Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка мобільного додатку інтеграції соціальних мереж на платформі iOS»

Студента 4 курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» _____ підпис студента — Марценюка Антона Сергійовича

Науковий керівник кандидат технічних наук, доцент _____ підпис керівника — Харченко Олександр Анатолійович

Гарант освітньої програми кандидат технічних наук, доцент _____ — Цензура Микола Олександрович

КИЇВ – 2020

АНОТАЦІЯ

Дипломна робота на тему «Розробка мобільного додатку інтеграції соціальних мереж на платформі iOS» містить 41 сторінки, 31 рисунків та 1 додаток. Перелік посилань налічує 6 найменувань.

Мета дослідження: розробка мобільного додатка на платформі ios, що дозволяє інтегрувати кілька соціальних мереж для перегляду, створення, редагування і публікації постів.

Об’єкт дослідження: додаток для iOS, архітектура системи.

Предметом дослідження: розробка програмного продукту з інтеграцією соціальних мереж.

В першому розділі були проаналізовані існуючі рішення схожих додатків. Були виявлені недолікі та переваги кожного з них, які допомогли поєднати в собі функціональні вимоги додатку. Опрацьовані сценарії використання додатку. Перш за все, при складанні сценарію виявляється мета, якої будуть досягати користувачі на кожному з екранів додатка.

В другому розділі ми розглянули детально всі шари iOS архітектури, переглянули життєвий цикли мобільного додатку, як відбувається компілюється додаток, та обрали архітектурний патерн який будеме застосовувати для дизайну. Обрали мову програмування котра відповідає сучасним вимогам. Розглянули переваги середи розробки iOS додатків, яку підтримує Apple. Для оптимізація екранів застосовуємо Кординатор.

В третьому розділі було стоверннo зв'язок з сервером котрий буде зберігати данні про користувача. Створили навігацію по проекту між контролерами. Інетгрували дані до соціальних мереж та протестували додаток на коректність різних інтерфейсів на системі iOS.

ANNOTATION

The thesis on the topic "Development of a mobile application for the integration of social networks on the iOS platform" contains 41 pages, 31 drawings, and 1 appendix. The list of links includes 6 items.

The purpose of the study is to development of a mobile application on the ios platform, which allows you to integrate multiple social networks for viewing, creating, editing and publishing posts.

Research subject: application for iOS, system architecture.

The study's subject is software product development with integration of social networks.

The first section analyzed the existing solutions of similar applications. The disadvantages and advantages of each of them were identified, which helped to combine the functional requirements of the application. Scenarios for using the application have been processed. First of all, when compiling a script, the goal is revealed, which will be achieved by users on each of the screens of the application.

In the second section, we looked in detail at all the layers of the iOS architecture, reviewed the life cycles of the mobile application, how the application is compiled, and chose the architectural pattern that we will use for the design. They chose a programming language that meets modern requirements. Considered the benefits of the iOS application development environment supported by Apple. To optimize the screens we use the Coordinator.

In the third section there was a real connection with the server which will store data on the user. Created project navigation between controllers. We integrated data into social networks and tested the application for the correctness of various interfaces on the iOS system.

Київський національний торговельно-економічний університет

Факультет *інформаційних технологій* _____
Кафедра *інженерії програмного забезпечення та кібербезпеки*_____
Освітній ступінь *бакалавр*_____
Спеціальність *121 «Інженерія програмного забезпечення»*_____
Спеціалізація / освітня програма *121 «Інженерія програмного забезпечення»*

Затверджую
Зав. кафедри інженерії
програмного забезпечення
та кібербезпеки
Криворучко О. В.
"7" листопада 2019 р.

Завдання
на випускну кваліфікаційну роботу студентові

Марценюку Антону Сергійовичу

1. Тема випускної кваліфікаційної роботи: «*Розробка мобільного додатку інтеграції соціальних мереж на платформі iOS».*

Затверджена наказом ректора від *«02» грудня 2019 р. № 4112*

2. Строк здачі студентом закінченої роботи _____
_____
3. Цільова установка та вихідні дані до роботи:

***Мета роботи*:** *розробка мобільного додатка на платформі ios, що дозволяє інтегрувати кілька соціальних мереж для перегляду, створення, редагування і публікації постів».*

Об'єкт дослідження: *додаток для iOS, архітектура системи.*

Предмет дослідження: *розробка програмного продукту з інтеграцією соціальних мереж.*

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

Вступ

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ФУНКЦІОНАЛЬНІ ВИМОГИ

1.1 Facebook Pages Manager

1.2 Buffer

1.3 HootSuite

1.4. Функціональні вимоги

1.5. Діаграми варіантів використання

1.6. Діаграми варіантів використання

1.7 Висновки до розідлу 1

РОЗДІЛ 2. АРХІТЕКТУРА IOS ДОДАТКУ

2.1. Архітектура ОС iOS

2.1.1. Життєвий цикл iOS додатку

2.1.2. Потоки iOS додатка

2.1.3. Використовувані інструменти розробки

2.1.3.1. Середовище розробки

2.1.3.2. Мова програмування

2.2. Архітектура iOS додатків

2.2.1. View

2.2.2. ViewController

2.3. MVVM и Coordinators в iOS приложениях

2.3.1. MVVM

2.3.2. Coordinator

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
	Вибір теми випускної кваліфікаційної роботи		
	Вступ та перелік літературних джерел		
	Розділ 1. Огляд існуючих рішень та функціональні вимоги		
	Розділ 2. Архітектура ios додатку		
	Розділ 3 Розробка додатку		
	Висновки		
	Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)		
	Підготовка автореферату та презентації доповіді		
	Попередній захист випускної кваліфікаційної роботи		
	Зовнішнє рецензування випускної кваліфікаційної роботи		
	Здача прошитої випускної кваліфікаційної роботи на кафедру		
	Публічний захист випускної кваліфікаційної роботи		

7. Дата видачі завдання «___»_____20__ р.

8. Науковий керівник випускної кваліфікаційної роботи Харченко А. О.

9. Гарант освітньої програми Цензура М. О.

10. Завдання прийняв до виконання студент Марценюк А. С.

11. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Науковий керівник випускної кваліфікаційної роботи ________________

(підпис, дата)

Відмітка про попередній захист Харченко А. О. _______________

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студента Мороза Я. Р. може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Цензура М. О.

Завідувач кафедри Криворучко О. В.

«_____»__________________20 ___ р.

ЗМІСТ

ВСТУП

У сучасному світі мобільний телефон і соціальні мережі стали невід'ємною частиною життя кожної людини. Трафік мобільного інтернету зростає з кожним роком. Мобільні додатки мають перевагу над комп'ютерними, тим, що ними можна користуватися в будь-якому місці світу: по дорозі на роботу, в кафе на обіді, на прогулянці в парку. Мобільний програмне забезпечення вирішує більшість повсякденних завдань людини, що дозволяє звести час витрачений на рутинні справи до нуля.

Соціальні мережі дозволяють додавати друзів, спілкуватися користувачам текстовими повідомленнями, обмінюватися фотографіями і відеозаписами, вступати в групи відрізняються їх інтересам і бути в курсі останніх новин і подій. У наш час соціальні мережі можуть поборотися за увагу з такими великими медійними апаратами як новинні портали, відеохостінги, месенджери, розважальні портали.

Більшість юридичних компаній і індивідуальних підприємців ведуть свою діяльність в соціальних мережах. Соціальні мережі дозволяють просувати свою бізнес ідею, нарощувати нових потенційних клієнтів, сповіщати свою аудиторію про спеціальні умови і пропозиції, вести блог про свою діяльність, реагувати на відгуки та запитання клієнтів.

З кожним роком з'являється нова соціальна мережа. Зараз налічується близько 10 популярних соціальних мереж. Найбільші це Facebook, Twitter, Linkedin, Instagram, Pinterest. Працювати окремо з кожною дуже клопітно, так як будь-яка соціальна мережа має свої особливості, плюси і мінуси для просування своєї компанії. У великих компаніях є кілька акаунтів для соціальної мережі, що значно збільшує ефективність, але також і збільшує трудомісткість по

використання такого підходу.

Щоб усунути ці проблеми, в рамках даної дипломної роботи було вирішено розробити мобільний додаток на платформі iOS з можливістю керувати безліччю соціальних мереж та акаунтами належним їм.

Метою дослідження випускної кваліфікаційної роботи є розробка мобільного додатка на платформі ios, що дозволяє інтегрувати кілька соціальних мереж для перегляду, створення, редагування і публікації постів. *Об'єктом дослідження* є додаток для iOS, архітектура системи.

Завдання дослідження:

- Проаналізувати існуючі рішення;
- Вивчити особливості мобільної платформи iOS;
- Сформувати список вимог щодо розроблюваного з додатком;
- Ознайомитися з різними API соціальних мереж;
- Спроектувати архітектуру програми;
- Реалізувати програму;
- Протестувати додаток.

Метод дослідження: теоретичне дослідження і практична реалізація.

Область застосування - організації, для поширення в AppStore.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ФУНКЦІОНАЛЬНІ ВИМОГИ

У даній предметній сфері існує багато готових рішень. Розглянемо кілька найпопулярніших на даний момент, що допомагають працювати з соціальними мережами на мобільних пристроях

1.1. Facebook Pages Manager

Рис. 1.1 – Головна сторінка

Рис. 1.2 – Активність

Додаток Facebook Pages Manager надає користувачеві керувати акаунтами соціальної мережі Facebook. Додаток підтримує додавання декількох акаунтів.

Переваги:

- Підтримка збору загальної статистики сторінки (Рис. 1.1);
- Отримання інформації про кожний запис (лайки, репости, перегляди) (Рис. 1.2).

Недоліки:

- Немає підтримки декількох соціальних мереж;
- Немає можливості планувати пости на певний час.

