Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія розробки соціальної мережі VinRiders»

Виконав: студент 2 курсу,
групи 1КН-19м
спеціальності 122 «Комп'ютерні науки»
Пантелюк Д. С.

Керівник: Яровий А.А.

Рецензент: Черноволик Г.О.

Вінниця
2020

ЗАТВЕРДЖУЮ
Завідувач кафедри____КН____
д.т.н., проф.. Яровий А.А.

_____
(підпис)
“____” _____2020 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.007.19.360

Магістранта групи 1КН-19м Пантелюка Дмитра Станіславовича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія розробки соціальної мережі VinRiders»

Вхідні дані: мова програмування – Swift5, повинна забезпечувати можливість маніпулювання даними і управління базами даних, сервер яких підтримує HTTP протокол; мова програмування для клієнтського додатку Swift 5; база даних типу RealtimeDatabase; кількість користувачів не менш ніж 100.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: фрагмент схеми алгоритму соціальної мережі,структурна схема соціальної мережі,схема аутентифікації користувача.
2. Текстова (пояснювальна записка): вступ, обґрунтування доцільності розробки інформаційної технології створення соціальної мережі VinRiders; моделювання інформаційної технології створення соціальної мережі VinRiders, структурна організація та особливості програмної реалізації соціальної мережі VinRiders, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Обґрунтування доцільності розробки			Розділ 1
2	Моделювання інформаційної технології організації соціальної мережі			Розділ 2
3	Структурна організація та особливості програмної реалізації розробки			Розділ 3
4	Підготовка економічної частини			Розділ 4
5	Апробація результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ д.т.н., проф., зав. каф. КН
(підпис) наук. ступінь, вчене звання (посада)
“____” _____20___ р. _____Яровий А. А._____
ініціали та прізвище

2. Економічна частина _____ канд. екон. наук, доц., доц. каф. ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
_____М. В. Бальзан _____
ініціали та прізвище
“____” _____20___ р.

Дата попереднього захисту роботи “____” _____20___ р.

Рецензент _____ _к.т.н., доц.кафедри ПЗ_____
(підпис) наук. ступінь, вчене звання (посада)
_____Черноволик Г.О.
ініціали та прізвище

Завдання видав
науковий керівник _____ д.т.н., проф., зав. каф. КН
(підпис) наук. ступінь, вчене звання (посада)
_____Яровий А. А._____
ініціали та прізвище
“____” _____20___ р.

Завдання отримав магістрант _____ _____Пантелюк Д. С._____
(підпис) ініціали та прізвище
“____” _____20___ р.

АНОТАЦІЯ

У магістерській кваліфікаційній роботі здійснено аналіз існуючих соціальних мереж, виділення основного функціоналу, порівняння технологій розробки соціальних мереж. Проведено дослідження соціальних мереж, їх актуальності та існуючих технологій для створення повноцінної соціальної мережі.

Здійснено аналіз предметної області розробки соціальної мережі під платформу iOS.

Здійснено проектування додатку соціальної мережі VinRiders. В результаті проектування було вибрано архітектуру та засоби реалізації додатку.

Здійснено розробку додатку соціальної мережі VinRider. На мові програмування Swift 5.0 для платформи iOS.

Здійснено тестування додатку соціальної мережі VinRiders. В результаті темстування доведено що розроблений додаток працює коректно.

Здійснено рохрахунок комерційного потерціалу розробки. В результаті розробки було розраховано витрати на розробку додатку, а також комерційний потерціал.

ANOTATION

In the master qualification robot, an analytical analysis of the basic social networks, visible basic function, advanced technology of the social networking machines. Provided a consistent social network, the current state of the art technology for the creation of a later social network.

Analyzed the subject area of the social network under the iOS platform.

Extra projected addition to the VinRiders social network. In the result, the projected bull vibrating architecture will be realized even further.

Add a little more VinRider social network. On my programming Swift 5.0 for iOS platforms.

Last tested with the social network VinRiders. In the result, the temptation brought to the crushing appendage is sent correctly.

Здійснено рохрахунок комерційного потерціалу розробки. In the result, the grapes grow in the vitreous of the grapevine, and the commercial pottery also grows.

ЗМІСТ

ВСТУП

У сучасному світі люди стали майже не розлучні зі своїми смартфонами. Смартфон поряд з нами увесь наш час. Починаючи зі сніданку закінчуючи переглядом фільму перед сном. Але найбільш популярними додатками є мессенджени, оскільки кожен з нас бажає залишатись на зв'язку. Авто та мото спільноти Вінничини також не залишаються осторонь, та користуються різними мессенджерами для розповсюдження інформації про події які будуть організовуватись. Саме тому вирішено розробити спеціальну соціальну мережу яка об'єднає увесь функціонал найпопулярніших мессенджерів та матиме назву VinRiders. Особливими компонентами для соціальних мереж є: чати з різними користувачами, завантажження фотоглафій користувачів, а також зручна розсилка новин або новий функціонал який не був реалізований у інших додатках. Нажаль готових реалізацій таких соціальних мереж не існує, і саме тому розробка додатку VinRiders актуально. Через це спільноти використовують різні соціальні мережі. Телеграм для ssh шифрування данних, Instagram для охвату великої аудиторії. Саме тому додаток VinRiders обєднає весь необхідний функціонал для авто мото спільноти Вінничини.

Зв'язок роботи з науковими програмами, планами, темами.

Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Розробка спеціалізованих засобів штучного інтелекту на основі інтелектуального аналізу даних та машинного навчання» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей спеціалізованої соціальної мережі VinRiders, а також пришвидшення роботи програмного продукту за рахунок переходу на Socket-підключення.

Для досягнення мети необхідно виконати такі завдання:

− аналіз та обґрунтування доцільності розробки інформаційної технології соціальної мережі VinRiders.

− аналіз та вибір методів та технологій розробки соціальної мережі VinRiders.

− розробити структурну організацію соціальної мережі VinRiders та здійснити її програмну реалізацію.

− здійснити тестування програмного продукту.

− економічно обґрунтувати доцільність розробки соціальної мережі VinRiders.

Об’єктом дослідження – процес розробки інформаційної технології соціальної мережі VinRiders під платформу iOS.

Предметом дослідження – програмні засоби розробки інформаційної технології соціальної мережі VinRiders.

Методи дослідження. У роботі використовуються такі методи дослідження: метод аналізу – для соціальних мереж; методи цифрової обробки інформації; методи взаємодії між клієнт-серверними додатками; методи роботи з базою даних; методи об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному:

– набула подальшого розвитку інформаційна технологія розробки соціальної мережі, що відрізняється від існуючих вдосконаленим способом отримання данних з серверу за допомогою відмови від https-протоколів та переходу на Socket-підключення, що забезпечило пришвидшення роботи та розширення функціональних можливостей соціальної мережі VinRiders.

Практичне значення одержаних результатів полягає у програмній реалізації інформаційної технології соціальної мережі VinRiders. Зокрема:

1. Розроблено програмний продукт, що реалізує соціальну мережу VinRiders. При цьому розширено функціонал програмного продукту, в тому числі розроблено новий функціонал мапи. В порівнянні з іншими системами-аналогами можливо розміщувати на мапі всіх користувачів, які надали доступ до

своїх данних геолокації. Також розроблено новий графічний інтерфейс маркеру на мапі, що забезпечило швидкий доступ до розширення функціоналу маркера. Удосконалено процедуру оновлення даних компонентів View за допомогою відмови від програмної реалізації Class та переходу на програмну реалізацію Struct, що забезпечело динамічну зміну компонентів View без необхідності перезавантаження контроллера.

2. Розроблено алгоритм роботи додатку без HTTP-запитів;

3. Розроблено алгоритм роботи клієнтської частини на базі технології Socket. При цьому удосконалено спосіб отримання данних з серверу за допомогою переходу на Socket-підключення, що забезпечило отримання нових даних з серверу без необхідності відправлення повторного https-запиту. Перехід на Socket підключення не впливає на конфеденційність даних оскільки протокол роботи Socket неможливо переглянути через перехоплювач даних.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, порівнянням результатів з відомими, та збіжністю результатів моделювання з результатами, що отримані під час впровадження розроблених програмних засобів, а саме додатку для платформи iOS та зареєстрованого Bundle ID додатку.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати досліджень було апробовано на XLVIII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2019)

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тези доповідей на міжнародній науковій конференції [1].

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ СОЦІАЛЬНОЇ МЕРЕЖІ VinRiders

1.1 Аналіз існуючих соціальних мереж

У сучасному світі існує більш ніж 1,5 тисячі соціальних мереж. Починаючи від невеликих місцевих соціальних мереж, закінчуючи компаніями гігантами такими як Facebook. Але найбільш популярні серед авто мото спільнот є такі соціальні мережі:

- Telegtam
- Instagram
- Viber

Розглянемо більш детально ці соціальні мережі.

Telegram - багатоплатформовий месенджер, що дозволяє обмінюватися повідомленнями і медіафайлів багатьох форматів . Клієнтські програми Telegram доступні для Android, iOS, Windows Phone, Windows, macOS і GNU / Linux . Користувачі можуть додавати і обмінюватися фотографіями, стікерами, голосовими повідомленнями, файлами будь-якого типу, а також робити аудіо- і відеодзвінки[1].

Кількість щомісячних активних користувачів сервісу, станом на кінець квітня 2020 року, становить понад 400 млн чоловік. У серпні 2017 року своєму Telegram-каналі Павло Дуров заявив, що кількість користувачів месенджера щодня збільшується більш ніж на 600 тисяч.

За даними дослідницького холдингу Romir на лютий 2018 року, в середньому користувачі Telegram в Росії витрачають на нього 10-11 хвилин в день. Найбільша частка користувачів припадає на росіян у віці 18-24 років. У Москві Telegram в два рази популярніше, ніж в інших містах Росії в цілому, особливо серед аудиторії від 35 до 44 років.

Проект фінансується Павлом Дуров в обсязі близько 13 млн доларів США щорічно [1].

Instagram - додаток для обміну фотографіями і відеозаписами з елементами соціальної мережі, що дозволяє знімати фотографії та відео, застосовувати до них фільтри, а також поширювати їх через свій сервіс і ряд інших соціальних мереж . Раніше Instagram дозволяв робити фотографії квадратної форми, як камери моментальної фотографії Polaroid, Kodak Instamatic і среднеформатниє камери 6 × 6 (більшість мобільних фотопріложеніе використовує співвідношення сторін 3: 2), але з 26 серпня 2015 року Instagram ввів можливість додавати фото і відео з ландшафтної і портретної орієнтаціями без обрізання до квадратної форми[2]..

