МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

КРОСПЛАТФОРМЕНИЙ ЗАСТОСУНОК «ТРЕКЕР ЗВИЧОК» ДЛЯ МОБІЛЬНИХ ДЕВАЙСІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910110

Виконав студент 4-го курсу, групи 401
_____С. А. Дичко
«19» червня 2023 р.

Керівник: PhD, ст. викладач
_____К. О. Антіпова
«19» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Дичко Сергію Анатолійовичу.

1. Тема кваліфікаційної роботи «Кросплатформений застосунок «Трекер звичок» для мобільних девайсів».

Керівник роботи Антіпова Катерина Олександрівна, PhD, ст. викладач.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи «____» _____ 2023 р

3. Вхідні (початкові) дані до роботи: необхідність ефективного відстеження та управління особистими звичками та цілями за допомогою мобільного застосунку.

Очікуваним результатом роботи є кросплатформений мобільний застосунок.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):аналіз предметної сфери розробки мобільних застосунків;

- аналіз систем-аналогів;
- специфікація вимог до ПЗ;
- проєктування та моделювання власного рішення;
- розробка ПЗ та тестування системи.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Дослідження питань охорони праці, які безпосередньо пов'язані з діяльністю розробника програмного забезпечення»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., к.т.н., доцент	

Керівник роботи PhD, ст. викладач Антіпова К. О.
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання Дичко С. А.
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Кросплатформений застосунок «Трекер звичок» для мобільних девайсів

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2022	29.10.2022	Виконано
2	Отримання завдання на виконання БКР	20.11.2022	20.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	01.12.2022	06.12.2022	Виконано
4	Отримання завдання на переддипломну практику	28.04.2023	01.05.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: аналіз сучасних аналогів мобільний застосунків, огляд існуючих технологій, створення специфікацій та розробка ПЗ	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	Виконано

Розробив студент _____Дичко С. А._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __PhD, ст. викладач Антіпова К. О.___ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__09_» ___12_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Дичка Сергія Анатолійовича

Тема: «Кросплатформений застосунок «Трекер звичок» для мобільних девайсів»

Актуальність роботи пов’язана з розвитком мобільних технологій, який відкриває нові можливості для користувачів мобільних пристроїв, забезпечуючи їм доступ до різноманітних сервісів та застосунків. Кросплатформені застосунки стали популярними серед розробників, які бажають забезпечити своїм користувачам доступ до застосунків на різних мобільних платформах. Один із таких застосунків – «Трекер звичок» – допомагає користувачам вести контроль за своїм здоров'ям та дотримуватися режиму життя.

Об'єктом кваліфікаційної роботи є кросплатформений застосунок "Трекер звичок".

Предметом кваліфікаційної роботи є технології розробки кросплатформених застосунків для мобільних пристроїв.

Кваліфікаційна робота бакалавра присвячена розробці кросплатформеного застосунку "Трекер звичок" для мобільних девайсів

Метою даної кваліфікаційної роботи є покращення доступності та зручності користування застосунком "Трекер звичок" для користувачів мобільних пристроїв шляхом розробки кросплатформеної версії застосунку.

Для досягнення цієї мети було проведено аналіз існуючих рішень у цій області та визначені вимоги до програмного продукту. Далі, на основі обраних технологій та методологій розробки програмного забезпечення було розроблено та реалізовано застосунок для операційних систем Android та iOS. У результаті виконаної роботи було отримано функціональний та зручний інструмент для відстеження та контролювання особистих звичок, який може бути використаний користувачами

мобільних пристроїв. Отже, кваліфікаційна робота бакалавра є актуальною та важливою у розвитку мобільних технологій та користувацької продуктивності.

У вступі розкривається актуальність дослідження за обраним напрямом, ставиться проблема, мета і завдання дослідження, визначаються об'єкт та предмет дослідження, обґрунтування основних проєктних рішень, вказується його теоретична, практична значущість.