1.2. Buffer

Рис. 1.3 – Створення посту

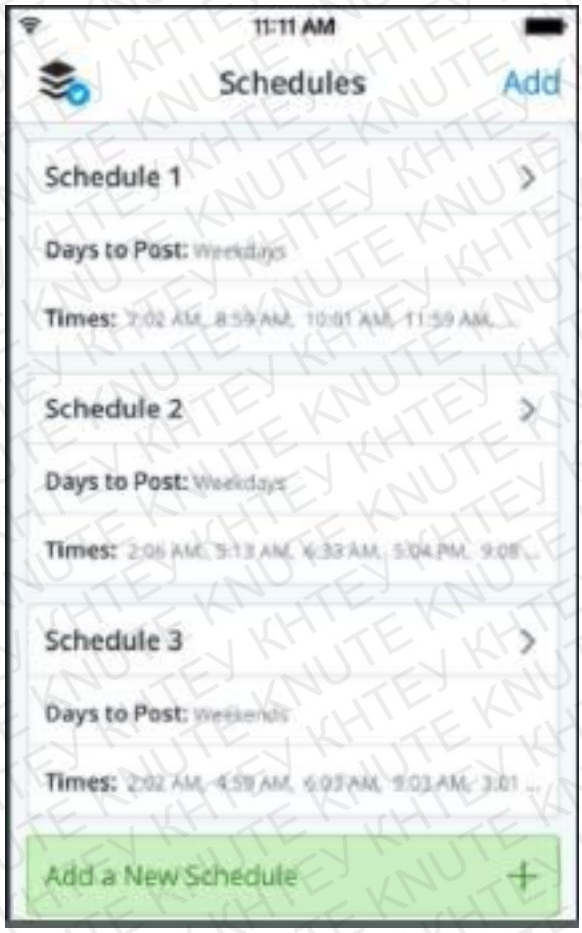


Рис. 1.4 – Графік посту

Додаток Buffer об'єднує в собі кілька соціальних мереж, що дозволяє створити один пост для всіх соціальних мереж за раз. Додаток дозволяє редагувати створені пости раніше, додавати фотографії (Рис. 1.3).

Переваги:

- Підтримка декількох соціальних мереж;
- Можливість запланувати пост (Рис. 1.4).

Недоліки:

- Підтримка одного аккаунта на соціальну мережу;
- Незручна система планування постів, не можна просто ввести дату передбачуваного часу поста, потрібно порахувати через скільки це настане.

1.3. HootSuite

Рис. 1.5 – Гафік посту

Рис. 1.6 – Пошук

Додаток HootSuite працює з соціальною мережею twitter. Користувач має можливість подивитися детальну статистику по постам і сторінки в цілому.

Переваги:

- Підтримка планування поста на конкретну дату (Рис. 1.5);
- Отримання повідомлень з соціальних мереж (Рис. 1.6) .

Недоліки:

- Немає підтримки додавання різних мереж;
- Немає підтримки створення поста з відео.

Таким чином, вже існуюче програмне забезпечення не дозволяє повністю охопити весь список вимог, що свідчить про актуальність цієї роботи.

1.4. Функціональні вимоги

Система повинна являти собою мобільний додаток працює на операційній система iOS. Додаток має мати доступ в інтернет, а також підтримувати різні пристрої, такі як iPhone, iPad, iPod. Розглянемо основні компоненти проектування, описані в керівництві [1].

Система повинна мати можливість:

1) Реєструвати користувача.
2) Відновлювати пароль.
3) Авторизувати користувача.
4) Додати аккаунт соціальної мережі.
5) Видаляти акаунт соціальної мережі.
6) Створювати пост.
7) Використовувати камеру для додавання фото або відео до посту.
8) Використовувати галерею для додавання фото або відео до посту.
9) Редагувати пост.
10) Призначати час автоматичного опублікування поста.
11) Опублікувати пост моментально.

12) Переглянути список запланованих на публікацію постів.

13) Переглянути список опублікованих постів.

14) Отримувати і обробляти push повідомлення.

1.5. Діаграми варіантів використання

На рисунках 1.7- 1.9 показані діаграми варіантів використання додатку.

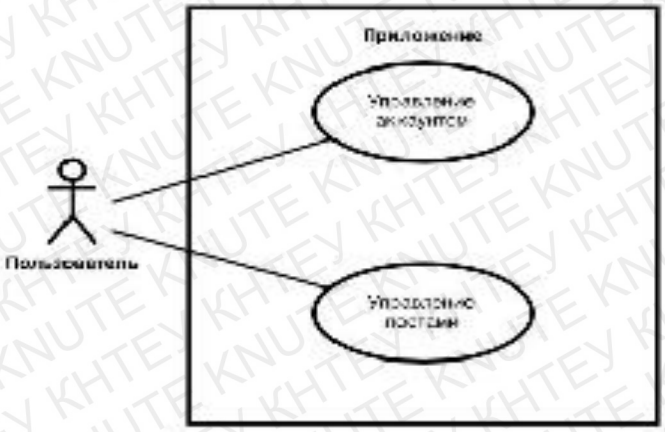


Рис. 1.7 – Загальна діаграма варіантів використання.

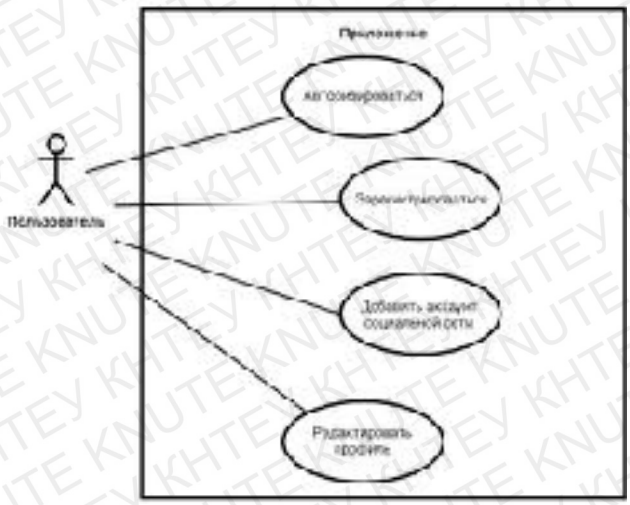


Рис. 1.8 – Діаграма варіантів використання "Управління аккаунтом".

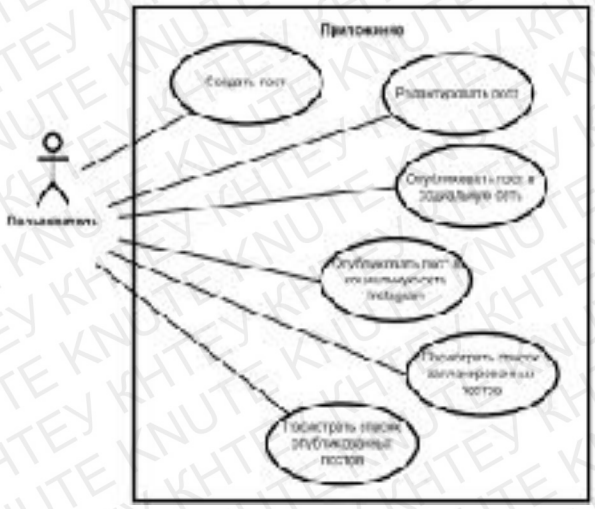


Рис. 1.9 – Діаграма варіантів використання "Управління постами".

1.6. Сценарії варіантів використання

Сценарій варіанти використання «Авторизуватись»:

1. Користувач вводить логін і пароль в текстові поля на екрані авторизація.
2. Система перевіряє дані на валідність.
3. Користувач натискає на кнопку «Авторизуватись».
4. Система відправляє на сервер введені дані.
5. Система отримує відповідь від сервера і показує помилку якщо користувач ввів некоректні дані або направляє його на головний екран.

Сценарій варіанти використання «Зареєструватися»:

1. Користувач вводить username, пошту, пароль, перевірки пароль в текстові поля на екрані реєстрація.
2. Система перевіряє, що паролі збігаються.
3. Користувач натискає на кнопку "Зареєструватися".
4. Система відправляє на сервер введені дані.
5. Система отримує відповідь, після чого відкриває головний екран додатка якщо авторизація пройшла успішно, в іншому випадку показує помилку.

Сценарій варіанти використання «Редагувати профіль»:

1. Користувач натискає на кнопку "Налаштування" в лівому меню головного екрану.

2. Система відкриває екран настройки.

3. Користувач може змінити пошту і картинку, потім натиснути на кнопку “Зберегти"

4. Система відправляє данні на сервер та оновлює дані користувача.

Сценарій варіанти використання «Створити пост»:

1. Користувач натискає на кнопку "Створити пост" на головному екрані.

2. Система відкриває екран створення поста.

3. Користувач вводить заголовок поста, вибирає фото або відео для поста.

4. Користувач натискає на кнопку "Обрати акаунти".

5. Система відкриває екран зі списком доступних акантів користувачеві.

6. Користувач вибирає акаунти і натискає кнопку "Готово".

7. Система повертає користувача на екран створення поста, а також відображає список обраних акаунтів соціальних мереж.

8. Користувач натискає на кнопку «Вибрати дату».

9. Система показує екран вибору дати.

10. Користувач вибирає число на календарі, виставляє годинник і хвилини, потім натискає кнопку “Готово".

11. Система повертає користувача на головний єкран, де відображається створений новий пост в списку запланованих постів.

Сценарій варіанти використання «Редагувати пост»:

1. Користувач натискає кнопку "Редагувати" на конкретному пості, на екрані запланованих постів.

2. Система відкриває екран для редагування поста.

3. Користувач змінює заголовок, фото або відео, обірає акаунти со-

ціальних мереж і виставляє дату і натискає кнопку «Готово».

4. Система видправляє нові данні на сервер та повертає користувача на головний екран.

Сценарій варіанти використання «Опублікувати пост в соціальну мережу»:

1. Користувач натискає кнопку "Опублікувати" на конкретному пості, в списку запланованих постів.

2. Система відкриває екран з детальною інформацією про піст.

3. Користувач переглядає вміст поста і натискає кнопку "Опублікувати".

4. Система публікує пост в соціальну мережу, видаляє його із запланованого списку і додає до опублікованого списку, повертає користувача на головний екран.

Сценарій варіанти використання «Опублікувати пост в соціальну мережу Instagram»:

1. Користувач натискає на кнопку "Опублікувати" на конкретному пості, який належить соціальної мережі Instagram.

2. Система відкриває екран з детальною інформацією про піст.

3. Користувач переглядає вміст поста і натискає кнопку "Опублікувати".

4. Система відкриває встановлений додаток Instagram на екрані створити пост.

5. Користувач натискає кнопку "Опублікувати" в додатку Instagram.

6. Система публікує пост в соціальну мережу, видаляє його з запланованого списку і додає до опублікованого списку, повертає користувача на головний екран.

Сценарій варіанти використання «Подивитися перелік запланованих постів»:

1. Користувач натискає на кнопку "Заплановані" в лівому меню го-

ловного екрану.

2. Система відкриває екран зі списком запланованих постів.