Додаток сумісний зі специфікацією iPhone, iPad і iPod Touch на iOS 4.3 і вище, а також з телефонами на Android 2.2 і вище з підтримкою OpenGL ES 2. Воно поширюється через App Store і Google Play відповідно. 21 листопада 2013 роки з'явився Instagram Beta для Windows Phone 8 .

У квітні 2012 року Instagram був придбаний компанією Facebook за 300 млн доларів грошовими коштами і 23 млн акцій компанії, що в цілому склало $ 1 млрд .

За прогнозами експертів, Instagram за 2017 рік отримав від глобальної реклами близько $ 2,8 млрд.

На 2018 рік кількість зареєстрованих користувачів становить 1,1 млрд осіб [2].

Viber - додаток-месенджер, яке дозволяє відправляти повідомлення, здійснювати відео- і голосові VoIP-дзвінки через інтернет. Здійснювати дзвінки між користувачами з встановленим Viber безкоштовні (оплачується тільки інтернет-трафік за тарифом оператора зв'язку). Viber має можливість відправляти текстові, голосові та відео повідомлення, документи, зображення, відеозаписи та файли, а також в автономному режимі[3].

Для авторизації користувачів і пошуку контактів додаток використовує номер телефону і передає вміст телефонної адресної книги (імена і телефони всіх контактів) на сервери корпорації Viber Media S.à r.l., Люксембург. Вони ж збирають інформацію про здійснені дзвінки та переданих повідомленнях,

тривалості дзвінків, учасників дзвінків і чатів - в цілях поліпшення якості обслуговування і в інших цілях [3].

1.2. Методи та програмні засоби для створення соціальної мережі

Для розробки будь якого додатку для операційної системи iOS потрібно:

1. Середовище розрбки XCode
2. Фізичний пристрій під управлінням iOS
3. Аккаунт та сертифікат розробника Apple

Ці всі речі потрібні для створення додатку клієнту. Але у соціальній мережі має бути серверна частина. Для цієї задачі чуводо підходять 3 типи серверів.

- Firebase

  Зручний тип серверу з підтримкою технології Socket що допоможе пришвидшити відклик додатку, та зменшити затримки на отримання данних. Окрім цього підтримка Crahslitycs та Storage, що забезпечить зберігання даних не тільки у вигляді файлів з картинками а також у вигляді логів роботи та аналітики програми.

- Ajure

  Більш складніший тип серверу який надає змогу самостійно запрограмувати роботу серверної частини з налаштуванням усіх потрібних функцій серверу, але для програмування знадобиться знання Python.

- Сервер домашнього використання

Сервер який буде знаходитись у вашому домі, зібраний за вашими параметрами. Увесь функціонал потрібно налаштувати власноруч, але мова програмування може бути зручною для розробника.

1.3 Постановка задачі дослідження

Подальші задачі розробки інформаційної технології соціальної мережі VinRiders.

Список мінімального функціоналу:

- Вікно реєстрації;
- Вікно входу;
- Стрічка новин;
- Мапа з райдерами;
- Сторінка профілю;
- Додавання новин або івентів;
- Редагування профілю;

Список мінімального функціоналу додатку VinRiders. Після розробки усіх пункців та тестування додатку, статус соціальної мережі зміниться на MVP (Minimal Valuable Product).

Оскільки будь яка діяльність в соціальній мережі може містити заборонений зміст, було прийнято рішення при реєстрації нових користувачів обовязкове завантаження водійського посвідчення. Оскільки водійське посвідчення видають з 18 річного віку, ми можемо гарантувати що контент буде показаний тільки дорослим користувачам. Адміністрація додатку проти активності неповнолітніх участинів всередині додатку, а також на подіях які організовуються організацією.

Таж за допомогою завантаженгого водійського посвідчення, адміністрація може переконатись що заповнений профіль відповідає реальному користувачеві. А також у випадку неправомірних дій всередині додатку надати дійсну інформацію про певного правопонушника. Саме через це додаток не буде заборонений на території України та поліція міста Вінниця буде відноситись до учасниців соціальної мережі з повагою та розумінням. Водійське посвідчення буде доступне для перегляду тільки адміністрації додатку, тому користувачам не слід перейматись за конфеденційність персональних даних.

Ще одним обмеженням є програмне обмеження запитів на хвилину. Не більш ніж 10 запитів на хвилину допоможе уникнути атаку на сервер, що не зможе вивести його з ладу. Через підключення сокет технологій а також програмного обмеження в запитах на хвилину DDOS атака стає неможливою.

1.4 Висновок

Цей розділ аналізує та вивчає використання соціальних мереж, описує їх та цільову аудиторію для використання.

Наведено готові реалізації популярних соціальних мереж якими на данний момент користується авто мото спільнота.

Наведено основні програмні засоби для створення соціальних мереж. Показано взаємодію клієнт-сервер та описано основні мови програмування та фреймворки.

Описано апаратні обмеження додатку в кількості запитів та обмеження на вивантажження даних з сервера для стійкої та безперебійної роботи сенрвера.

Описано неможливий обман системи при реєстрації кристувача який не досяг 18 річного віку через обовязкову наявність водійського посвідчення. Після підтверження цього користувача адміністрацією він матиме доступ до додатку та зможе бути частиною спільноти.

Визначені завдання дослідження, описані вимоги до клієнта, сервера.

2. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ СОЦІАЛЬНОЇ МЕРЕЖІ VinRiders

2.1 Обґрунтування вибору засобів реалізації інформаційної технології створення соціальної мережі VinRiders

Будь яка соціальна мережа складається з двох основних модулів:

- Клієнт (Мобільний додаток, або веб сторінка)
- Сервер (Пристрій який обробляє запити клієнта та видає потрібний результат)

Спочатку розглянемо серверну частину додатку. Без необхідності реалізації домашнього серверу, можливо використовувати безкоштовні серверні модулі, такі як:

Firebase - американська компанія, постачальник хмарних послуг, заснована в 2011 році Ендрю Лі і Джеймсом Темпліном , і поглинена в 2014 році корпорацією Google [9].

Пройшла два раунди інвестицій: в травні 2012 року отримала $ 1,4 млн від Flybridge Capital Partners, Greylock Partners, NEA, в червні 2013 року залучила $ 5,6 мільйона від Union Square Ventures і Flybridge Capital Partners[9].

Основний сервіс - хмарна СУБД класу NoSQL, що дозволяє розробникам додатків зберігати і синхронізувати дані між декількома клієнтами [5] [6] [7]. Підтримані особливості інтеграції з додатками під операційні системи Android і iOS, реалізовано API для додатків на JavaScript, Java, Objective-C і Node.js, також можливо працювати безпосередньо з базою даних в стилі REST з ряду JavaScript-фреймворків, включаючи AngularJS, React, Vue.js, Ember.js і Backbone.js. Передбачено API для шифрування даних .

Серед інших послуг, що надавалися компанією - запущений 13 травня 2014 року хостинг для зберігання статичних файлів (таких як CSS, HTML, JavaScript), що забезпечує доставку через CDN і сервіс аутентифікації клієнта з

використанням коду тільки на стороні клієнта з підтримкою входу через Facebook, GitHub, Twitter і Google (Firebase Simple Login) [9].

Крім цього, компанією випущений під ліцензією MIT веб-редактор коду Firepad, що забезпечує одночасну спільну роботу декільком користувачам з одним документом, який став основою редакторів Stash Realtime Editor фірми Atlassian і Koding . Ще один вільний проект компанії - безкоштовний месенджер Firechat, також випущений під ліцензією MIT.

Azure DevOps Server - продукт корпорації Microsoft , що представляє собою комплексне рішення, що поєднує в собі систему управління версіями , збір даних, побудова звітів , відстеження статусів і змін по проекту і призначене для спільної роботи над проектами по розробці програмного забезпечення . Продукт доступний у вигляді окремого додатка, схожого за функціями з хмарним сервісом Azure DevOps Services (до 2019 року називався Visual Studio Team Services , VSTS)[10] .

Розглянуті серверні модулі є найбільш поширеними для програмування власних додатків.

Але сервіс Azure не надає власний код для реалізації модулів необхідних для зручної роботи)[10] .

Сервіс Firebase надає зручну роботу з сервером а також повний код що надає змогу працювати з базою даних без необхідності програмування серверної частини. Уся робота з сервером відбувається на клієнтській частині.

Окрім відсутності необхідності написання серверної частини, сервіс Firebase надає зручні модулі для роботи з іншими модулями, такими як Crashlitycs. Даний модуль надає змогу записувати LOG файли, що забезпечить перегляд роботи стабільності програми. У разі винекнення збою в роботі програми, при наступному завантаженні програми збій буде відображатись в списку аналітики. А також буде встановлений коєфіцієнт стабільності роботи додатку)[10] .

Розробка для операційної системи iOS це складний процес. Оскільки компанія Apple не надає доступ до своїх пристроїв з сторонніх систем. Це

зроблено для неможливості взлому телефону за допомогою стороннього коду іншими системами розробки.

Найбільш розповсюдженна IDE для розробки програм вважаэться Microsort Visual Studio. Однак для забезпечення надійності телефонів iPhone компанія Apple розробила систему яка не дозволяє розробляи додатки з інших IDE.

Саме тому iOS розробники не можуть вибрати середовище розробки більш зручне для них.

Для доступу до системних параметрів смартфону необхідно отримати сертифікат розробника Apple Developer Account. Окрім цього потрібно мати фізичний комп'ютер та девайс для тестування.

Тому середовищем розробки було вибрано IDE XCode

Xcode - інтегроване середовище розробки (IDE) програмного забезпечення для платформ macOS, iOS, watchOS і tvOS, розроблена корпорацією Apple. Перша версія випущена в 2003 році. Стабільні версії поширюються безкоштовно через Mac App Store. Зареєстровані розробники також мають доступ до бета-збірок через сайт Apple Developer [13].

Apple Developer Account - це особистий акаунт робробника, за допомогою якого у кожного додатку є автор та потрібні сертифікати розробки. Для реєстрації акаунту потрібно фізично довести що робробник це фізична людина та не видає себе за іншого, для цього потрібно надіслати копію паспорту або водійського посвідчення [11].

Обгрунтування вибору платформи реалізації.

Оскільки смартфони iPhone є найбільш популярними в світі і мільйони людей є їх фанатами, тому ця платформа розробки є найбільш популярною. Саме лаконічність дизайну, графіки, та інші не значні моменти притягують розробників до цієї платформ розробки.

Окрім того маючи необхідні інструменти для розробки для платформи iOS було вибрано саме її. Операційна система Android працює не стабільно а також

вона є в публічному доступі, само тому на смартфони під операційною системою Android існує велика кількість вірусних програм, а також постійні атаки зі сторони хакерів.

Саме тому було вибрано розробку додатку для операційної системи iOS.