У першому розділі роботи проводиться системний аналіз обраної предметної області та, на його основі, формулюється постановка задачі та специфікація вимог до програмного забезпечення.

У другому розділі розробляються проєктні рішення, що забезпечують виконання специфікації вимог до ПЗ.

У третьому розділі описується виконана робота з моделювання та конструювання ПЗ.

У четвертому розділі представляється виконана робота з кодування.

КРБ викладена на ___сторінки, вона містить ___ розділи, ___ілюстрацій, ___ таблиці, 14_ джерел в переліку посилань.

Ключові слова: *React Native, Nest JS, Habit tracker, Android, IOS.*

ABSTRACT

of the Bachelor’s Thesis

Cross-platform application "Habit Tracker" for mobile devices

Student of group 401 Black Sea National University

Serhii Dychko Anatolievich

The bachelor's thesis is dedicated to the development of a cross-platform application "Habit Tracker" for mobile devices. The main objective of the work is to develop a convenient and effective tool for tracking and controlling the personal habits of the user. To achieve this goal, an analysis of existing solutions in this area was carried out, and the requirements for the software product were determined. Based on the chosen technologies and software development methodologies, an application was developed and implemented for Android and iOS operating systems. As a result of the work performed, a functional and convenient tool for tracking and controlling personal habits was obtained, which can be used by users of mobile devices. Therefore, the bachelor's thesis is relevant and important in the development of mobile technologies and user productivity.

The object of the qualifying work is the cross-platform application "Habit Tracker".

The subject of the qualifying work is the technologies for developing cross-platform applications for mobile devices.

The introduction reveals the relevance of the study in the chosen area, sets the problem, purpose and objectives of the study, defines the object and subject of research, justification of major design decisions.

In the first chapter of the work a systematic analysis of the selected subject area is carried out and, on its basis, the problem statement and specification of software requirements are formulated.

The second chapter develops design solutions that ensure compliance with the specification of software requirements.

The third chapter describes the work performed on software modeling and design.

The fourth chapter presents the work done on coding.

KRB is laid out on ___ pages, it contains ___ sections, ___ illustrations, ___ tables, _14__ sources in the list of links.

Keywords: *React Native, Nest JS, Habit tracker, Android, IOS.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ORM – Object-Relational Mapping

REST – Representational State Transfer

CRUD – Create, Read, Update, and Delete

API – Application Programming Interface

REST – Representational State Transfer

TS – TypeScript

ВСТУП

Актуальність роботи пов’язана з розвитком мобільних технологій, який відкриває нові можливості для користувачів мобільних пристроїв, забезпечуючи їм доступ до різноманітних сервісів та застосунків. Кросплатформені застосунки стали популярними серед розробників, які бажають забезпечити своїм користувачам доступ до застосунків на різних мобільних платформах. Один із таких – «Трекер звичок» – допомагає користувачам вести контроль за своїм здоров'ям та дотримуватися режиму життя. Даний застосунок розроблений для різних мобільних платформ, включаючи Android та IOS, та забезпечує функціональні можливості для стеження за різними аспектами життя користувача, такими як харчування, фізична активність, сон та інші.

Об'єктом кваліфікаційної роботи є кросплатформений застосунок "Трекер звичок".

Предметом кваліфікаційної роботи є технології розробки кросплатформених застосунків для мобільних пристроїв.

Метою даної кваліфікаційної роботи є покращення доступності та зручності користування застосунком "Трекер звичок" для користувачів мобільних пристроїв шляхом розробки кросплатформеної версії застосунку. Розробка застосунку дозволить забезпечити доступ користувачів до функціональних можливостей застосунку на різних мобільних платформах, що значно поліпшить його корисність та використання.

Завданням роботи є вивчення кросплатформених технологій розробки. Оцінити їх можливості, переваги та недоліки, а також визначити, який з них найбільше відповідає потребам розробки кросплатформеної версії застосунку "Трекер звичок". Потрібно розробити зручний та ефективний інструмент, який буде простий у використанні та зможе задовольнити потреби різних категорій користувачів.