Сценарій варіанти використання «Подивитися перелік опублікованих постів»:

1. Користувач натискає на кнопку "Опубліковані" в лівому меню головного екрану.

2. Система відкриває екран зі списком опублікованих постів.

1.7. Висновки до розідлу 1

В першому розділі були проаналізовані існуючі рішення схожих додатків. Були виявлені недолікі та переваги кожного з них, які допомогли поєднати в собі функціональні вимоги додатку. Опрацьовані сценарії використання додатку. Перш за все, при складанні сценарію виявляється мета, якої будуть досягати користувачі на кожному з екранів додатка. У разі повідомлень, мета - бути в курсі подій в своєму оточенні, тобто всього, що пов'язано з самим користувачем, його друзями, колегами і т.д. Після отримання повідомлення користувач може перейти на відповідну сторінку. Залежно від типу повідомлення (оповіщення з ігор, запрошення до дружби, коментування, позначки «Подобається» ...) можливі різні сценарії. Користувач може переходити в ігри або профіль іншого користувача, може написати кому-небудь повідомлення, і так далі.

РОЗДІЛ 2
АРХІТЕКТУРА IOS ДОДАТКУ

2.1. Архітектура ОС iOS

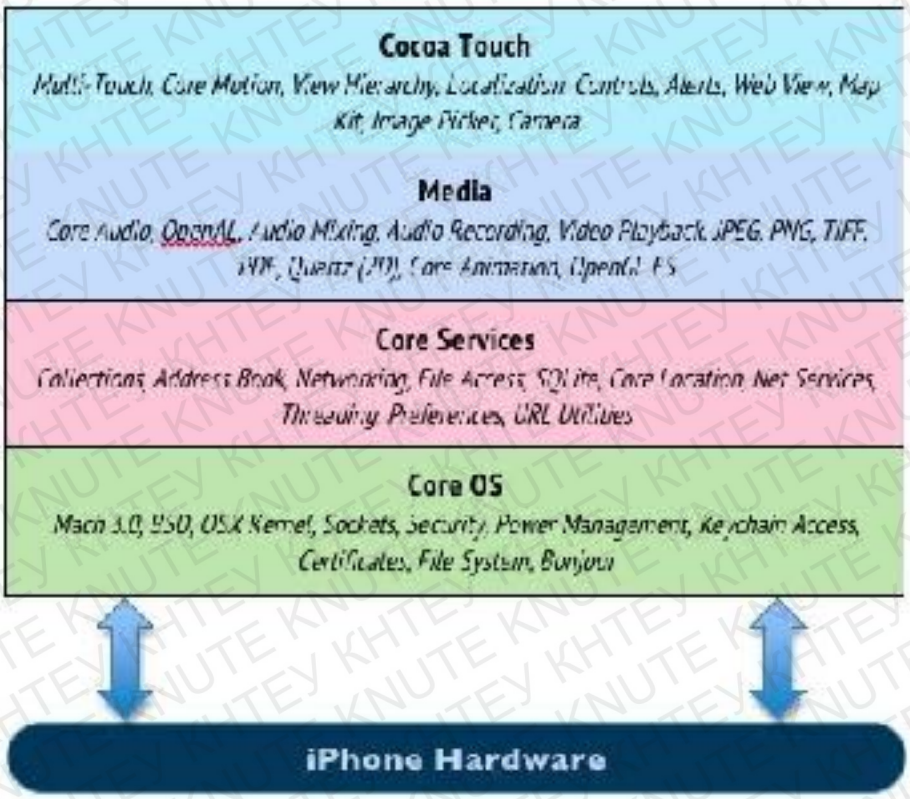


Рис. 2.1 – Шари ОС iOS.

Архітектуру iOS можна розділити на 4 шари (Рис. 2.1):

1) Рівень ядра (Core OS) - працює з файловою системою, контролює дійсність різних сертифікатів, що належать додаткам [2]. Також відповідає за безпеку всієї системи в цілому. Містить низький рівень доступу до елементів пристрою.

2) Рівень сервісів ядра (Core Service) - надає доступ до таких сервісів

як:

- Вбудована база даних (SQLite), що дозволяє створювати модель даних, а також здійснювати SQL запити.
- Файловий диспетчер (File Accesses), сервіс має високий рівень інтерфейсу для доступу до різних файлів належить додатком.
- Локація (Core Location) - сервіс, який дозволяє відстежити своє поточне місцезнаходження, а також відстежувати його тривалі переміщення.
- Інтернет сервіс (Net Service) - сервіс, який дозволяє отримувати і передавати дані через мережу інтернет.

3) Рівень медіа (Media) - містить інструменти дозволяють обробляти більшість форматів мультимедійних файлів, наприклад:

- Аудіо (Core Audio) - дозволяє записувати, прослуховувати, а також і редагувати різні аудіо доріжки. Можливість дістати з відео, певну доріжку.
- Відео (Video playback) - сервіс, який дозволяє відтворювати, перемотувати, зупиняти відео. Редагування відео на увазі зміну розширення, довжини або якості.
- Анімація (Core Animation) - бібліотека дозволяє створювати анімації. При створенні анімації може використовуватися як показ картинок послідовно, так і зміна різних параметрів елементів інтерфейсу, таких як висота, ширина, розташування, видимість і т.д.

4) Рівень ядра (Core OS) - працює з файловою системою, контролює дійсність різних сертифікатів, що належать додаткам [2]. Також відповідає за безпеку всієї системи в цілому. Містить низький рівень доступу до елементів пристрою.

5) Рівень сервісів ядра (Core Service) - надає доступ до таких сервісів як:

- Вбудована база даних (SQLite), що дозволяє створювати модель даних, а також здійснювати SQL запити.

- Файловий диспетчер (File Accesses), сервіс має високий рівень інтерфейсу для доступу до різних файлів належить додатком.
- Локація (Core Location) - сервіс, який дозволяє відстежити своє поточне місцезнаходження, а також відстежувати його тривалі переміщення.
- Інтернет сервіс (Net Service) - сервіс, який дозволяє отримувати і передавати дані через мережу інтернет.

6) Рівень медіа (Media) - містить інструменти дозволяють обробляти більшість форматів мультимедійних файлів, наприклад:

- Аудіо (Core Audio) - дозволяє записувати, прослуховувати, а також і редагувати різні аудіо доріжки. Можливість дістати з відео, певну доріжку.
- Відео (Video playback) - сервіс, який дозволяє відтворювати, перемотувати, зупиняти відео. Редагування відео на увазі зміну розширення, довжини або якості.
- Анімація (Core Animation) - бібліотека дозволяє створювати анімації. При створенні анімації може використовуватися як показ картинок послідовно, так і зміна різних параметрів елементів інтерфейсу, таких як висота, ширина, розташування, видимість і т.д.

7) Рівень інтерфейсу (Cocoa Touch) - має безліч елементів для створення мобільних інтерфейсів, а а також надає іншим верствам інформацію вхідну від користувача. У нього входять такі елементи як:

- Оброблювач множинних натискань (Multi-Touch) - дозволяє обробляти кілька натискань на екран.
- Оброблювач жестів (Core Motion) - дозволяє обробити різні жести виробляються на екрані пристрою, такі як зрушення вправо, вниз і т.д.
- Галерея (ImagePickers) - надає доступ до фотографій і відео лежать на пристрої. За допомогою нього додатки можуть отримати доступ і завантажити до себе файли з галереї.
- Камера (Camera) - використовується для знімком і подальшим за-

вантаженням даних в додаток.

2.1.1. Життєвий цикл iOS додатку

Додатки мають кілька станів, щоб дізнатися як вони змінюються, розглянемо життєвий цикл (Рис. 2.2).

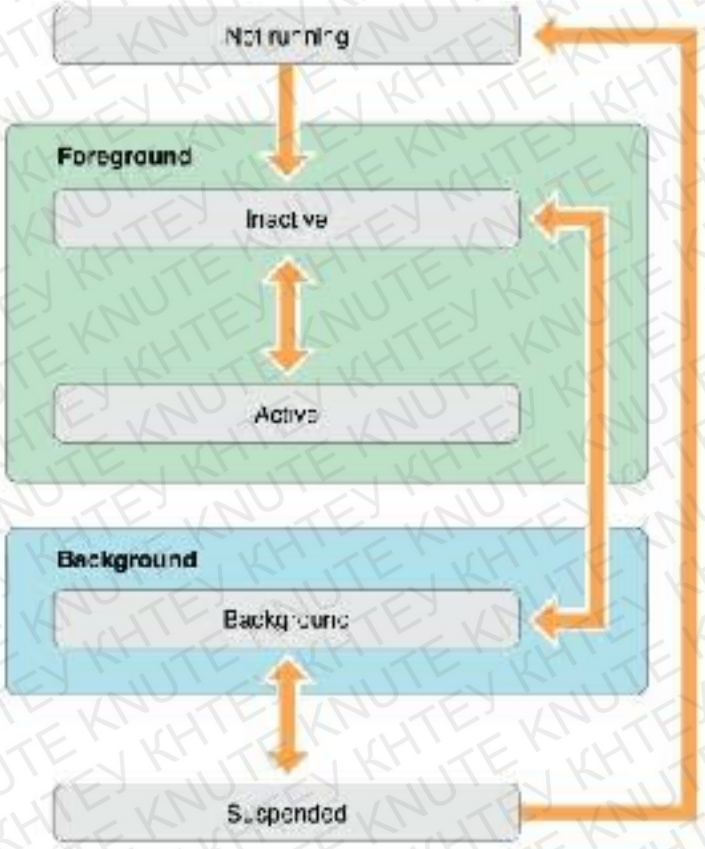


Рис. 2.2 – Стани iOS додатку.

Стану залежать один від одного і змінюються строго своєю ієрархією. Кожна програма починає свою роботу після натискання іконки програми [3]. Розглянемо стану докладніше:

1) Not running - програма не запущено або система видалила його з пам'яті.

2) Inactive - додаток запущено на передньому плані, але все ще не приймає події. Зазвичай додаток залишається в цьому стані тільки не

надовго, оскільки переходить в інший стан.

3) Active - звичайний режим при якому додаток працює на передньому плані і приймає події.

4) Background - додаток знаходиться в фоновому режимі і виконує якийсь код. Більшість додатків вводять цей стан не надовго, після чого йде припинення програми. Однак якщо потрібно більше часу для роботи у фоновому режимі, додаток може запитати його у системи і залишитися в цьому стані на деякий час.