2.2. Розробка алгоритму базової роботи інформаційної технології соціальної мережі VinRiders

Алгоритм роботи програми не лінійний, оскільки у ньому досить багато розгалуджень та звязків між контроллерами. Однак незмінною стартовою точкою залишається SplashViewController рисунок 2.1.

SplashViewController відповідає за логіку входу в додаток. У його логіці немає нічого складного. При запуску додатку він повинен визначити чи сессія активна та завантажити данні які повинні відобразитись. Якщо сессія не активна SplashViewController перенаправить нас на вікно входу, якщо активна то на головний екран.

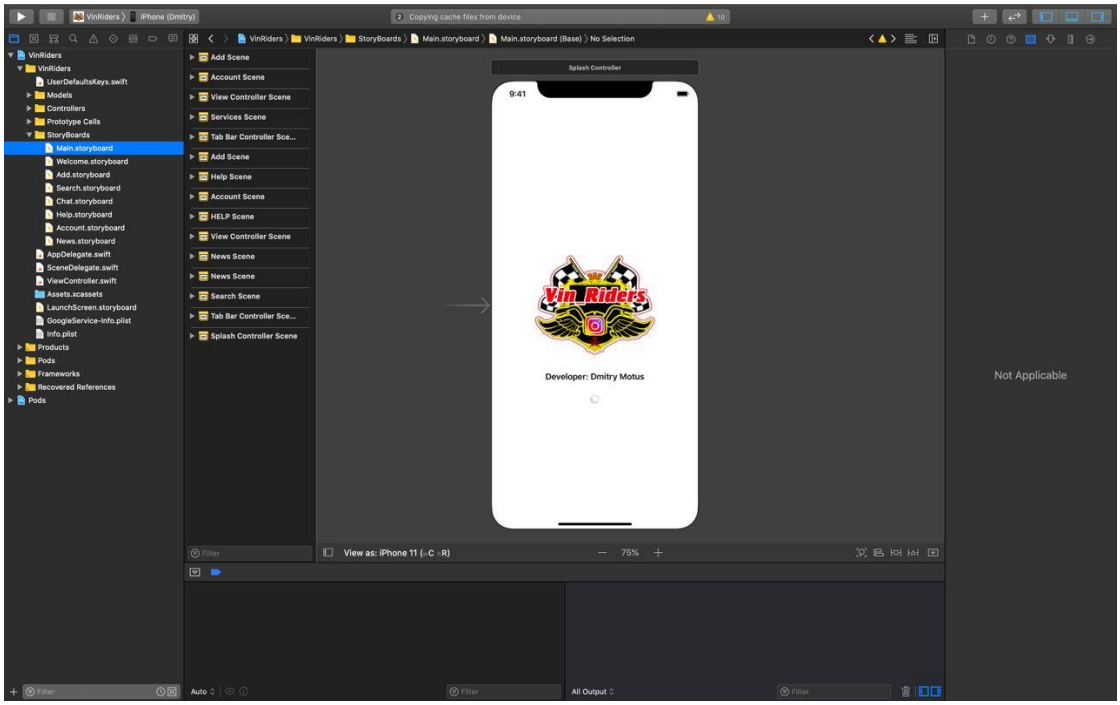

Рисунок 2.1 – Вигляд SplashViewController в Interface Builder

Після завантаження данних SplashViewController у випадку не закінченої сессії перейду у MainTabViewController рисунок 2.2, який містить у собі усі

ViewController для користування додатком. У додатку VinRiders пять активних контроллерів які відповідають за:

- Стрічка новин
- Мапа з активними райдерами
- Додавання новин
- Швидкий виклик допомоги
- Особистий акаунт

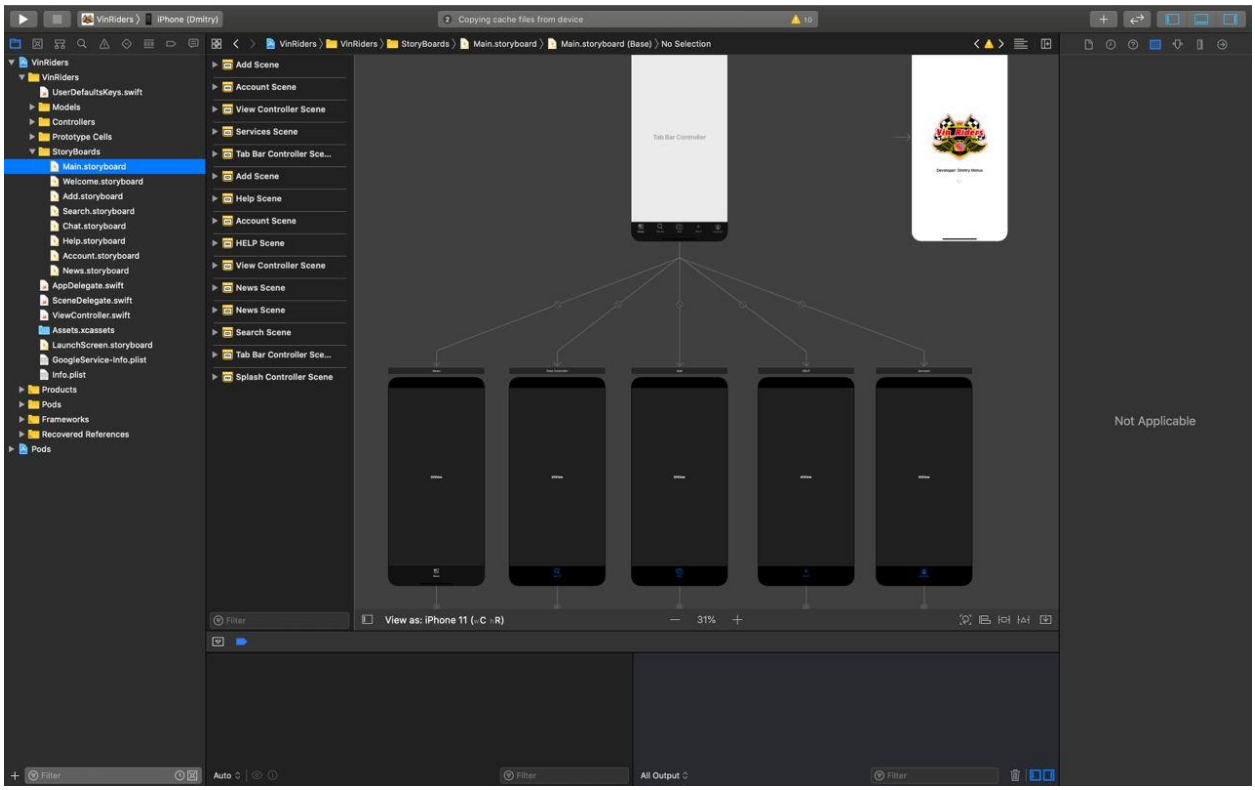

Рисунок 2.2 - Вигляд MainTabViewController в Interface Builder

У випадку якщо сессія буде завершеною, або користувач не буде зареєстрований. SplashViewController направить користувача на вікно входу. Вхід в додаток проходить за допомогою SMS підтвердження. Після вводу номеру телефону користувач отримує шестизначний код який він повинен ввести в наступному вікні. Якщо такий користувач існує, програма перенаправить його на головне вікно програми MainTabViewController, якщо такого користувача не буде зареєстровано, програма перенаправить його на

вікно з реєстрацією. У випадку не правильного введення коду програма не надасть токен сессії а значить користувач не зможе користуватись сервісом.

Варто зазначити що не всі зміни користувацького інтерфейсу стають доступними у Interface Builder рисунок 2.3. Через зміни зовнішнього вигляду в коді. Для прикладу кнопка Confirm в Interface Builder буде виглядати прямокутною, однак у додатку вона буде виглядати овальною.

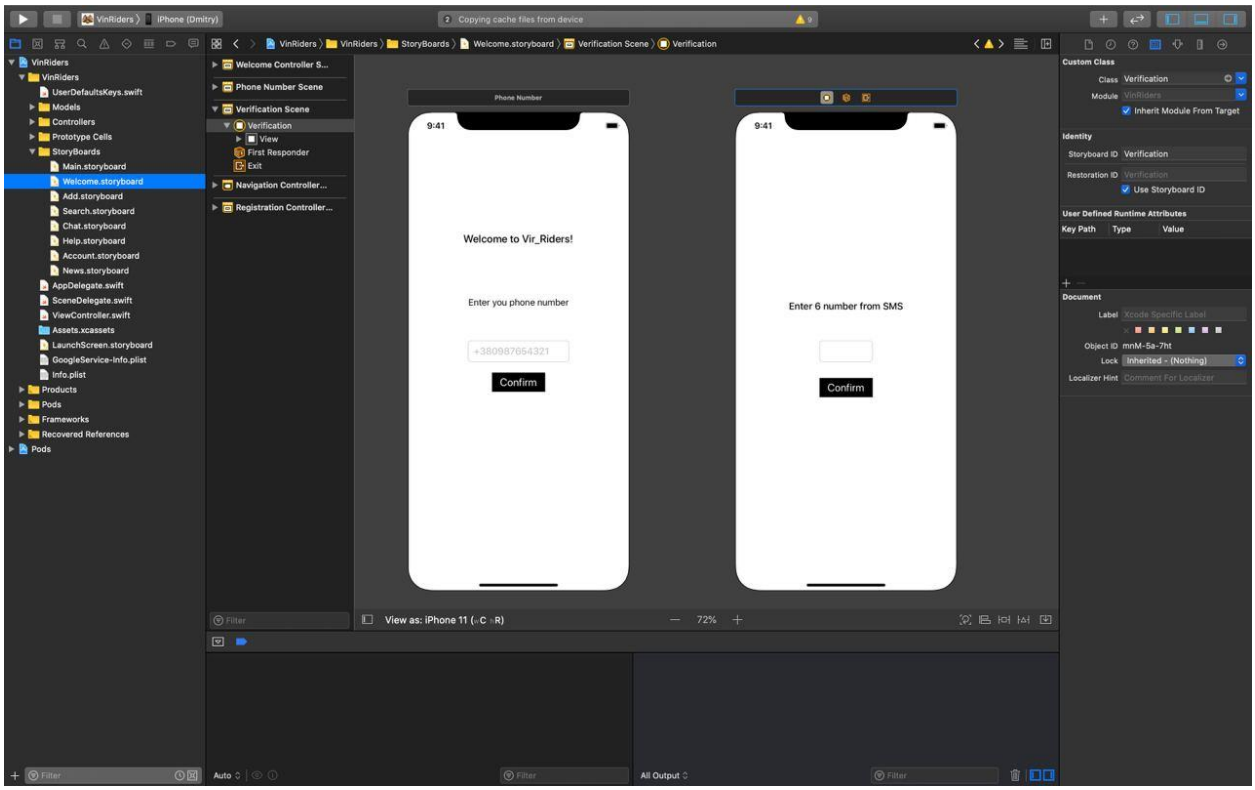

Рисунок 2.3 - Вигляд вікна входу в Interface Builder

Далі кожен контроллер буде опрацьовувати свій план задач. Від базової генерації користувацького інтерфейсу до отримання даних з серверу.