Для досягнення поставленої мети, в рамках кваліфікаційної роботи було проведено аналіз існуючих рішень в галузі трекінгу та контролю звичок. Були проаналізовані такі застосунки, як Google Fit, Loop Habit Tracker, Todoist та інші. При цьому були визначені переваги та недоліки кожної з програм, що дозволило визначити ключові функції та вимоги до розроблюваного продукту.

Однією з головних переваг розроблюваного кросплатформеного застосунку "Трекер звичок" є його універсальність та доступність для користувачів різних мобільних платформ, таких як Android та iOS. Застосунок буде мати інтуїтивний та зручний інтерфейс, що дозволить легко встановлювати та керувати звичками, стежити за їх виконанням та визначати прогрес досягнення цілей. Застосунок буде включати такі функції, як статистика та аналітика, налаштування та редагування звичок, а також інші корисні інструменти для покращення продуктивності та досягнення цілей користувачів.

Таким чином, розробка кросплатформеного застосунку "Трекер звичок" для мобільних девайсів має великий потенціал для використання користувачами та покращення їхнього життя, надаючи зручний та ефективний інструмент для керування особистими звичками та досягнення поставлених цілей.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ РОЗРОБКИ КРОСПЛАТФОРМЕНИХ МОБІЛЬНИХ ЗАСТОСУНКІВ

Розробка кросплатформених мобільних застосунків - це процес створення програмного забезпечення, яке може працювати на різних мобільних платформах, таких як IOS та Android, за допомогою спільного коду. Цей підхід є досить популярним серед розробників, оскільки він дозволяє значно знизити витрати на розробку та підтримку застосунків, оскільки один і той же код можна використовувати для створення застосунків на різних платформах.

Основні технології, які використовуються для розробки кросплатформених мобільних застосунків, включають:

React Native - це фреймворк для розробки мобільних застосунків, який дозволяє використовувати JavaScript для створення застосунків на iOS та Android. Він надає доступ до нативних компонентів та API, що дозволяє створювати швидкі та ефективні застосунки.

Flutter - це SDK для розробки мобільних застосунків, який використовує мову програмування Dart. Він надає доступ до широкого спектру готових компонентів та бібліотек, що дозволяє швидко створювати застосунки на різних платформах.

Xamarin - це платформа для розробки мобільних застосунків, яка використовує мову програмування C#. Вона надає доступ до нативних компонентів та API, що дозволяє створювати застосунки на iOS та Android.

PhoneGap - це фреймворк для розробки мобільних застосунків, який дозволяє використовувати веб-технології, такі як HTML, CSS та JavaScript, для створення застосунків на різних платформах.

1.1 Види мобільних застосунків

Мобільні застосунки є програмними засобами, призначеними для використання на мобільних пристроях, таких як смартфони та планшети. Вони можуть мати різноманітні функції та призначення, залежно від потреб користувачів та бізнес-цілей [1].

Соціальні мережі

Соціальні мережі – застосунки для спілкування, обміну контентом, знайомств та зберігання даних користувачів.