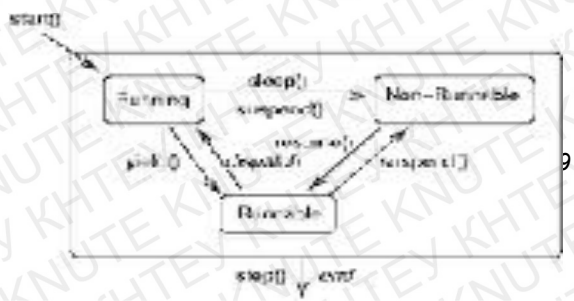
5) Suspended - додаток знаходиться в фоновому режимі і не виконує код. Система автоматично переводить додатки в цей стан і не повідомляє їх про це. Додаток призупинено, але залишається в пам'яті. При виникненні ситуації, коли закінчується пам'ять на пристрої, система може видалити призупинені додатки без повідомлення, щоб звільнити місце для наступного додатки.

Додатки повинні бути готові до припинення роботи в будь-який час і не повинні чекати, щоб зберегти призначені для користувача дані і виконати інші важливі завдання.

Користувач також як і система здатний видалити програму з пам'яті використовуючи багатозадачність інтерфейс, в цьому випадку програма не отримає ніяких повідомлень про це.

2.1.2. Потоки iOS додатка

Додаток створює головний потік вашого додатку і при необхідності можна створювати додаткові потоки для виконання інших завдань [4]. Для iOS додатків найкращим методом є використання Grand Central Dispatch (GCD), операційних об'єктів і інших асинхронних інтерфейсів програмування, а не створення і управління потоками самостійно. Такі технології, як GCD, дозволяють визначати роботу, яку ви хочете зробити, і порядок, в якому ви хочете це



зробити, а також дозволяти системі вирішувати, як краще за все виконати цю роботу на доступних CPU (Рис. 2.3).

Рис. 2.3 – Робота потоків в iOS додатку.

Надання системі управління потоками спрощує код, який ви повинні писати, спрощує забезпечення правильності цього коду і забезпечує більш високу загальну продуктивність.

Розглянемо основні підходи роботи з потоками:

1. Робота з уявленнями, основний анімацією і багатьма іншими класами UIKit зазвичай повинна відбуватися в головному потоці програми. Існують деякі винятки з цього правила - наприклад, маніпуляції на основі зображень часто можуть виконуватися у фоновому потоці, але якщо є якісь труднощі з виконанням, тоді слід запускати цю задачу в головному потоці.
2. Довгі завдання, завжди повинні виконуватися у фоновому потоці.
3. Будь-які завдання, пов'язані з мережевим доступом, доступом до файлів або великими обсягами обробки даних, повинні виконуватися асинхронно з використанням об'єктів GCD або операцій.

Під час запуску ваше застосування повинне використовувати доступне час для установки свого призначеного для користувача інтерфейсу якомога швидше. У головному потоці повинні виконуватися тільки ті завдання, які сприяють налаштування призначеного для користувача інтерфейсу. Всі інші завдання повинні виконуватися асинхронно, результати яких мають бути показані користувачеві, як тільки вони будуть готові.

2.1.3. Використовувані інструменти розробки

2.1.3.1. Середовище розробки

Xcode - інтегроване середовище розробки під macOS і iOS, розроблена компанією Apple, що дозволяє створювати додатки для iPhone, iPad, Apple Watch, Mac і Apple TV. Першу версію анонсували в 2001 році. Програмне за-

безпечення є безкоштовним. Xcode містить в собі велику частину документації розробників від Apple. На момент написання використовувався Xcode 11.

Основні переваги Xcode:

1) Текстовий редактор (Source Editor) - інструмент дозволяє розширення коду, згортання коду, здійснює підсвічування синтаксису, відображає попередження, помилки та іншу контекстно-залежну інформацію, вбудовану в ваш код.

2) Каталог картинок (Asset Catalog) - інструмент дозволяє управляти зображеннями вашого додатками, групуючи дозволу одного і того ж ресурсу.

3) Редактор версії (Version Editor) - інструмент дозволяє відображати поточну тимчасову шкалу коммітов, допомагає визначити причину збою і дозволяє повертати файли до минулих коммітов для порівняння з вихідним файлом.

4) Симулятор (Simulator) - за допомогою iOS SDK XCode створювати, встановлювати, запускати і налагоджувати написані програми в симуляторі на базі Mac, що оптимізує робочий процес розробки.

5) Вбудований інтерфейс билдер (Interface Builder) - інструмент дозволяє створювати і тестувати інтерфейс без написання рядка коду.

2.1.3.2. Мова програмування

Swift - відкритий компільований мову програмування використовується в першу чергу для створення програмного забезпечення під iOS і Mac. Мова сумісний з основною кодової базою, написаної на Objective-C.

Переваги мови:

1) Більш легкий для читання і стійкий до помилок програміста мова, ніж передував йому Objective-C.

2) Більш безпечний. Swift містить безліч інструментів для роботи з опціональнимі типами, що дає програмісту впевненість в написаному їм коді.

3) Підтримує динамічні бібліотеки.
4) Swift набагато швидше, ніж його попередник Objective-C.
5) Спрощує успадкування класів.

2.2. Архітектура iOS додатків

Для початку потрібно познайомитися з двома основними компонентами iOS додатки це View і ViewController (Рис. 2.4).

Рис. 2.4 – Взаємозв'язок головних компонентів.

2.2.1. View

У додатках на платформі iOS за користувальницький інтерфейс відповідає один або кілька компонентів уявлень (view). Загальний інтерфейс складається з безлічі компонентів (subviews) успадковані від загального класу View (Рис. 2.5).

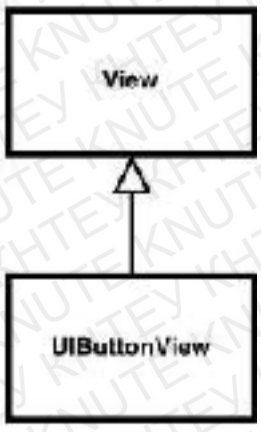


Рис. 2.5 – Успадкування UIButtonView від View.

До таких доступним елементів інтерфейсу відносяться: текстові поля, кнопки, таблиці, контейнери для картинок та інше.

Компоненти здатні групувати між собою елементи:

1) TableView - групує елементи TableViewCell у вигляді горизонтального списку, де в одному рядку буде знаходитися тільки один елемент.

2) CollectionView - групує елементи CollectionViewCell, як в горизонтальний, так і вертикальний список, в одному рядку може бути кілька елементів.

3) Horizontal Stack View і Vertical Stack View - стек елементів, що дозволяє з легкістю налаштувати загальні розміри, відступи елементів.

4) ScrollView - контейнер, може містити в собі будь-які елементи, а також розширюється в залежності від кількості елементів в ньому.

Існує кілька способів створення view:

1) Створювати і настроювати view програмно.

2) Використовувати інтерфейс білдер (Рис. 2.6).

Рис. 2.6 – Інтерфейс білдер.

При створення своєї view надалі з її допомогою можна будувати інтерфейси різних екранів.

Основні вимоги до дизайну інтерфейсу мобільних додатків:

1) Мінімізація кількості елементів на екрані. Не слід перевантажувати невеликий простір дисплея мобільного пристрою великою кількістю об'єктів. Необхідно постаратися вмістити максимум функціоналу в лаконічний і доброзичливий інтерфейс.

2) Управління сучасними смартфонами з сенсорними екранами здійснюється за допомогою пальця. З огляду на те, що площа дотику пальця набагато більше розмірів покажчика комп'ютерної миші, інтерфейс не повинен містити дрібних елементів. Якщо ж наприклад кнопка має маленькі розміри, що ускладнює потрапляння по ній пальцем, то слід збільшити простір навколо кнопки, яке також обробляє дотику.

3) Розмір тексту в полях додатки повинен бути досить великий, щоб була можливість з легкістю його читати на віддаленому відстань від пристрою.

2.2.2. ViewController

ViewControlers - компоненти iOS додатки, що відповідають за управління набором view і займаються навігацією за додатком [5]. Зазвичай компоненти створюються ізольовано один від одного, мають різні уявлення [6]. Так наприклад один контролер може показувати список елементів, а в той час інший контролер буде відображати обраний елемент з цього списку (Рис. 2.7).

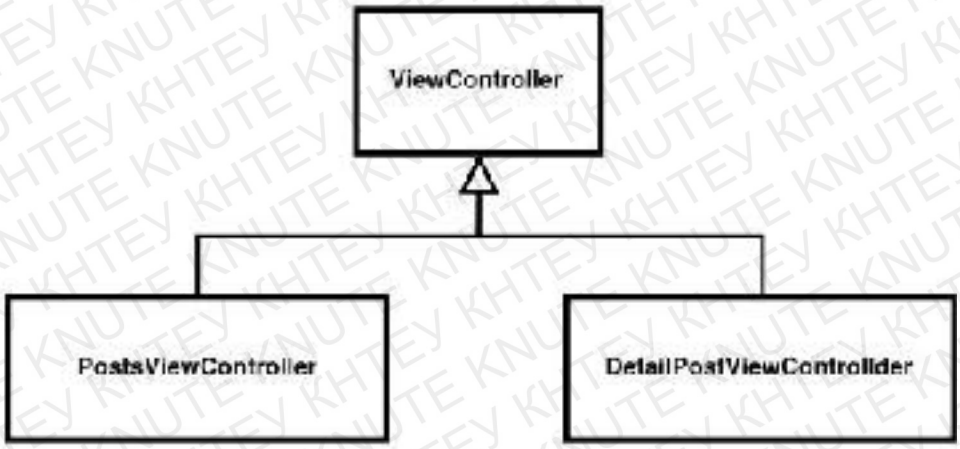


Рис. 2.7 – Успадкування контролерів.

ViewController виконує такі завдання як:

1) Оновлює зміст view, звичайно при оновленні будь-яких даних, оновлюється і уявлення.

2) Реагує на взаємодію користувача з інтерфейсом.

3) Змінює розміри і перемальовує уявлення в залежності від отриманих подій: переворот екрану, додавання нового елемента та інші.

2.3. MVVM и Coordinators в iOS приложениях

2.3.1. MVVM

MVVM - шаблон застосовують для проектування архітектури додатку, орієнтований на сучасні платформи розробки [7]. Реалізація патерну для iOS архітектури містить 4 елементи: ViewController, View, Model, ViewModel (Рис. 2.8).