Будь який контроллер праює в таступному порядку:

1) Ініціалізація контроллеру в памяті.

   На цьому еттапі програма виділяє пам'ять на ініціалізацію контроррера а також на елементи в середині нього без його представлення, лише після повного завантаження контроллера программа перейде на інший контроллер представлення.

2) Запит на відкриття Socket підключення.

Цей етап відкриває прямий зв'язок з сервером ще до представлення контроллера користувачу. Таким чином контроллер який буде показаний вже матиме усі необхідні данні для показу їх на графічному інтерфейсі.

3) Генерація графічних елеменів

На цьому етапі программа розраховує усі необхідні розміри, позиції, кольори та інші властивості усіх графічних елементів.

4) Специфіка роботи контроллера

На цьому етапі кожен контроллера починає опрацьовувати свій план задач необхідні для роботи певного модуля. Наприклад для мапи це отримання маркерів та виставлення їх на мапі.

2.3 Розробка ієрархії додатку та звязку між контроллерами

Архітекрура контроллера є абстрактним занченням що характерезує порядок звязку між різними елементами контроллера. Саме тому будь яка архітектура є абстрактним розумінням побудови кожного контроллеру.

Архітектура описує звязки різних елементів контроллеру, на приклад:

MVC

Model – модель данних необхідна для показу на екрані.

View – модель зовнішнього вигляду контроллеру.

Controller – елемент який обробляє всі взаємодії від серверу, апаратної частини та користувача.

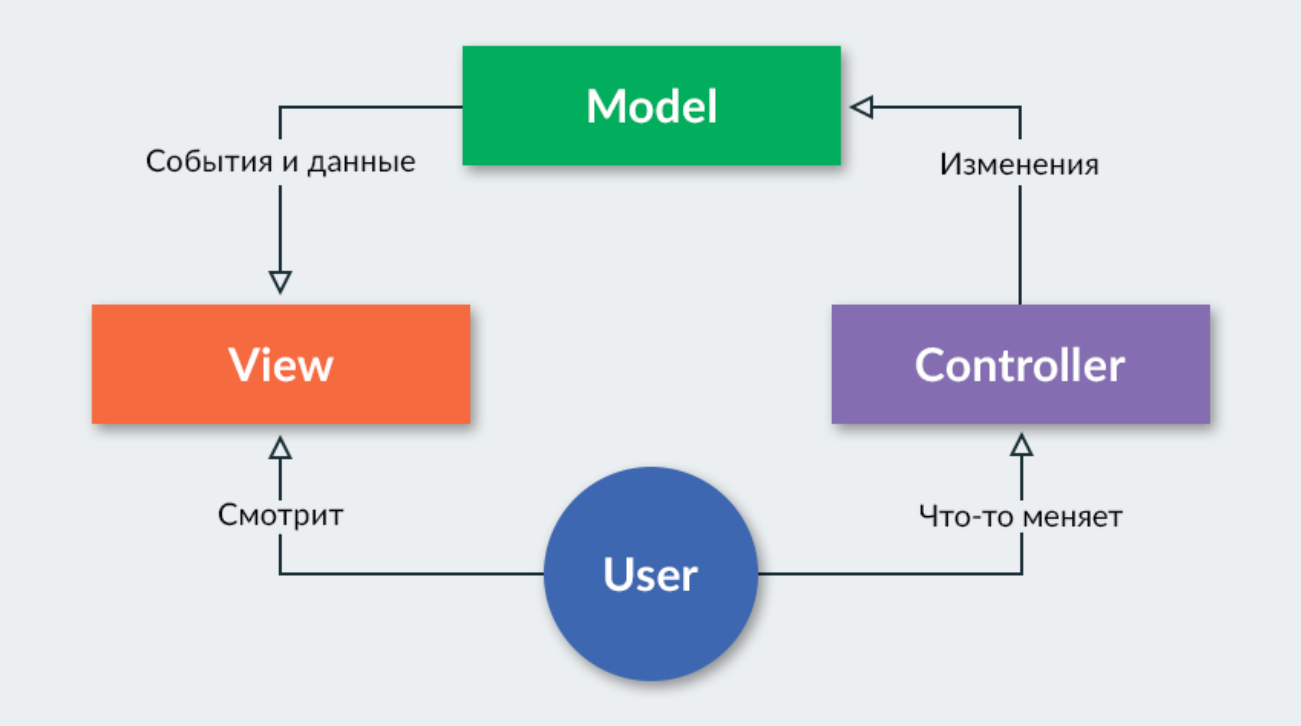

Рисунок 2.4 – Схема роботи архітекрути MVC

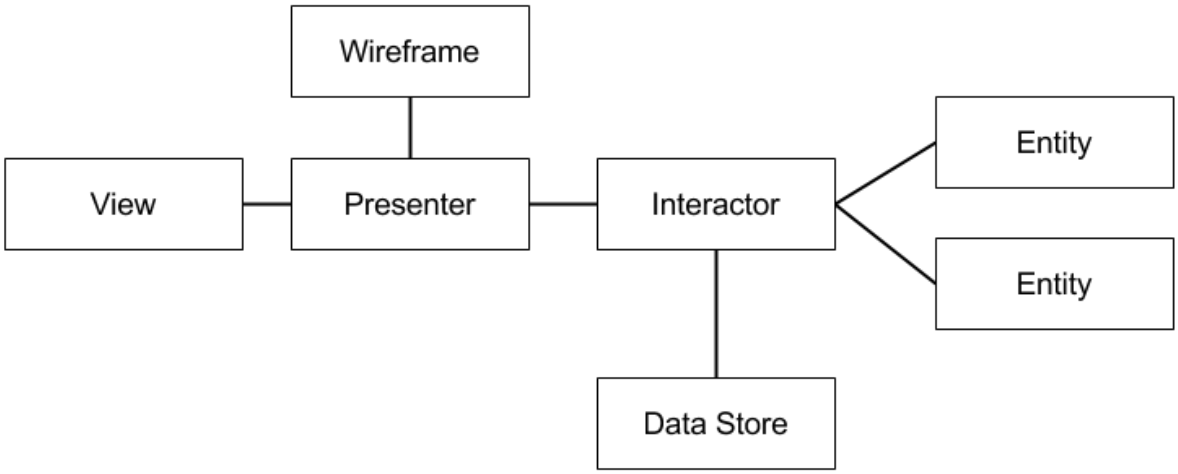

Рисунок 2.5 – Схема роботи архітекрути VIPER

Розглянувши схеми зображені на рисунках 2.4 та 2.5 ми можемо зрозуміти що компоненти залишаються не змінними, однах є нові елементи. Кожен елемент в схемі це окремий файл [6,7,8].

Принцип розробки SOLID говорить про те що файл має нести одну відповідальність. Тобто елемент який відповідає за зв'язок з сервером повинен займатись тільки звязком з сервером, файл відповідальний за модель данних повинен відповідати тільки за моделі данних. Саме через це різні архітекрути містять нові компоненти.

Архітектура MVC є найбільш розповсюдженною архітектурою для додатків. Через її простоту та зручних перехід між екранами представлення.

Архітектура MVC дозволяє нам розділити код програми на 3 частини: Модель (Model), Вид або Представлення (View) і Контролер (Controller). Вперше вона була описана в 1978 році, і призначалася для додатків з графічним інтерфейсом (віконцями і кнопками), але пізніше була адаптована і для веб-додатків.

Поділ на частини дозволяє спростити великий за обсягом код. Якщо код писати одним довгим скриптом, в ньому стає важко розібратися, і важко вносити зміни, не допустивши помилку.

MVC не прив'язана до якогось конкретного мови програмування, і не вимагає використання об'єктно-орієнтованого програмування або якийсь інший парадигми [6].

Поділ на частини тут не означає, що в коді повинно бути рівно 3 файлу (або 3 папки з файлами, або 3 класу) з назвами model, view і controller. MVC нічого не говорить нам з приводу того, як організовувати файли з кодом. На практиці модель часто займає основний обсяг додатки, і представлена у вигляді великого числа різнотипних класів - сутностей, сервісів, класів роботи з БД, і для кожного виду класів роблять окремі папки.

MVC застосовна до різних видів додатків - і до серверних веб-додатків, і до десктопних (клієнтським) додатків. Різниця між ними в тому, що в веб-

додатку програма отримує один запит від користувача, обробляє його, виводить результат (зазвичай це веб-сторінка) і завершується. Якщо прийде ще один запит, буде запущена нова, незалежна копія програми для його обробки. На відміну від веб-додатків, десктопні, мобільні додатки (а також написані на яваскрипт додатки, які працюють на сторінці браузера) довгоживучі. Вони обробляють багато запитів від користувача і оновлюють інформацію на екрані, не завершуючи.

Взаємодія між компонентами MVC реалізується трохи по-різному в серверних і в десктопних додатках через те, що веб-додаток - короткоживучі, обробляє один запит користувача і завершується, а десктопних програм обробляє багато запитів без перезапуску.

