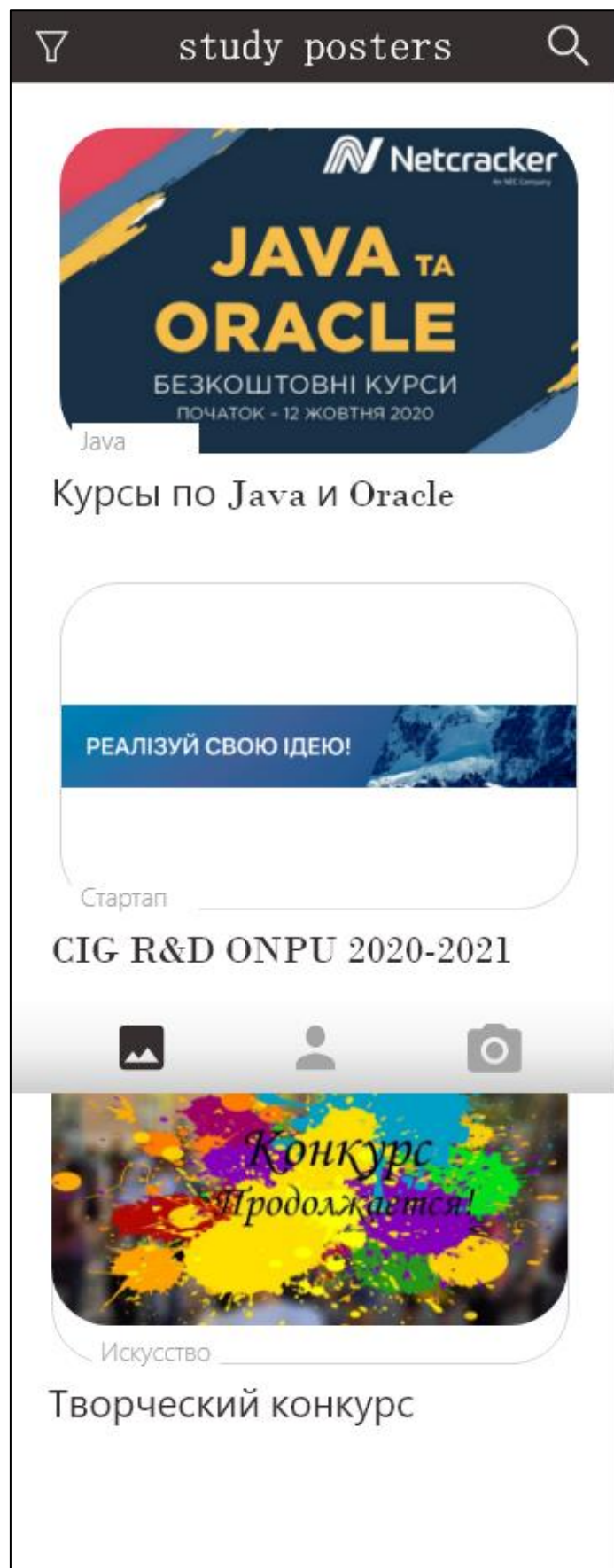


Рис 5.2.1.3



Рис 5.2.1.4

ПП надає можливість користувачу додатку додавати афіші та ключові слова до розділу «Обрані», як показано на рисунках 5.2.1.5 та 5.2.1.6

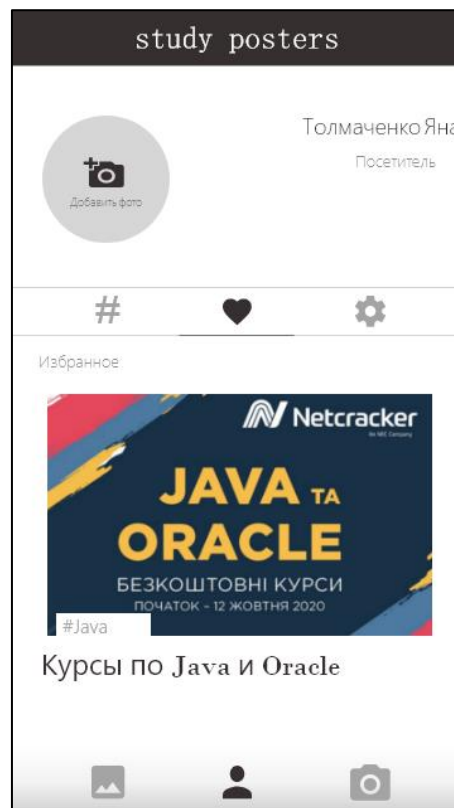


Рис 5.2.1.5

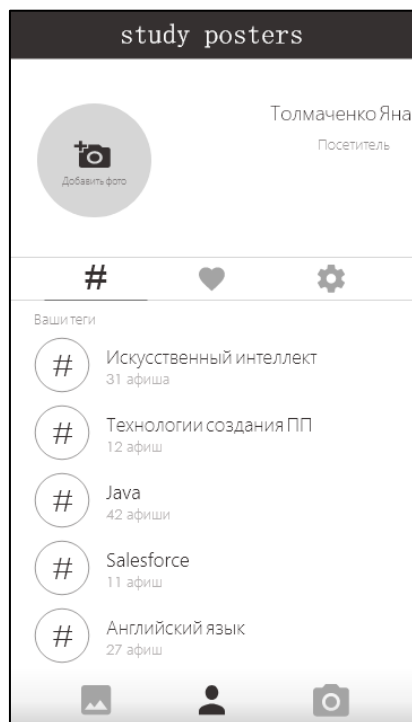


Рис 5.2.1.6

ПП надає можливість адміністратору додатку додавати нові афиши та редагувати їх, як показано на рисунках 5.2.1.7 та 5.2.1.8

Рис 5.2.1.7



Рис 5.2.1.8

5.3 Результати валідації програмного продукту

Метою програмного продукту було підвищення рівня інформативності студентів та викладачів з урахуванням інтересів кожного, структуризація афіш та загальної інформації щодо заходів що будуть проводитись у навчальному закладі.

В даний період часу проект знаходиться на ранній стадії свого розвитку, але внаслідок буде розвинений більше. Його глибше впровадження буде проходити з часом.

У розробленому ПП доступна можливість додавання афіш та ключових слів до обраних, отже мета підвищення рівня доступності інформації на цю тему виконана.

Розробка подібних інформаційних систем може значно допомогти як студентам, так і викладачам. Вона повинна вплинути на збільшення додаткового часу у користувачів, та структурувати їх професіональні інтереси.

Висновки

В результаті створення програмного продукту була досягнута наступна мета його споживача: «підвищення рівня інформативності студентів та викладачів з урахуванням інтересів кожного, структуризація афіш та загальної інформації щодо заходів що будуть проводитись у навчальному закладі.».

Доказом цього є наступні факти. Програмний продукт «Study Posters» задовольняє такі потреби споживача:

- 1) Пошук потрібних афіш;
- 2) Перегляд інформації щодо потрібної афіші;
- 3) Додавання потрібних афіш до розділу «Обране» ;
- 4) Додавання потрібні ключові слова до розділу «Обране»;
- 5) Можливість адміністраторів редагувати афіши.

В процесі створення програмного продукту виникли такі труднощі:

- 1) організаційні труднощі роботи у команді;
- 2) брак часу;
- 3) відсутність досвіду у android-розробці.

Через вищеописані непередбачені труднощі, а також через обмежений час на створення програмного продукту, залишилися нереалізованими такі прецеденти або їх окремі кроки роботи:

Зазначені недоробки планується реалізувати в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.