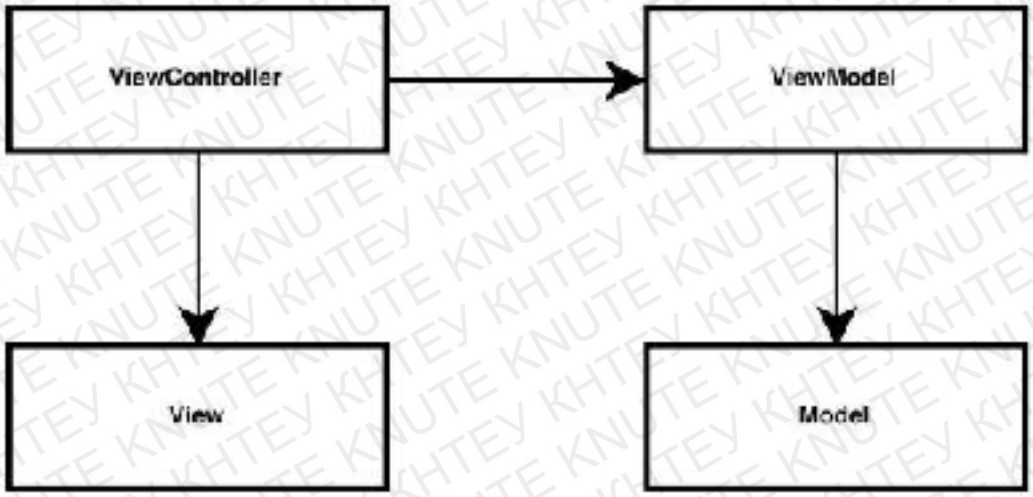


Рис. 2.8 – Модель MVVP в iOS додатку.

MVVM застосовується для поділу моделі і її уявлення, що необхідно для зміни їх окремо один від одного. Наприклад, розробник працює з логікою даних, в той час дизайнер відповідно розробляє призначений для користувача інтерфейс. MVVM зручно використовувати замість класичного MVC і йому подібних в тих випадках, коли в платформі, на якій ведеться розробка, присутня явна зв'язаність між призначеним для користувача інтерфейсом і бізнес логікою.

2.3.2. Coordinator

Coordinator (Координатор) - патерн дозволяє инкапсулировать всю навігацію по додатком [8]. Зазвичай відповідальний за перехід на наступний контролер був поточний контролер. Об'єкт координатор дозволяє взяти всі ці обов'язки на себе. Він візьме на себе управління переходами між контролерами, що сприяє збільшенню слабких залежностей і правильного перевикористання коду. Координатор може управляти також координаторами, доданими до головного батьківського координатору. Розглянемо основний інтерфейс координатора (Рис. 2.9)

Рис. 2.9 – Інтерфейс координатора.

Зазвичай додаток стартує з головного координатора, який в залежності від стану покаже необхідний додатком координатор. Об'єднавши координатор з патерном MVVM ми отримаємо наступну структуру (Рис. 2.10)

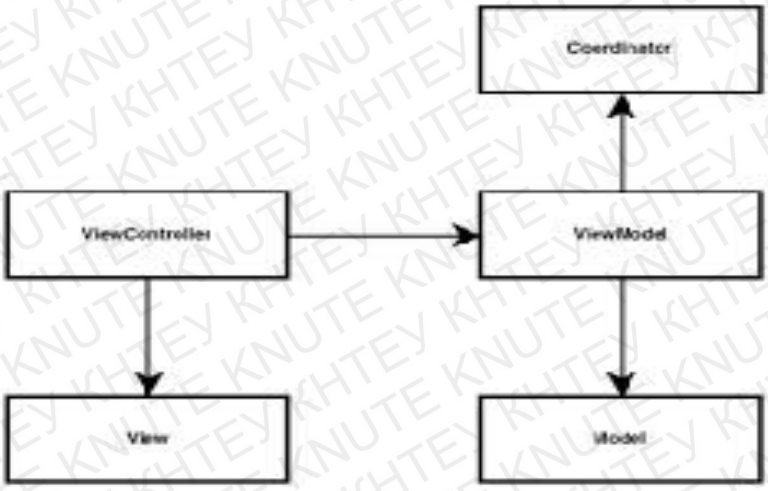


Рис. 2.10 – модель MVVM и Coordinator.

Розглянемо обов'язки кожного елемента архітектури докладніше:

1) View (Представлення) - відображає призначений для користувача інтерфейс.

2) Model (Модель) - містить в собі призначені для користувача дані, так ж виробляє з ними маніпуляції і повідомляє про всі зміни ViewModel.

3) ViewModel (Модель подання) - отримує дані від моделі і перетво-

рює їх в потрібний вид. Отримує повідомлення від контролера і передає їх координатору, для коректної навігації в додатку.

4) ViewController (Контролер) - запитує дані у ViewModel і використовуючи їх формує уявлення потрібним чином. Обробляє і передає події вчинені користувачем ViewModel.

5) Coordinator (Координатор) - обробляє повідомлення отримані від ViewModel, тим самим робить навігацію по додатком.

Позбавлення контроллеов від обязонностей роботи з моделлю даних і навігації за додатком, робить їх набагато читабельнее і дозволяє використовувати їх в інших проектах. Избавление контроллеов от обязоности работы с моделью данных и навигацие по приложению, делает их намного читабельнее и позволяет использовать их в других проектах.

2.4. Висновки до розідлу 2

В другому розділі ми розглянули детально всі шари iOS архітектури, переглянули життєвий цикли мобільного додатку, як відбувається компілюється додаток, та обрали архітектурний патерн який будеме застосовувати для дизайну. Обрали мову програмування котра відповідає сучасним вимогам. Розглянули переваги середи розробки iOS додатків, яку підтримує Apple. Для оптимізація екранів застосовуємо Кординатор.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ

3.1. Архітектура системи

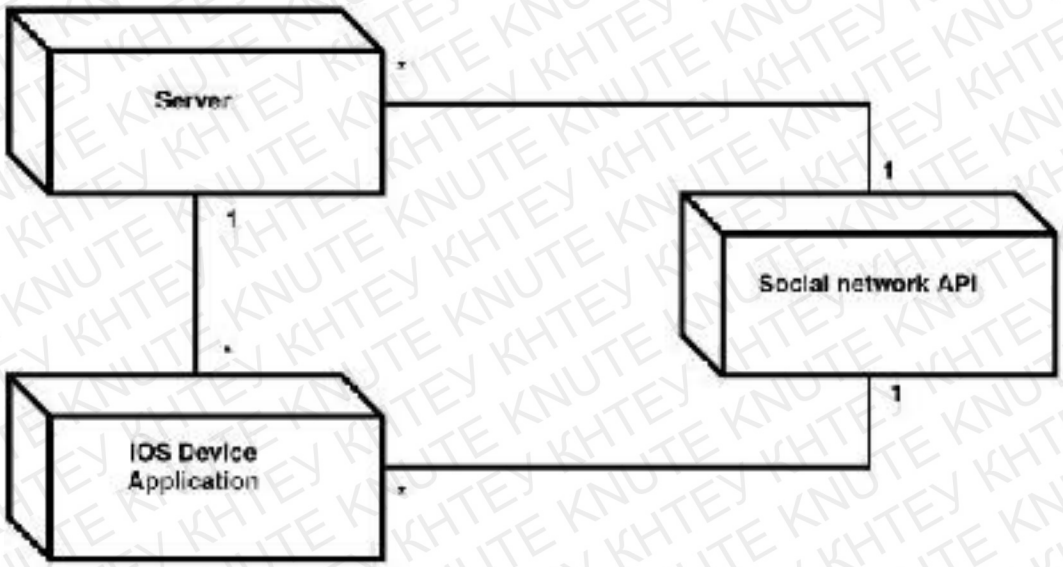


Рис. 3.1 - Діаграма розгортання системи.

Розглянемо систему розгортання на малюнок 20. Система складається з 3 частин:

1) Додатки для платформи iOS.
2) WebService
3) Група серверів соціальних мереж, що надають API.

У даній дипломній роботі сервер не розглядається, тому що він реалізований іншим розробником.

Додаток взаємодіє з сервером по https протоколу. Додаток відправляє на сервер post, get, upload запити. Сервер і додаток обмінюються інформацією у вигляді текстового формату json.

Переваги використання json формату:

1) Легко читається людьми (Рис. 3.2)

2) Більш відповідний для сериализации складних структур, ніж XML.

3) В серед iOS розробки існує безліч бібліотек, спрощують серіалізацію.

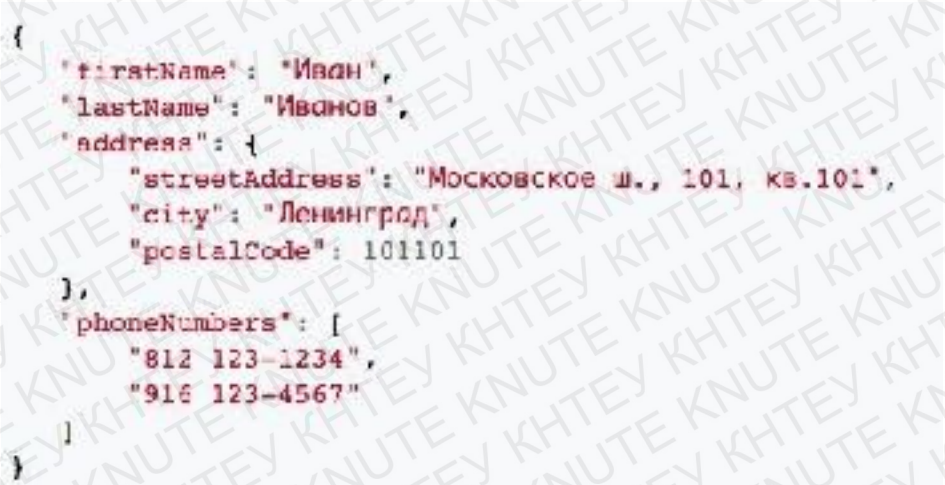


Рис. 3.2 – Приклад json відповіді.

Додаток такоємодіє з серверами соціальних мереж. Деякі соціальні мережі такі як Facebook, Google+ надають SDK вбудовуються в проект. SDK спрямовані на інтеграцію додатки з соціальною мережею, що істотно спрощує розробку. Але у більшості соціальних мереж немає свого SDK, тоді потрібно використовувати їх API і створювати свої веб запити.