У серверних додатках зазвичай використовується "пасивна" модель, а в десктопних додатках - "активна" рисунок 2.6. Активна модель, на відміну від пасивної, дозволяє підписуватися і отримувати повідомлення про зміну в ній. У серверних додатках це не потрібно. Ось схема, яка зображує взаємодію компонентів:

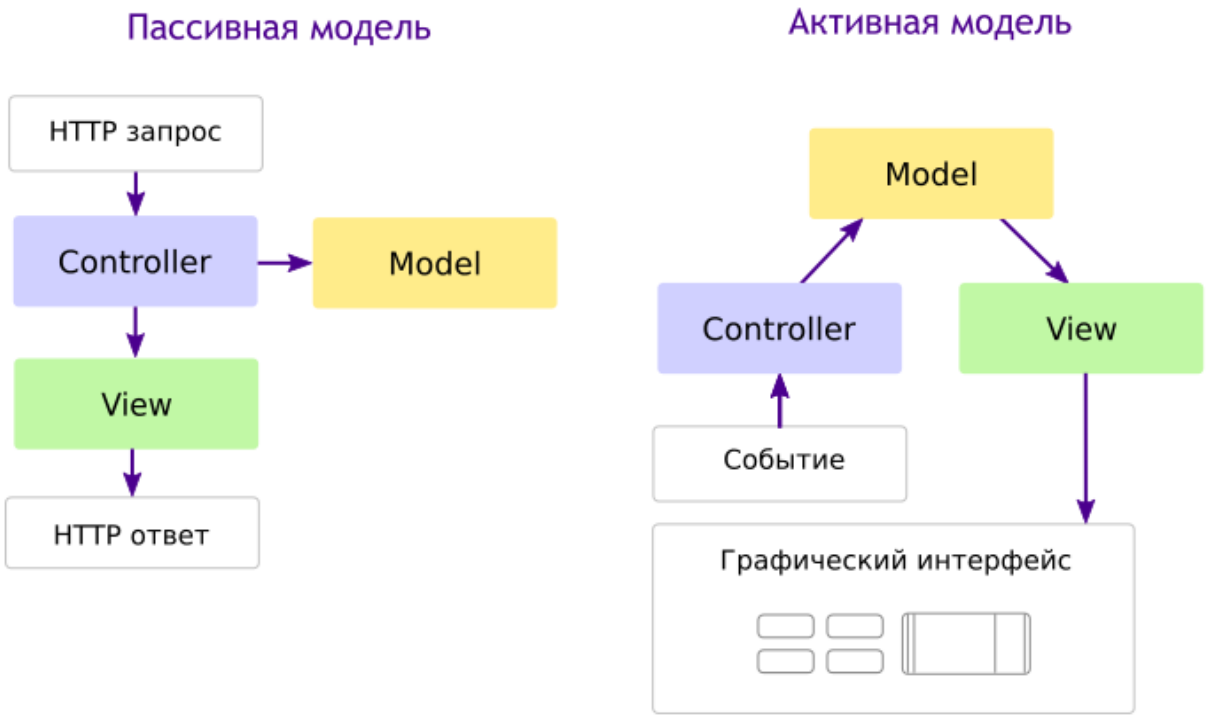

Рисунок 2.6 - Схема активної та пасивної моделі

У схемі з активною моделлю Вид підписується на зміни в Моделі. Потім, коли відбувається якась подія (наприклад, користувач натискає кнопку), викликається Контролер. Він дає Моделі команду на зміну даних. Модель повідомляє своїм передплатникам (в тому числі Виду), що дані змінилися, і Вид оновлює інтерфейс програми. Ми не будемо далі розбирати цей варіант MVC, про нього докладно написано в моєму уроці по MVC в JS-додатках.

Схема з активною моделлю нагадує кільце або цикл, так як вона розрахована на тривалу роботу. На відміну від неї, схема з пасивної моделлю зазвичай використовується в коротко-існуючих додатках.

У серверних додатках використовується схема з пасивної моделлю. Припустимо, користувач заходить на сторінку форуму. Його браузер відправляє HTTP запит на отримання сторінки зі списком повідомлень. При цьому запускається Контролер, який аналізує запит користувача і запитує у Моделі список повідомлень. Отримавши його, він викликає Вид і передає йому список, і той відображає його у вигляді веб-сторінки. Після цього скрипт завершується. Якщо користувач захоче додати повідомлення, він заповнить форму, відправить її, зголоситься Контролер, який відповідає за обробку даних цієї форми, прийме дані, попросить Модель перевірити і вставити в базу даних нове повідомлення, і потім віддасть HTTP відповідь з перенаправленням на сторінку перегляду повідомлень.

Однак через відмову від HTTP запитів, та перехід на Socket підключення обробка результатів змін не відповідає стандартам архітектури MVC, через це будь які зміни на сервері будуть викликати фенкцію яка буде самостійно викликати перезавантаження структур з даними. Через функцію нотифікацію не потрібно буде перезавантажувати контроллер.

Озброївшись цими знаннями, спробуємо написати найпростіший веб-додаток з використанням MVC.

2.4 Висновок

У данному розділі було розглянуто варіанти вибору серверу, архітектури, та засобів розробки соціальної мережі.

Обрано архітектуру розробки MVC, розглянуто різні типи серверної частини та проаналізовано їх функціонал.

Розроблено інтерфейс користувача для всіх екранів та підключена логіка роботи всіх елементів, які відображаються в інтерфейсі.

Розроблена ієрархія контроллерів представлення та структура показу екранів. Сформовано функціонал для користування користувачами, а також розроблено унікальний функціонал який не був реалізований у інших додатках.

Розроблено базовий алгоритм роботи програми, з моменту запуску програми. Додаток буде написаний на мові програмування Swift у середовищі розробки XCode.

3. СТРУКТУРНА ОРГАНІЗАЦІЯ ТА ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СОЦІАЛЬНОЇ МЕРЕЖІ VinRiders

3.1 Реалізація основної сторінки користувача

Перед початком реалізації модулів потрібно розробити ієрархію переходів між контроллерами. Ієрархія переходів забезпечує план перехов між різними контроллерами та показує з яких контроллерів ми можемо переходити до інших модулів рисунок 3.1.

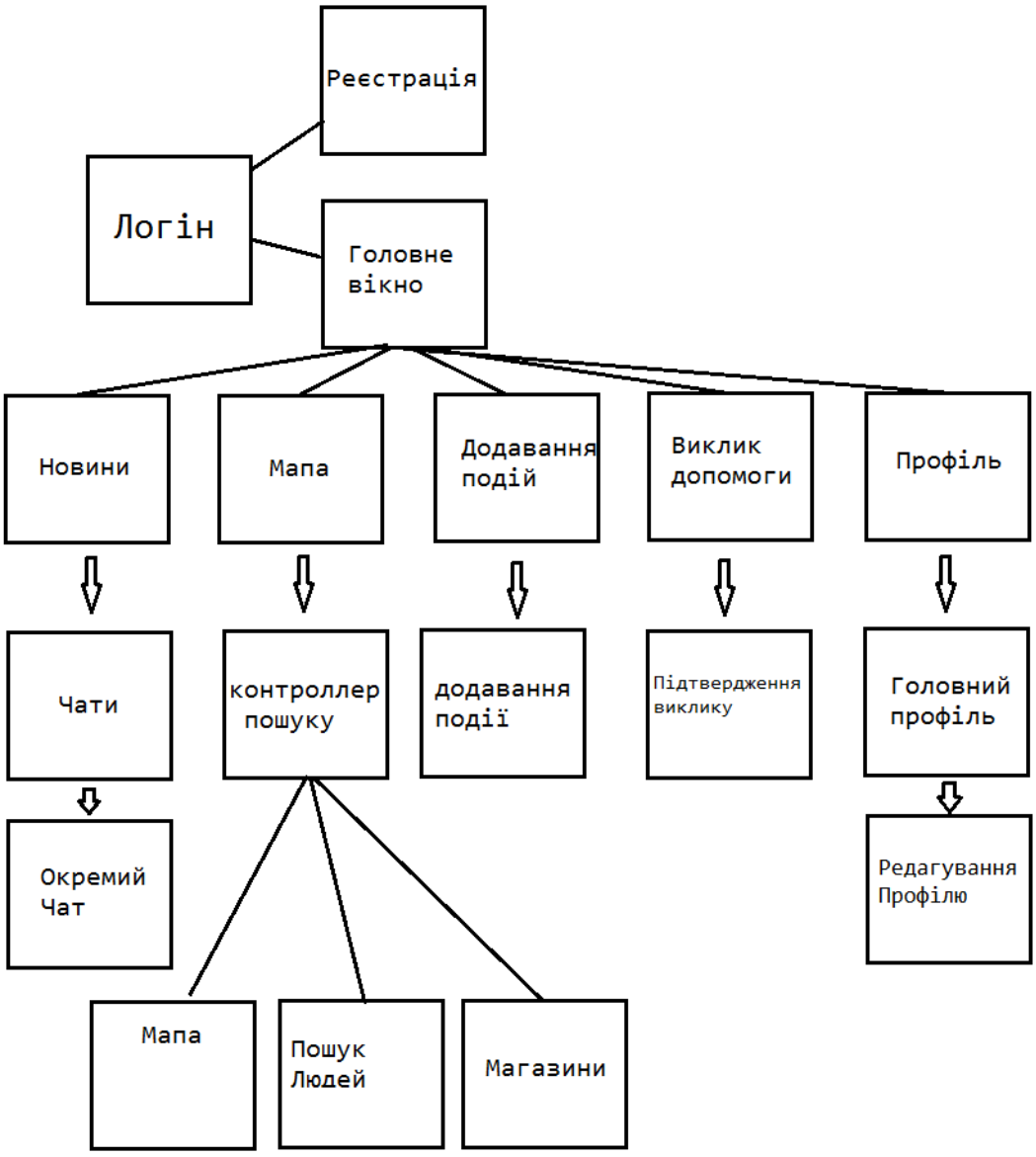

Рисунок 3.1 – Мапа екранів та переходів

Кожний екран це окремий класс, тому можемо вважати що на рисунку 3.1 зображено діаграму класів. Однак в мобільній розробкі розробити діаграму класів досить важко через велику кількість звязків між класами та підкласами.

Для прикладу один клас контроллера може використовувати близько ста різних класів для обробки подій, анімацій та інше.

Головна сторінка користувача повинна показувати інформацію про певного користувача. Але будь який користувач може перейти на сторінку іншого користувача. Саме тому портнібно розробити два різних дизайни сторінки.

Перший на випадок якщо користувач переглядає свою сторінку, а другий якщо він переглядає сторінку іншого користувача.