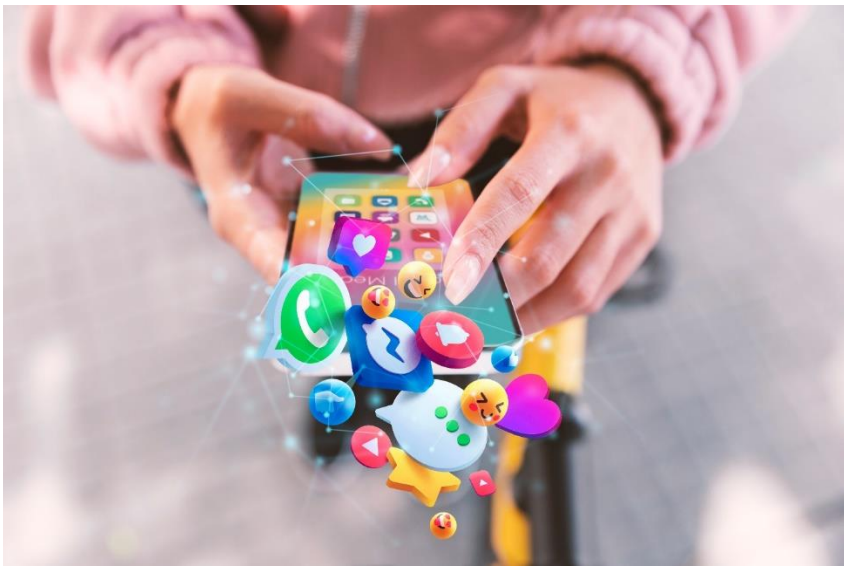

Рисунок 1.1 – Соціальні мережі

Соціальні мережі є одним з найбільш популярних видів мобільних застосунків. Вони дозволяють користувачам обмінюватись повідомленнями, фотографіями та відео, спілкуватись з друзями та знайомими, створювати групи за інтересами та знайомитись з новими людьми.

Найбільш популярні соціальні мережі включають Facebook, Instagram, Twitter, TikTok, LinkedIn та Snapchat. Кожна з них має свої особливості та функції, які роблять їх унікальними та привабливими для користувачів.

Наприклад, Facebook є однією з найбільш широко використовуваних соціальних мереж, яка дозволяє користувачам зв'язуватись з друзями та родичами, обмінюватись повідомленнями та спільно переглядати фотографії та відео. Instagram, з іншого боку, є соціальною мережею, спрямованою на візуальний контент, зокрема фотографії та відео. Також вона дозволяє користувачам ділитись своїми історіями та взаємодіяти з популярними брендами та знаменитостями.

Бізнес-застосунки

Застосунки, що допомагають у проведенні бізнес-операцій та оптимізації роботи компаній, включаючи системи управління продажами, фінансові та аналітичні інструменти.

Найбільш популярні бізнес-застосунки включають в себе Microsoft Office, Nova Poshta, Google Workspace, Slack, Trello, Asana, Zoom та Skype. Кожен з цих застосунків має свої особливості та функції, які можуть допомогти в різних аспектах роботи в компанії.

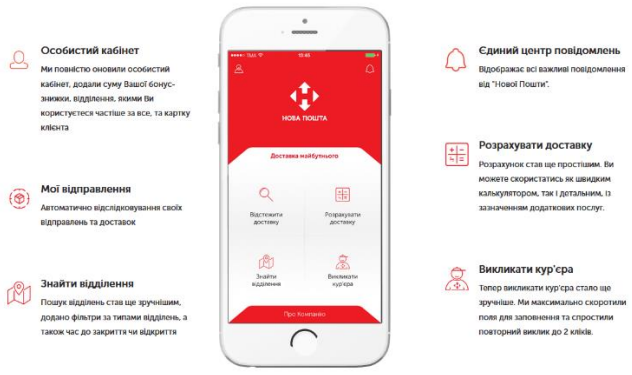

Рисунок 1.2 – Інтерфейс мобільного застосунку «Нова Пошта»

Застосунок Нової пошти є логістичним застосунком, який допомагає користувачам здійснювати відправлення та отримання посилок, відстежувати їх статус та знаходити найближчі відділення компанії. Цей застосунок є прикладом бізнес-застосунка, орієнтованого на вирішення специфічних завдань у логістичному секторі.

Ігрові застосунки

Ігри - мобільні застосунки різного типу, включаючи ігри на вдачу та логіку, стратегічні ігри, спортивні ігри та інші.