3.2. Структура програми

Додаток починає працювати з виклику applications в AppDelegate. В цьому методі ми створимо AppCoordinator і виконаємо метод didFinishLaunchingWithOptions (Рис. 3.3).

```
func application(_ application: UIApplication,
                 didFinishLaunchingWithOptions launchOptions:
                    [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    let keyWindow = UIWindow(frame: UIScreen.main.bounds)
    window = keyWindow
    appCoordinator = AppCoordinator(window: keyWindow)
    appCoordinator?.start()
    return true
}
```

Рис. 3.3 – Запуск головного координатора.

AppCoordinator в свою чергу містить і управляє 3 координаторами (Рис. 3.4).

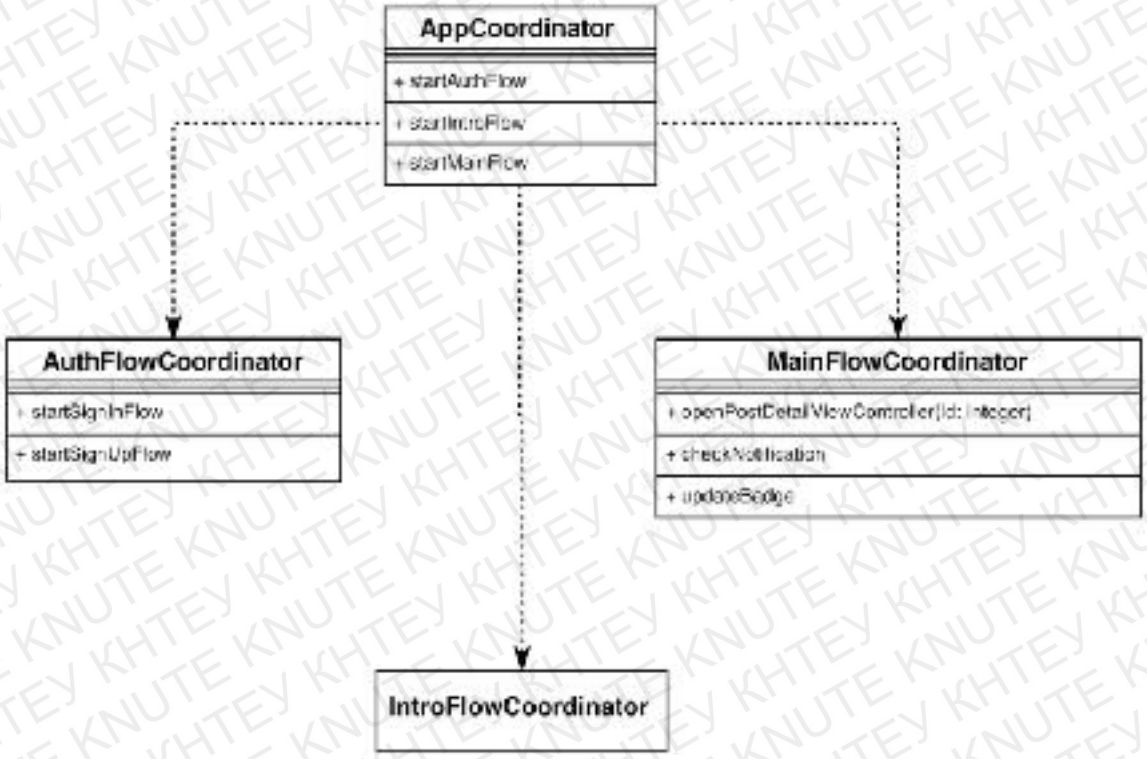


Рис. 3.4 – Діаграма класів, показує залежності між координаторами

.

Залежно від стану головний координатор вирішує, який показати.

Якщо користувач перший раз запустив додаток, то AppCoordinator виконає метод startIntroFlow. Це в свою чергу запустить роботу IntroFlowCoordinator, який відповідає за показ навчальних слайдів.

Якщо користувач не авторизований в системі, то AppCoordinator вико-

нає метод startAuthFlow, тим самим запустить роботу AuthFlowCoordinator. Цей координатор відповідає за авторизацію і реєстрацію користувача в додатку.

Якщо користувач вже авторизований в додатку, то AppCoordinator запустить MainFlowCoordinator, який відповідає за основний функціонал програми.

3.2.1. AuthFlowCoordinator

AuthFlowCoordinator - координатор програми відповідає за навігацію при авторизації і реєстрації в додатку. Розглянемо його структуру докладніше (Рис. 3.5).

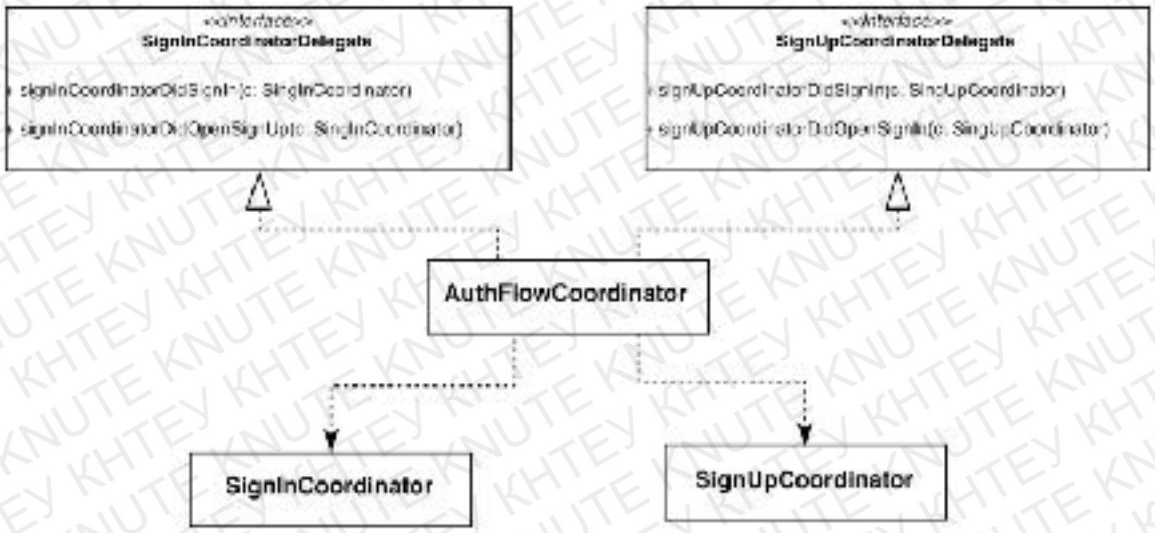


Рис. 3.5 – Діаграма класів, показує залежності AuthFlowCoordinator.

Як ми можемо помітити, AuthFlowCoordinator містить і управляє 2 координаторами, а також реалізує два інтерфейсу. Головний координатор при необхідності заміщає один координатор іншим.

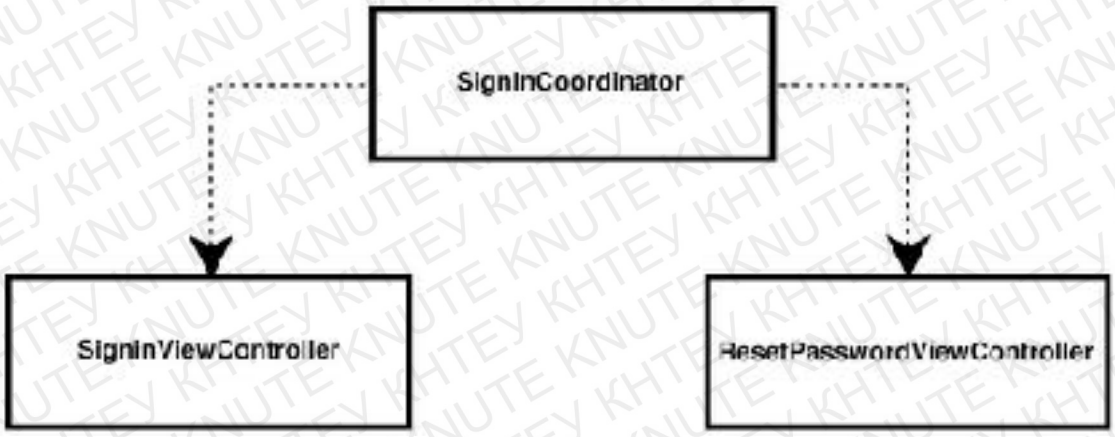


Рис. 3.6 – Діаграма класів, показує залежності SignInCoordinator.

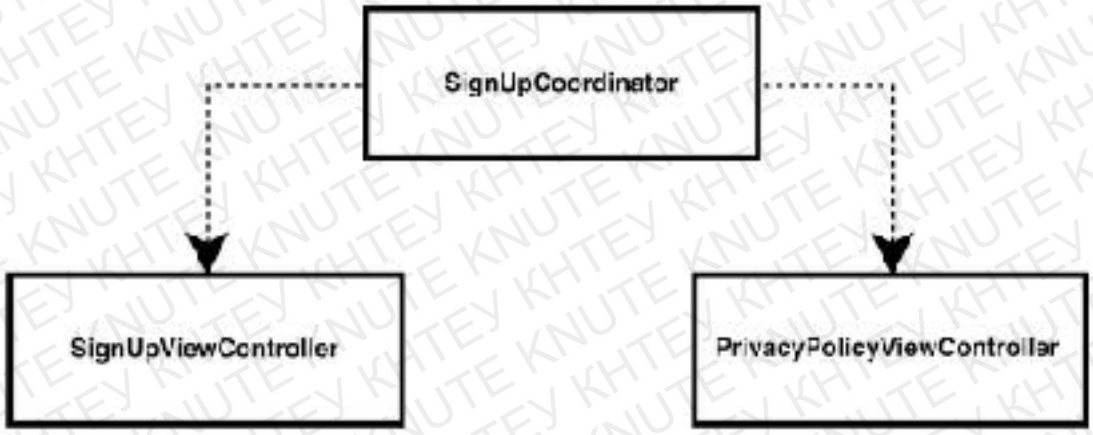


Рис. 3.7 – Діаграма класів, показує залежності SignUpCoordinator з контролерами.

ViewModel кожного контролера має посилання на свій координатор. Розглянемо кожен контролер докладніше:

1) SignInViewController: необхідний для реалізації самої авторизації.
2) ResetPasswordViewController: необхідний для відновлення пароля.
3) SignUpViewController: необхідний для самої регістрації в додатку.
4) PrivacyPolicyViewController**:** необхідний для перегляду політкики конфіденційності додатку.