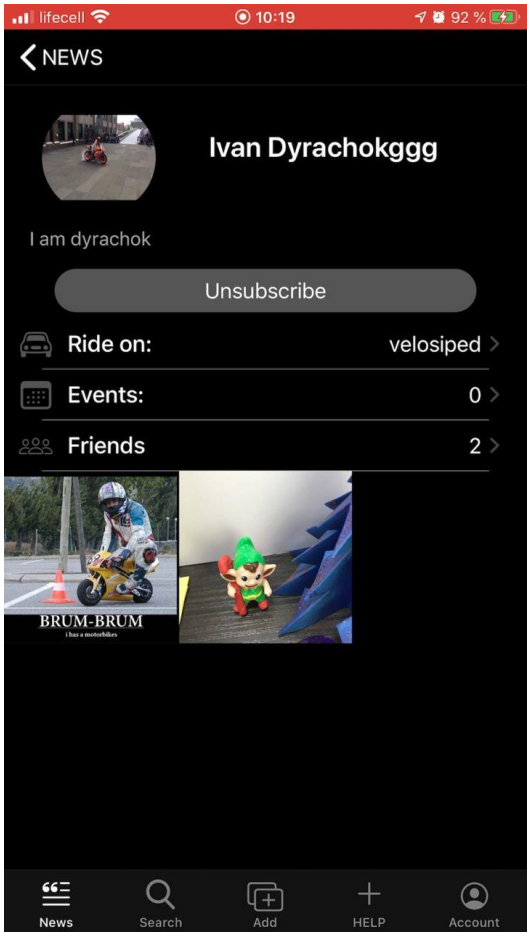

Рисунок 3.2 – Головна сторінка іншого користувача

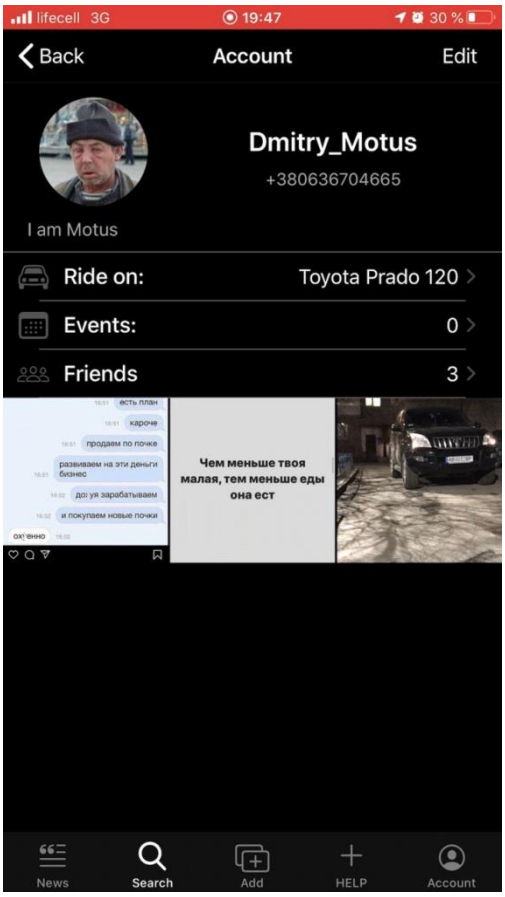

Рисунок 3.3 – Особиста сторінка користувача

Основний графічний елемент це UITableView, представлення в табличному вигляді, де кожний рядок таблиці це окремий тип UITableViewCell елементу.

UITableViewCell – це окремий контроллер з особистим дизайном та логікою роботи певного рядка

На екрані користувача відображаються такі елементи:

- Фотографія, Ім’я, Номер телефону та опис
- Назва транспорту яким керує користувач
- Кількість подій які організовував користувач
- Кількість друзів
- Фотографії які користувач завантажив на свою сторінку

На екрані інших користувачів також відображається кнопка підписки на них, або ж кнопка відписки якщо ви підписані рисунок 3.1

Логіка програми передбачає використання одного і того ж контроллера для цих двох екранів. При переході в вкладку Account програма одразу розуміє що перехід відбувається в особистий аккаунт та не показує кнопку додавання в друзі. Однак при переході у цей контрллер з іншого місця в додатку ми перевіряєм чи ID користувача не співпадає з ID особисто користувача. Якщо ID співпадає з ID сообобистого користувача програма не добавить кнопку підписатись на екран, якщо ID не співпадає контроллер перевірить чи користувачі є в списку друзів один одного, в результаті цього кнопка підписки буде змінювати свій колір.

Дана функція отримує данні про користувача та форматує її в модель представлення.

```
func getData(){
 let userID = Auth.auth().currentUser?.uid
ref.child("users").child(userID!).observeSingleEvent(of: .value, with: { (snapshot) in
// Get user value
let value = snapshot.value as? NSDictionary
 let rideOn = value?["rideOn"] as? String ?? ""
let events = value?["events"] as? Int ?? 0
let friends = value?["friends"] as? Int ?? 0
 let descryption = value?["descryption"] as? String ?? ""
let avatarPath = value?["avatarPath"] as? String ?? ""
let photosDictionaryArray = value?["photos"] as? [NSDictionary]
let photosCount = photosDictionaryArray?.count ?? 0
self.photoIndex = photosCount
if photosCount != 0{
```

```
for i in 0...photosCount-1{

 let photoObject = photosDictionaryArray?[i]
self.photos.append(photosStruct(photo: photoObject?["path"] as! String, descryption:
photoObject?["descryption"] as! String, sortId: i))
self.arrayPhotosData.append(photoObject?["path"] as! String)

self.photos.sort { (photos1, photos2) -> Bool in return photos1.sortId >
photos2.sortId } } } globalUser.avatarPath = avatarPath
self.tuples.append(accountData(type: "rideOn", data: rideOn ))
self.tuples.append(accountData(type: "events", data: String(events)))
self.tuples.append(accountData(type: "friends", data: String(friends)))
self.tuples.append(accountData(type: "descryption", data: String(descryption)))
self.refreshControl.endRefreshing() self.accountTableView.reloadData() }) { (error) in
print(error.localizedDescription) } }
```

3.2 Реалізація сторінки з новинами

Контроллер стрічки з новинами передбачає елемент UITableView з однотипними рядками.

Контроллер працює в наступному режимі:

- Запит на новини кожного з друзів
- Порівняння з минулими новинами на наявність актуальних
- Завантаження актуальних новин
- Форматування данних в модель
- Сортування новин по даті
- Перезавантаження таблиці з новинами.

Для розуміння сортування новин було обрано варіант сортування за датою, тому що похибка складає 1 секунду. Дата відліку починається з 01.01.1970 року з інтервалом в одну секунду.

Рисунок 3.4 – Приклад стрічки новин

Таким чином контроллер спочатку буде завжди показувати новини від друзів і тільки після того як новини друзів закінчаться контроллер буде показувати новини будь яких користувачів. Псевдовипадковим чином показуючи різних користувачів. Псевдовипадковим чином тому що комп’ютер не здатний дійно вибрати випадкове число, тому що програма обробляє випадкові числа за певними алгоритмами.

Дана функцыя забезпечує отримення новин для відображення їх на стрічкі новин

```
func getPostData(allCount:Int){

let ref: DatabaseReference! = Database.database().reference()
```

```
for i in 0...allCount-1{ ref.child("posts").child(String(i)).observeSingleEvent(of:
.value, with: { (snapshot) in

let user = snapshot.value as? NSDictionary

let author = user?["author"] as? String ?? ""

let descryption = user?["descryption"] as? String ?? ""

 let type = user?["type"] as? String ?? ""

let authorID = user?["authorID"] as? String ?? ""

 let pathImage = user?["path"] as? String ??
"https://ih1.redbubble.net/image.492975046.7150/flat,1000x1000,075,f.u1.jpg"

let pathAvatarImage = user?["avatarPath"] as? String ??
"https://ih1.redbubble.net/image.492975046.7150/flat,1000x1000,075,f.u1.jpg"
self.posts.append(PostModel(indexForSort: i, author: author, authorID: authorID,
descryption: descryption, type: type, image: pathImage,avatarPath:
pathAvatarImage))

self.posts.sort { (post1, post2) -> Bool in return post1.indexForSort! >
post2.indexForSort! }

if i == self.allCount-1{ self.refreshControl.endRefreshing() }
self.tableView.reloadData() }) } }
```

3.3 Реалізація мапи з водіями

Контролер відповідає за розширений функціонал. Окрім мапи контролер також вміщає в себе магазини та пошук інших користувачів на окремих UIChildViewController елементах, які не завантажуються в пам'ять до моменту їх виклику.

Контроллер який відповідає за мапу перед початком роботи повинен отримати доступ до геопозиції від користувача. І тільки після цього контроллер почне свою роботу. Оновлення геопозиції відбувається один раз в 15 секунд.

Після отримання даних геопозиції на карті з’явиться маркер з місцеположенням користувача. А також маркери тих користувачів які зараз онлайн.

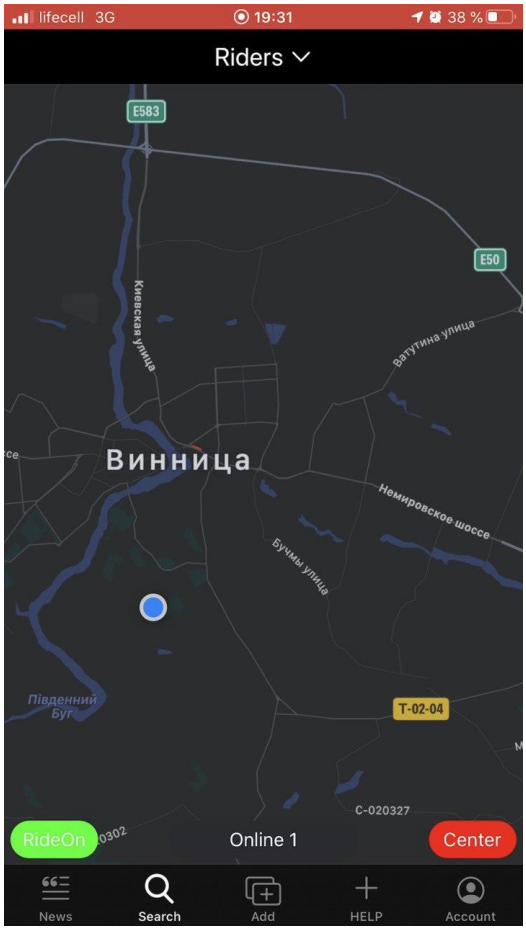

Рисунок 3.5 – Мапа з водіями

При натисканні кнопки Riders відображається меню яке запропонує перейти у інші контроллери. При натисканні кнопки RideOn користувач стане «Видимим» на карті іншим користувачем з активною позначкою де він зараз перебуває.

У випадку якщо сессію роботи програми буде припинено програма автоматично перейде у статус OffLine і відмінить показ місця перебування

користувача на карті. Окрім цього у випадку якщо користувач забув вимкнути свій статус та не заходить в додаток більше 1 години програма автоматично знімає статус та приховає місцеперебування.

В нижньому меню в центрі перебуває спикок кількості активних користувачів. Натиснувши на кнопку ми побачимо список користувачів.

В правій стороні нижнього меню перебуває кнопка центрування мапи по вашій позиції. Мапа буде автоматично переміщутись за вашою гепопозицією або буде залишатись статично.

Натиснувши на маркер активного користувача відображене меню покаже хто саме та на якому транспорті перебуває. І запропонує варіант написати повідомлення цьому користувачеві.

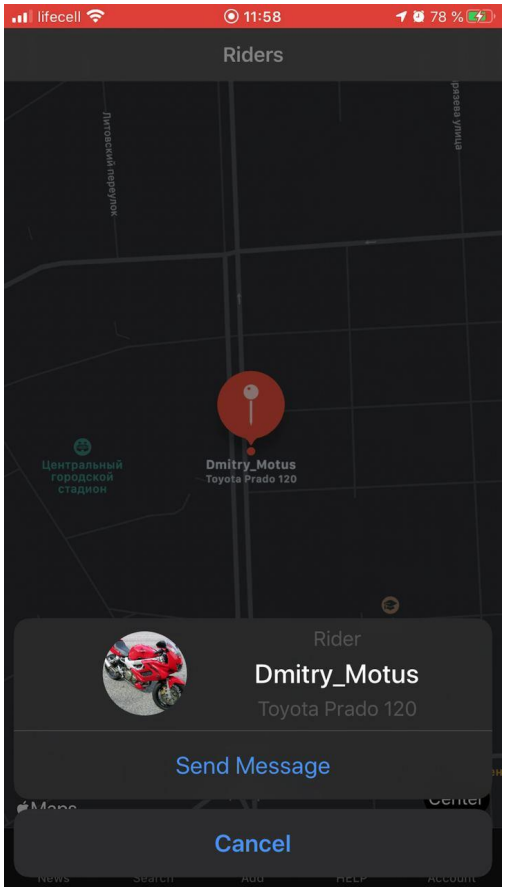

Рисунок 3.6 – Меню маркера

Дана функція забезпечує отримання локації та розставлення маркерів на мапі.