Після успішної авторизації або реєстрації, відповідний координатор повідомляє про це AuthFlowCorrdinator, який в свою чергу повідомляє це головному координатору, які вже вирішує показувати чи головний функціонал додатка.

3.2.2. IntroFlowCoordinator

IntroFlowCoordinator - відповідає за навігацію між екранами в додатку, що допомагають новому користувачі з легкістю розібратися з основним функціоналом.

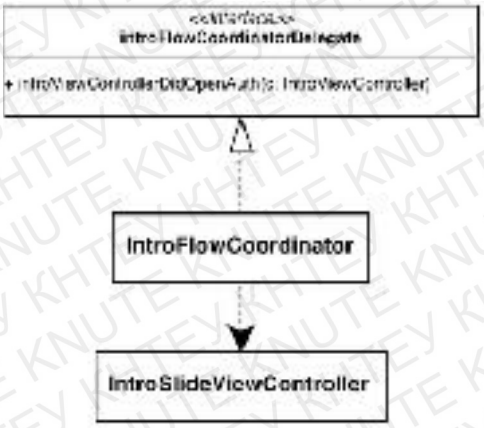


Рис. 3.8 – Діаграма класів, показує залежності SignUpCoordinator.

Координатор керує IntroSlideViewController і реалізує інтерфейс IntroFlowCoordinatorDelegate. IntroSlideViewController: перемикає view, реагуючи на жести користувача (Рис. 3.8). Можна переміщатися вліво і вправо. На останній view здійснюється перехід на координатор AuthFlowCoordinator.

3.2.3. MainFlowCoordinator

MainFlowCoordinator - відповідає за навігацію між основними екранами пріложенія.Координатор приймає кілька інтерфейсів і має метод openPostDetailViewController, який дозволяє переглядати пост прийшов в push повідомлення (Рис. 3.9). Незалежно на якому екрані ми будемо знаходитися, контролер

PostDetailViewController буде відображатися вище всіх в ієрархії контролерів.

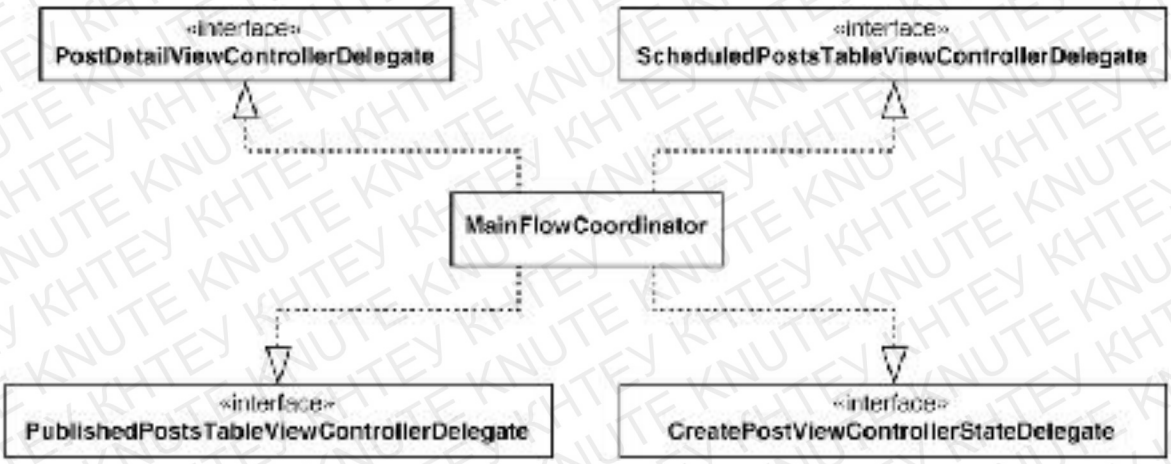


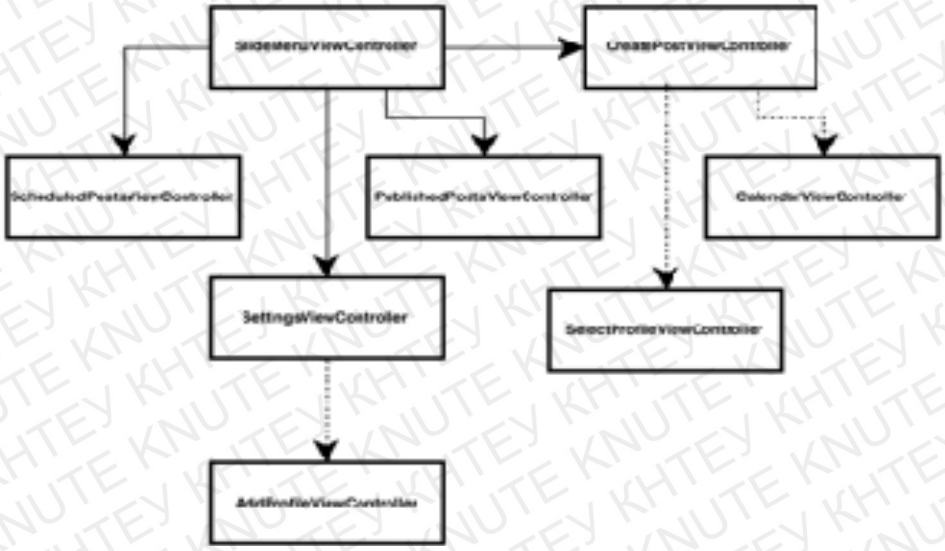
Рис. 3.9 – Діаграма класів, показывающаязависимости MainFlow-Coordinator.

MainFlowCoordinator взаємодіє з декількома контролерами, які обмінюються повідомленнями один з одним через делегати. Користувач управляє навігацією через висувний меню зліва, за це відповідає SlideMenuView-Controller (Рис. 3.10).

Рис. 3.10 – Діаграма класів, показує залежності SignUpCoordinator з котроллерами.

SlideMenuViewController містить і керує безліччю окремих контролерів (Рис. 3.11).

Рис. 3.11 – діаграма класів, що показує залежно SlideMenuView-



Controller.

Розглянемо кожен контролер докладніше:

- ScheduledPostsViewController: необхідний для відображення списку запланованих постів, відображається у вигляді таблиці, де кожна клітинка є постом.
- PublishedPostsViewController: необхідний для відображення списку опублікованих постів.
- CreatePostViewController: контролер дозволяє створювати нові пости.
- SelectProfileViewController: дозволяє вибрати один або кілька акаунтів соціальних мереж, що потрібно при створенні поста.
- CalendarViewController: контролер який дозволяє вибрати день і виставити час для посту, під час його створення.
- SettingsViewController: дозволяє змінювати профіль користувача аккаунта.
- AddProfileViewController: дозволяє додати новий обліковий запис для соціальних мереж.

3.3. Інтеграція соціальних мереж

Соціальні мережі має різні технології авторизації і публікації постів.

Потрібно навчиться взаємодіяти з різними API і SDK. Для інтеграції соціальних мереж, наведемо функціонал до загального класу SocialNetwork. Для кожної соціальної мережі створимо успадкований клас від SocialNetwork і реалізуємо кожен метод в залежності від технології (Рис. 3.12). Зберігати всі об'єкти будемо в упорядкованому масиві. Як тільки потрібно доступ до певної соціальної мережі, з масиву дістається заздалегідь сконфігурованих об'єкт.

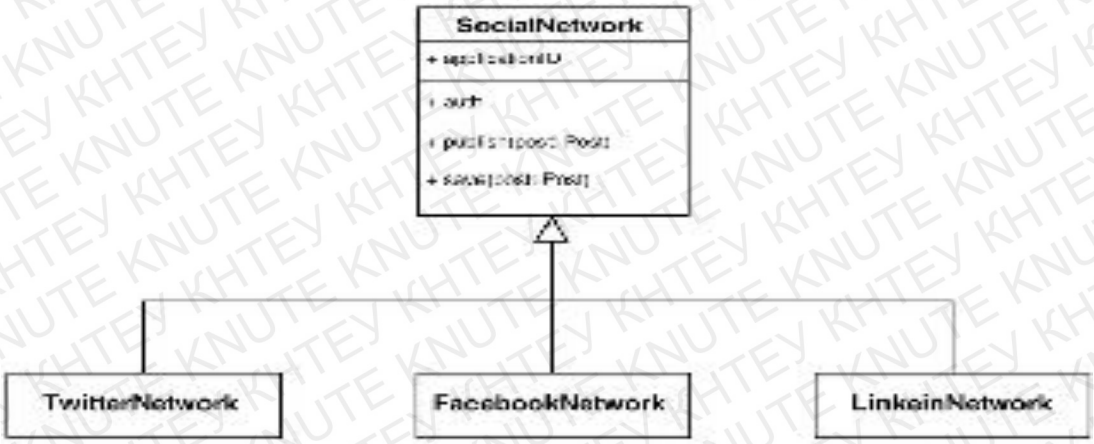


Рис. 3.12 – Діаграма класів соціальних мереж.

У соціальній мережі Instagram немає можливості публікації постів через API. Єдиний спосіб, це відправляти зміст посту заздалегідь встановленої програми Instagram і використовуючи його публікувати.

3.4. Тестування розробленого додатка

Тестування програми було вироблено на пристроях з різними характеристиками (Таблиця 1).

Таблиця 1: Тестові пристрої

Пристрій	Дозвіл	Діагональ	Версія iOS
iPhone 6s	1334 × 750	4.7”	13.4
iPhone 8	1334 × 750	4.7”	12.2
iPhone 8 Plus	1920 × 1080	5.5”	12.1
iPhone X	2436×1125	5.8”	12.5
iPhone XR	1792 × 828	6.1”	13.1
iPhone XS Max	2688 ×1242	6.5”	13.5
iPad Pro	2732 x 2080	12.9”	10

При тестуванні приділяється увага характеристики: діагональ екрана, дозвіл екрана, версія ОС iOS.

Під час процесу тестування були виявлені і своєчасно усунені такі помилки:

1. Некоректне відображення призначеного для користувача інтерфейсу: через великий діагоналі на пристроях iPad деякі елементи інтерфейсу розташовувалися в неправильному місці, а також розтягувалися.
2. Відмінності в версіях операційної системи вплинуло на push повідомлення, в 9 і 10 версіях iOS SDK надає різну обробку приходять пові-

домлень, що вимагало реалізації двох варіантів.

3. Некоректно відображення анімацій переходу між екранами.
4. Помилка при конвертації відео і фото.

3.5. Висновки до розділу 3

В третьому розділі було стоверннo зв'язок з сервером котрий буде зберігати данні про користувача. Створили навігацію по проекту між контролерами. Інетгрували дані до соціальних мереж та протестували додаток на коректність різних інтерфейсів на системі iOS.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У виконаній роботу:

1) Проаналізовано існуючі рішення
2) Вивчено особливості мобільної платформи iOS
3) Сформовано вимоги до розробляється з додатком
4) Вивчено взаємодію з різними API соціальних мереж
5) Спроектована архітектура додатки
6) Реалізовано додаток
7) Протестовано додаток

В роботі розроблено iOS-додаток дозволяє інтегрувати кілька соціальних мереж для перегляду, створення, редагування і публікації постів, тобто поставлена мета роботи була виконана. Крім того, в рамках даної роботи були проаналізовані технології взаємодії з соціальними мережами. В даний момент в додаток інтегровано 5 соціальних мереж. Однак реалізований простий приклад додавання нових мереж, що дозволить в майбутньому розширити функціональність.

Результати роботи - вивчена середовище розробки Xcode для пристроїв від компанії Apple; досліджена архітектура ОС iOS; проаналізовані API соціальних мереж; спроектована структура мобільного додатка; розроблено мобільний додаток, доступне через AppStore.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фаулер М., Архітектура корпоративних програмних додатків // М.Фаулер. — М.: Издательский дом "Вильяме", 2006. — 544 с.
2. iOS Developer Library URL: http://developer.apple.com/library/ios/navigation/ (Дата звернення: 10.11.2019).
3. Life Cycle [Електронний ресурс] // Apple Inc. — Електрон. дан. — [Б.м.], 2017 — URL: https://developer.apple.com/library/content/documentation/iPhone/ Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle.html
4. Thread [Електронний ресурс // Google Inc. — Електрон. дан.— [Б.м.], 2020 — URL: https://developer.apple.com/library/content/documentation /Cocoa/Conceptual/Multithreading/(Дата звернення: 24.03.2020)
5. ViewController [Електронний ресурс] // Apple Inc. — Електрон. дан. — [Б.м.], 2019 — URL: https://developer.apple.com/library/content/featuredarticles/ViewControll er/ (Дата звернення: 10.11.2019)
6. Гамма Э., Прийоми об'єктно-орієнтованого проектування // Э.Гамма, Р.Хелм, Р.Джонсон. — СпБ: Питер, 2001. — 368 с.
7. Introduction to MVVM [Електронний ресурс] – Електрон. дан. – Режим доступу: https://www.objc.io/issues/13-architecture/mvvm/, вільний.
8. An iOS Coordinator Pattern [Електронний ресурс] – Електрон. дан. – Режим доступу: https://will.townsend.io/2016/an-ios-coordinator-pattern, вільний.

ДОДАТКИ

Додаток А

```
import
UIKit

        enum AMLoginSignupViewMode {
            case login
            case signup
        }

        class ViewController: UIViewController {


            let animationDuration = 0.25
            var mode:AMLoginSignupViewMode = .signup


            //MARK: - background image constraints
            @IBOutlet weak var backImageLeftConstraint: NSLayoutConstraint!
            @IBOutlet weak var backImageBottomConstraint: NSLayoutConstraint!


            //MARK: - login views and constrains
            @IBOutlet weak var loginView: UIView!
            @IBOutlet weak var loginContentView: UIView!
            @IBOutlet weak var loginButton: UIButton!
            @IBOutlet weak var loginButtonVerticalCenterConstraint: NSLayoutConstraint!
            @IBOutlet weak var loginButtonTopConstraint: NSLayoutConstraint!
            @IBOutlet weak var loginWidthConstraint: NSLayoutConstraint!


            //MARK: - signup views and constrains
            @IBOutlet weak var signupView: UIView!
            @IBOutlet weak var signupContentView: UIView!
            @IBOutlet weak var signupButton: UIButton!
            @IBOutlet weak var signupButtonVerticalCenterConstraint: NSLayoutConstraint!
            @IBOutlet weak var signupButtonTopConstraint: NSLayoutConstraint!
```

```
    //MARK: - logo and constrains
    @IBOutlet weak var logoView: UIView!
    @IBOutlet weak var logoTopConstraint: NSLayoutConstraint!
    @IBOutlet weak var logoHeightConstraint: NSLayoutConstraint!
    @IBOutlet weak var logoBottomConstraint: NSLayoutConstraint!
    @IBOutlet weak var logoButtomInSingupConstraint: NSLayoutConstraint!
    @IBOutlet weak var logoCenterConstraint: NSLayoutConstraint!


    @IBOutlet weak var forgotPassTopConstraint: NSLayoutConstraint!
    @IBOutlet weak var socialsView: UIView!


    //MARK: - input views
    @IBOutlet weak var loginEmailInputView: AMInputView!
    @IBOutlet weak var loginPasswordInputView: AMInputView!
    @IBOutlet weak var signupEmailInputView: AMInputView!
    @IBOutlet weak var signupPasswordInputView: AMInputView!
    @IBOutlet weak var signupPasswordConfirmInputView: AMInputView!



    //MARK: - controller
    override func viewDidLoad() {
        super.viewDidLoad()

        // set view to login mode
        toggleViewMode(animated: false)

        //add keyboard notification
         NotificationCenter.default.addObserver(self, selector:
#selector(keyboarFrameChange(notification:)), name: .UIKeyboardWillChangeFrame,
object: nil)
    }

    override func didReceiveMemoryWarning() {
```

```
        super.didReceiveMemoryWarning()
    }



    //MARK: - button actions
    @IBAction func loginButtonTouchUpInside(_ sender: AnyObject) {

        if mode == .signup {
            toggleViewMode(animated: true)

        }else{

            //TODO: login by this data
            NSLog("Email:\(loginEmailInputView.textFieldView.text) Password:\
(loginPasswordInputView.textFieldView.text)")
        }
    }

    @IBAction func signupButtonTouchUpInside(_ sender: AnyObject) {

        if mode == .login {
            toggleViewMode(animated: true)
        }else{

            //TODO: signup by this data
            NSLog("Email:\(signupEmailInputView.textFieldView.text) Password:\
(signupPasswordInputView.textFieldView.text), PasswordConfirm:\
(signupPasswordConfirmInputView.textFieldView.text)")
        }
    }



    //MARK: - toggle view
    func toggleViewMode(animated:Bool){

        // toggle mode
```

```
        mode = mode == .login ? .signup:.login

        // set constraints changes
        backImageLeftConstraint.constant = mode == .login ? 0:-
self.view.frame.size.width

        loginWidthConstraint.isActive = mode == .signup ? true:false
        logoCenterConstraint.constant = (mode == .login ? -1:1) *
(loginWidthConstraint.multiplier * self.view.frame.size.width)/2
        loginButtonVerticalCenterConstraint.priority = mode == .login ? 300:900
        signupButtonVerticalCenterConstraint.priority = mode == .signup ?
300:900

        //animate
        self.view.endEditing(true)

        UIView.animate(withDuration:animated ? animationDuration:0) {

            //animate constraints
            self.view.layoutIfNeeded()

            //hide or show views
            self.loginContentView.alpha = self.mode == .login ? 1:0
            self.signupContentView.alpha = self.mode == .signup ? 1:0

            // rotate and scale login button
            let scaleLogin:CGFloat = self.mode == .login ? 1:0.4
            let rotateAngleLogin:CGFloat = self.mode == .login ? 0:CGFloat(-
M_PI_2)

            var transformLogin = CGAffineTransform(scaleX: scaleLogin, y:
scaleLogin)
            transformLogin = transformLogin.rotated(by: rotateAngleLogin)
            self.loginButton.transform = transformLogin
```

```
            // rotate and scale signup button
            let scaleSignup:CGFloat = self.mode == .signup ? 1:0.4
            let rotateAngleSignup:CGFloat = self.mode == .signup ? 0:CGFloat(-
M_PI_2)

            var transformSignup = CGAffineTransform(scaleX: scaleSignup, y:
scaleSignup)
            transformSignup = transformSignup.rotated(by: rotateAngleSignup)
            self.signupButton.transform = transformSignup
        }

    }

    //MARK: - keyboard
    func keyboarFrameChange(notification:NSNotification){

        let userInfo = notification.userInfo as! [String:AnyObject]

        // get top of keyboard in view
        let topOfKetboard = (userInfo[UIKeyboardFrameEndUserInfoKey] as!
NSValue).cgRectValue .origin.y


        // get animation curve for animate view like keyboard animation
        var animationDuration:TimeInterval = 0.25
        var animationCurve:UIViewAnimationCurve = .easeOut
        if let animDuration = userInfo[UIKeyboardAnimationDurationUserInfoKey]
as? NSNumber {
            animationDuration = animDuration.doubleValue
        }

        if let animCurve = userInfo[UIKeyboardAnimationCurveUserInfoKey] as?
NSNumber {
            animationCurve =  UIViewAnimationCurve.init(rawValue:
animCurve.intValue)!
        }
```

```
        // check keyboard is showing
        let keyboardShow = topOfKetboard != self.view.frame.size.height

        //hide logo in little devices
        let hideLogo = self.view.frame.size.height < 667

        // set constraints
        backImageBottomConstraint.constant = self.view.frame.size.height -
topOfKetboard

        logoTopConstraint.constant = keyboardShow ? (hideLogo ? 0:20):50
        logoHeightConstraint.constant = keyboardShow ? (hideLogo ? 0:40):60
        logoBottomConstraint.constant = keyboardShow ? 20:32
        logoButtomInSingupConstraint.constant = keyboardShow ? 20:32

        forgotPassTopConstraint.constant = keyboardShow ? 30:45

        loginButtonTopConstraint.constant = keyboardShow ? 25:30
        signupButtonTopConstraint.constant = keyboardShow ? 23:35

        loginButton.alpha = keyboardShow ? 1:0.7
        signupButton.alpha = keyboardShow ? 1:0.7


        // animate constraints changes
        UIView.beginAnimations(nil, context: nil)
        UIView.setAnimationDuration(animationDuration)
        UIView.setAnimationCurve(animationCurve)

        self.view.layoutIfNeeded()

        UIView.commitAnimations()

    }
```

```
    //MARK: - hide status bar in swift3

    override var prefersStatusBarHidden: Bool {
        return true
    }
}
```