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) { manager.allowsBackgroundLocationUpdates = true for location in locations {

let coordinate = location.coordinate coordinates.lon = coordinate.longitude coordinates.lat = coordinate.latitude print("latitude \(coordinate.latitude), longitude \(coordinate.longitude)\n")

 let ref: DatabaseReference! = Database.database().reference()

 let userID = Auth.auth().currentUser?.uid ref.child("users/\(userID!)/coordinates/lat").setValue(coordinates.lat) ref.child("users/\(userID!)/coordinates/lon").setValue(coordinates.lon)

let annotation = MKPointAnnotation() mapView.removeAnnotation(myOldLocation) annotation.coordinate = CLLocationCoordinate2D(latitude: coordinates.lat, longitude: coordinates.lon) annotation.title = globalUser.userName annotation.subtitle = globalUser.rideOn mapView.addAnnotation(annotation) myOldLocation = annotation

if centerMapFlag{ centerMap() } getDataAnnotation() checkSleepTime()

} }
```

3.4 Тестування інформаційної технології створення соціальної мережі VinRiders та аналіз результатів

Тестування відбувається в 3 етапи:

1) Тестування завантаження данних.
2) Тестування виняткових ситуацій.
3) Тестування графічних елементів та орфографії.

Під час тестування завантаження данних перевіряються всі екрани які не залишаються статичними а мають виконувати запити. Перевірка має показати що завантаження данних відбувається коректно при будь яких обставинах.

Під час тестування виняткових ситуацій перевіряються моменти коли інформація повинна бути обовязкова але її не було завантаженно або вона відсутня. Для запобігання цього було встановлені безпечні оператори розвертання.

Під час тестування графічних елементів переглядаються усі анімації в будь яких контроллерах представлення, анімації повинні бути без ривків, різних швидкостях рухів елементів, та анімацій показу графічних елементів.

Під час тестування було виявлено проблеми з завантаженням данних при зміні інтернет підключення. Таким чином не всі данні були завантаженні та контроллер залишався пустим.

Допомогу в тестуванні додатку надала команда MotusDrasgRacing, компанда складається з 6 чоловік які займаються авто перегонами в Вінниці та Вінницькій області. Тестування відбувалось на протязі 14 днів.

Під час тестування команда кативно складала враження про додаток та надавала зворотній зв'язок. Таким чином були удосконалені певні графічні елементи для покращення зручності користування.

Після відгуку тестуваньників було роброблено систему, яка слідкує за станом інтернет-підключення, якщо інтернет-підключення змінюється, а запит ще виконується, система знищує не виконаний запит і виконує його щераз на новому типі інтернет підключення. Саме це підвищило стабільність систем завантаження даних без втрати даних які не встигли завантажитись.

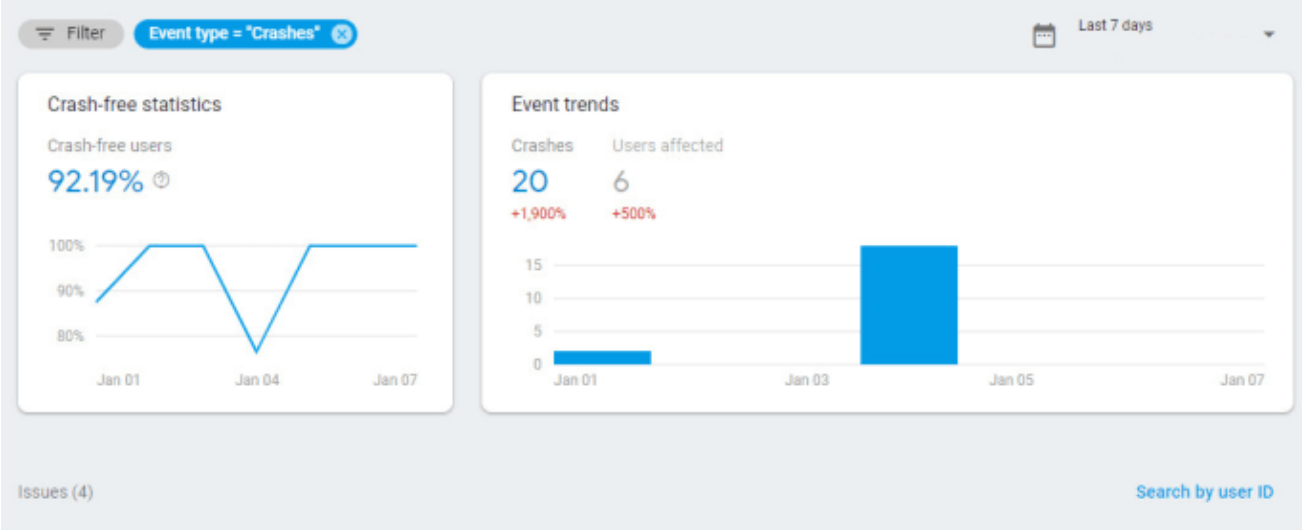

Рисунок 3.7 – Результати збоїв

Використання сервісу Firebase дозволило зберегти статистику збоїв в роботі програми. Як показано на рисунку 3.7 стабільність роботи системи складає 92.19% , що є досить високим показником. Інші збої в роботі системі були викликані недостатньою кількістю вільного простору на смартфорі, нажаль такий тип збою програмно вирішити неможливо.

Під час тестування додатку було виправлено всі несправності та додано обробку всіх виняткових ситуацій. Було перевірено всі модулі роботи. Проведено позитивне та негативне тестування. Отримана стабільна версія додатку з номером v1.2.14

- 1 версія
- 2 виправлення глобального модуля
- 14 правка

Тестувальники надали позитивний відгук про стабільність додатку та коректність відображення всіх візуальних елементів на будь яких моделях смартфонів iPhone.

3.5 Висновок

У данному розділі було розроблено модулі Мапи, Головного екрану та Стрічки новин.

При тестуванні було виявлено проблеми з звантаженням данних при зміні інтернет підключення що зменшувало ефективність роботи програми. Було виправлено цю проблему за допомогою функції яка починає завантаження на новому типі інтернет підключення.

Протестовано роботу нового модулю, який перезавантажує дані при зміні інтернет підключення. Протестовано всі виняткові ситуації та їх виправлення.

Розглянуто різні типи вирішення певних технічних моментів та їх вирішення. Розроблено стабільну версію додатку з порядковим номером v1.2.14, що надало змогу передати проект у відділ тестування та можливість випуску додатку в стор.

Проаналізовано стабільність роботи системи що склало 92.19%. Відгук команди MotusDragRacing надав позитивне враження від додатку, зазначивши про стабільну роботу системи та високу швидкість завантаження даних, а також новий зручний функціонал, який не був реалізований в будь яких додатках.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою аудиту технологій є оцінка потенціалу комерційного розвитку. Два незалежні експерти брали участь у технологічному аудиті. Такими експертами будуть Яровий А.А. та Крилик Л.В. Ми оцінюємо потенціал комерційного розвитку за 10 критеріями за 10 - бальною шкалою.
Результати оцінки потенціалу комерційного розвитку наведені в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	9	7
2	7	5
3	10	7
4	8	6
5	6	6
6	7	7
7	4	6
8	8	8
9	10	9
10	9	4

Обробка результатів опитування

Кількість балів	$СБ_1 = 78$	$СБ_2 = 65$
Середньоарифметична кількість балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{2} = 71{,}5$	

За результатами оцінювання, можемо зробити висновок високого комерційного потенціалу розробки.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи

Для розробки додатку потрібні наступні витрати:

1) Заробітна плата працівників.

Заробітна плата вираховується за формулую 4.1:

$$З_0 = \frac{М}{Т_р} \cdot t, \qquad (4.1)$$

де M - місячний оклад окремого розробника;

T_p - кількість робочих днів за місяць, $T_p = 24$ робочих днів;

t – кількість робочих днів, $t = 90$ днів.

Розрахування заробітної плати програміста та керівника наведино у таблиці 4.2.

Таблиця 4.2 – Розрахунок заробітної плати керівника та програміста.

Працівник	Оклад *М*, грн.	Оплата за робочий день, грн	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5000	208,33	4	833,33
Інженер-програміст	7500	312,5	90	28125
			Всього:	28958,33

Додаткова заробітна плата:

$$З_{дод} = 0{,}1 \cdot 28958{,}33 = 2895{,}83 \text{ (грн)}.$$

Нарахування заробітної плати операторам НЗП вважається 37,5… 40% їх основної та додаткової заробітної плати 4.2:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп} = (28958{,}33 + 2895{,}83) \cdot \frac{37{,}5}{100} = 11945{,}31.$$

Розрахунок амортизаційних витрат для програмного забезпечення 4.3:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (4.3)$$

де *Ц* – балансова вартість обладнання, грн;

H_a – річна норма амортизаційних витрат % (для програмного забезпечення 25%);

Т – термін використання, Т=3 міс.

Таблиця 4.3 – Розрахунок амортизаційних витрат

Найменування програмного забезпечення	Балансова вартість, грн	Норма амортизації, %	Термін використання, міс	Величина амортизаційних витрат, грн
MacBook	58000	25	3	145,25
			Всього:	145,25

Розрахунок витрат на комплектуючі 4.4:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i , \qquad (4.4)$$

де *n* – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих *і*-го виду, грн;

Кі – коефіцієнт транспортних витрат, *Кі* = 1,1.

Таблиця 4.4 - Витрати на комплектуючі для розробки ПЗ

Найменування комплектуючих	Одиниці виміру	Ціна, грн	Витрачено	Вартість, грн
Ручка	Шт	20	1	20
Блокнот	Шт	40	1	40
			Всього	60

Витрати за електроенергію розраховуються за формулою 4.5:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \qquad (4.4)$$

де *В* – вартість 1кВт-години електроенергії (В=1.9 грн/кВт);

П – установлена потужність комп’ютера (*П*=0,7кВт);

Ф – фактична кількість годин роботи комп'ютера (*Ф*=498 год.);

$К_п$ – коефіцієнт використання потужності ($К_п$< 1, $К_п$ = 0,8).

$$В_е= 1,9 \cdot 0,7 \cdot 498 \cdot 0,8 = 529,87 \text{ (грн)}.$$

Розрахуємо інші витрати $B_{ін}$. Іншими витрати $I_в$ можливо прийняти (100…300)% від суми заробітної плати усіх робітників, які виконували роботу, тобто 4.6:

$$В_{ін} = (1. .3) \cdot (З_о + З_р). \quad (4.6)$$

Розрахунок інших витрат:

$$В_{ін} = 1 * (28958,33 + 2895,83) = 31854,16 \text{ (грн)}.$$

Сума всіх витрат для данної роботи 4.7:

$$В=З_о+З_д+Н_{зп}+А+К+В_е+В_{ін}, \quad (4.7)$$

$$В = 28958,33 + 2895,83 + 11945,31 + 145,25 + 60 + 529,87 + 31854,16 = 76388,75(\text{грн}).$$

Розрахунок загальної вартістості наукової роботи $B_{заг}$ 4.8:

$$В_{заг} = \frac{В}{\alpha} \quad (4.8)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{76388,75}{1} = 76388,75.$$

Прогнозування загальних витрат на виконання та впровадження наукової роботи:

$$ЗВ = \frac{В_{заг}}{\beta},$$

де β – коефіцієнт, який характеризує етап виконання даної роботи. Розрахунок загальних витрат:

$$ЗВ = \frac{76388,75}{0.9} = 84876,38.$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Ми плануємо отримати прибуток від представлення наших результатів розвитку. Зростання чистого доходу можна виміряти в теперішній вартості грошей. Це забезпечить додаткові кошти компанії (організації), що поліпшить фінансові показники.

Оцінка приросту чистого прибутку підприємства від впровадження результатів наукових розробок. У цьому випадку приріст чистого прибутку підприємства за кожен рік, протягом якого будуть очікуватися позитивні результати від впровадження розробки 4.9:

$$\Delta П_{і} = \sum_{1}^{n}(\Delta П_{я} \cdot N + П_{я}\Delta N)_{i}\ , \qquad (4.9)$$

де $\Delta П_{я}$ – покращення основного якісного показника;

тN – основний кількісний показник;

ΔN – покращення основного кількісного показника

$П_я$ – основний якісний показник;

n – кількість років.

В результаті впровадження результатів наукових розробок вартість ІТ-продукції зменшиться на 15 грн (що автоматично збільшить чистий прибуток підприємства на 15 грн) та збільшить кількість користувачів: за перший рік – 500 користувачів, за другий рік – 450 користувачів, за третій рік – 350 користувачів.

Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1200 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 300 грн.

Збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 15 \cdot 1200 + (300 + 15) \cdot 500 = 205500.$$

Протягом другого року:

$$\Delta П_2 = 15 \cdot 1200 + (300 + 15) \cdot (500 + 450) = 347250.$$

Протягом третього року:

$$\Delta П_3 = 15 \cdot 1200 + (300 + 15) \cdot (500 + 450 + 350) = 426000.$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестицій.

Абсолютна ефективність $E_{абс}$ вкладених інвестицій 4.10:

$$Еабс = (ПП - PV), \quad (4.10)$$

де ΔПі – збільшення чистого прибутку у кожному із років, грн;

τ – ставка дисконтування, для України = 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 1,2,3 років.

Розрахунок прибудків 4.11:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t},), \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, грн;

τ – ставка дисконтування;

t – період часу.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{84876,38}{(1+0.1)^0} + \frac{205500}{(1+0.1)^1} + \frac{347250}{(1+0.1)^2} + \frac{426000}{(1+0.1)^3} = 625661,75.$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 625661,75 - 84876,38 = 540785,37.$$

Оскільки $E_{абс} > 0$, внесок коштів на виконання та впровадження результатів НДДКР буде мати позитивний результат.

Розрахунок щорічної ефективності вкладених в наукову розробку коштів 4.12:

$$E_в = \sqrt[Т]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.12)$$

де Еабс – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

Тж – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{540785,37}{84876,38}} - 1 = 6,37 \text{ або } 637\%.$$

Далі, розраховану величину $E_в$ порівнюємо з мінімальною ставкою дисконтування $\tau_{мін}$. Мінімальна ставка дисконтування $\tau_{мін}$ 4.13:

$$\tau = d + f, \quad (4.13)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках Україні d = 0,1;

f – показник, що характеризує ризикованість вкладень f = 0,1.

$$\tau = 0.1 + 0.1 = 0.2.$$

Оскільки $E_в = 637\% > \tau_{мін} = 0,2 = 20\%$, то інвестор буде зацікавлений вкладати гроші в розробку додатку.

Термін окупності вкладень у реалізацію наукового проекту інвестицій $Т_{ок}$ 4.14:

$$Т_{ок} = \frac{1}{E_в} = \frac{1}{6,37} = 0,15 \text{ року}. \quad (4.13)$$

Проаналізувавши період заробітної плати цієї наукової розробки, ми можемо зробити висновок, що фінансування цієї наукової розробки буде належним.

4.5 Висновок

У цьому розділі оцінюється потенціал комерційного розвитку інформаційних технологій організації соціальної мережі.

Проведено технологічний аудит за участю двох незалежних експертів. Встановлено, що рівень потенціалу комерційного розвитку вище середнього.

Згідно з оцінкою, запропонована розробка є кращою та більш конкурентоспроможною. Крім того, розраховуються зарплата, амортизаційні відрахування, компоненти та енергетичні витрати. Загальна вартість продажу 84876,38 грн.

Комерційний ефект прогнозується від реалізації результатів розвитку. Основний прибуток отримують протягом трьох років.

Чистий прибуток компанії за три роки після впровадження розробки становить 625661,75 грн. Також була розрахована ефективність вкладених інвестицій та термін повернення. Життєвий цикл наукового розвитку становить 3 роки. Абсолютна віддача інвестицій становить 540785,37 грн.

Щорічна відносна ефективність вкладених коштів становить 637%, мінімальний поріг - 20%. Термін повернення інвестиції становить 0,15 року. Тому фінансування розвитку інформаційних технологій для створення соціальної мережі VinRiders доцільне для інвесторів, і вони можуть отримати прибуток.

ВИСНОВКИ

У магістерській роботі запропоновано інформаційну технологію розробки соціальної мережі VinRiders.

У першому розділі проведено аналіз сучасних систем-аналогів, які використовуються для спілкування. В результаті аналізу аналогів та методів реалізації соціальних мереж сформульовано проблему недостатнього функціоналу та відсутність спеціалізованих соціальних мереж. Здійснено постановку задачі, досліджено перелік необхідних функцій які повинна містити інформаційна технологія.

Другий розділ присвячено розробці алгоритму роботи що використовується для задач роботи додатку. Обгрунтовано вибір мови програмування та методів розробки. Обгрунтовано вибір архітектури розробки та серсерну модель.Запропоновано структуру зберігання даних на серверній частині в зручному для обробки варіанті з урахуванням специфікацій сервісу Firebase. Спроектовано схему алгоритму роботи програми.

Третій розділ присвячено проектуванню та розробці інформаційної технології строрення соціальної мережі VinRiders. Розроблера специфіка роботи контроллерів. Розроблено ценкціонал контроллеру мапи з розставленням маркерів на мапі. Розширено функціонал в порівнянні з іншими системами аналогами. Розроблено функціонал контроллеру новин проведено аналіз зручності користування. Розроблено функціонал роботи контроллеру акаунту та логіки показу окремих графічних елементів. Протестовано стабільність роботи програми за участі команди MotusDragRacing, покращено функціонал відповідно до побажань тестувальників. В результаті тестування було отримано аналітику сервісу Firebase з результатом стабільності 92,19% що є досить високим показником для соціальних мереж.

У четвертому розділі здійснено оцінювання комерційного потенціалу розробки. Проведено технологічний аудит із залученням експертів. Здігдно висновків експертів рівень комерційного потенціалу розробки вище середнього. Головним конкурентом розробки є мессенджер Telegtam. Здійснено прогнозування витрат на виконання науково-дослідної роботи. Розраховано витрати на визаробітну плату та амортизаційні відрахування, а такоє витрати на електроенергію що становить 84876,38 грн та спрогнозовано прибуток за три роки використання програми, що становить 625661,75 грн за рік, а також інвестеційний потенціал.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. Барабан, А. Яровий, Д. Пантелюк, Д. Кудрявцев Robot “Scorpion”: computer vision system with images recognition. – Тези доповідей. XLVIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії. – В. ВНТУ, 2019. – [Електронний ресурс]. – Тип доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2019/paper/view/7814
2. MVP – Режим доступу: https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE_%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82
3. Telegram – Режим доступу: https://ru.wikipedia.org/wiki/Telegram
4. Instagram – Режим доступу: https://ru.wikipedia.org/wiki/Instagram
5. Viber – Режим доступу: https://ru.wikipedia.org/wiki/Viber
6. MVVM – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-ViewModel
7. Viper – Режим доступу: https://ru.wikipedia.org/wiki/Viper
8. MVC – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller
9. Firebase – Режим доступу: https://ru.wikipedia.org/wiki/Firebase
10. Azure – Режим доступу: https://ru.wikipedia.org/wiki/Azure_DevOps_Server
11. Apple Developer – Режим доступу: https://ru.wikipedia.org/wiki/Apple_Developer
12. Apple – Режим доступу: https://developer.apple.com/
13. Xcode – Режим доступу: https://ru.wikipedia.org/wiki/Xcode
14. SwiftBook – Режим доступу: https://swiftbook.ru

15. Firebase DevOps – Режим доступу: https://firebase.google.com/?&gclid=EAIaIQobChMI5ICz_IzL7QIVATcYCh33PAVQEAAYASAAEgJAHPD_BwE
16. Козловський В.О. Основи підприємництва. Курс лекцій. Част. 1\В.О. Козловський – Вінниця: ВНТУ, 2005 – 196 с.
17. Козловський В.О. Основи підприємництва. Курс лекцій. Част. 2\В.О. Козловський – Вінниця: ВНТУ, 2006 – 184 с.
18. Козловський В.О. Підприємницька діяльність. Курс лекцій. Част. 1 \ В.О. Козловський – Вінниця: ВНТУ, 2006 – 175 с.
19. Козловський В.О. Підприємницька діяльність. Курс лекцій. Част. 2 \ В.О. Козловський – Вінниця: ВНТУ, 2006 – 170 с.
20. Козловський В.О. Інноваційний менеджмент. Курс лекцій. Навчальний посібник \ В.О. Козловський – Вінниця: ВНТУ, 2007 – 210 с.
21. Козловський В.О., Лесько О.Й. Інноваційний менеджмент. Практикум. \ Козловський В.О., Лесько О.Й.– Вінниця: ВНТУ, 2006 – 166 с.