Рисунок 1.3 – Приклад ігрового мобільного застосунку

Мобільні ігри є одним з найпопулярніших типів мобільних застосунків, які надають розвагу користувачам на їх мобільних пристроях. Ігри можуть бути розроблені для різних платформ, таких як Android та iOS, і можуть бути доступні як безкоштовні, так і платні.

Ігри можуть бути різних жанрів, включаючи головоломки, стратегії, спортивні ігри, гонки, шутери, ролеві ігри та інші. Багато з цих ігор мають мультиплеєрний режим, який дозволяє користувачам змагатися з іншими гравцями онлайн.

Деякі з найпопулярніших мобільних ігор включають Angry Birds, Candy Crush, Clash of Clans, Fortnite, PUBG та Minecraft. Кожна з цих ігор має свою власну унікальну геймплей та характеристики, які привертають гравців з усього світу.

Розробка мобільних ігор може бути досить складною і вимагати значних ресурсів, таких як розробка графіки та звукового дизайну, програмування та тестування. Однак успіх мобільної гри може принести значний прибуток для розробників, якщо вона стане популярною серед гравців.

Медіа-застосунки

Мобільні застосунки для перегляду та споживання різних видів мультимедійного контенту, включаючи музику, фільми, телевізійні шоу та інше.

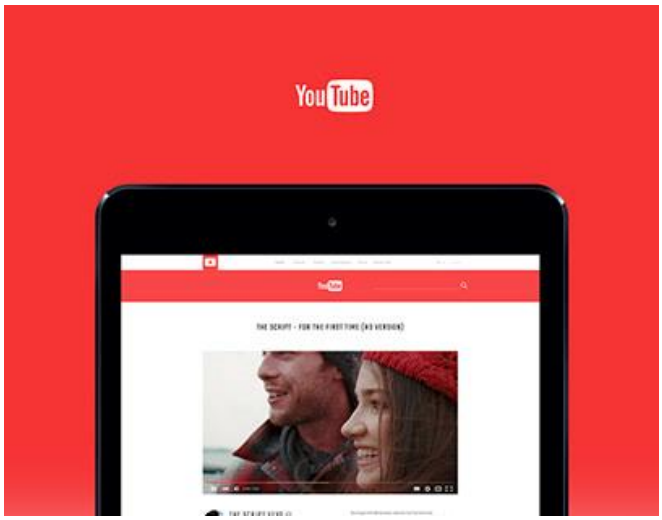

Рисунок 1.4 – Приклад медіа-застосунка

Оздоровчі застосунки

Оздоровчі застосунки - мобільні застосунки, що допомагають в моніторингу здоров'я, діагностиці та лікуванні різних захворювань та проблем.

Рисунок 1.5 – Інтерефейс оздоровчого застосунку

До оздоровчих застосунків також можна віднести застосунок, що надає можливість відслідковувати свою продуктивність під час виконання тих чи інших корисних звичок.

1.2 Аналіз сучасних аналогів мобільного застосунку з можливістю відстежувати свої активності

Залежно від теми роботи було проаналізовано застосунки, які використовують схожий функціонал, архітектуру та створені із застосуванням кастомних технологій. На сьогоднішній день на ринку існує декілька популярних мобільних застосунків для відстеження активності та здоров'я користувача. Розглянемо декілька сучасних аналогів таких застосунків.

Google Fit - це застосунок для Android, розроблений компанією Google. Він дозволяє користувачам відстежувати кроки, відстань, кількість активності та інші фізичні показники, а також збирати дані про серцевий ритм та рівень стресу. Google Fit синхронізується з іншими застосунками та фітнес-трекерами, що дозволяє користувачам зібрати всі дані в одному місці.

Рисунок 1.6 – Інтерфейс Google Fit

Таблиця 1.1 – Характеристики аналогу 1

Назва	Google Fit
Розробник	Google
Мова реалізації	Java, Kotlin
Функції	– відстеження кроків та відстані; – вимірювання пульсу; – відстеження активності впродовж дня; – відстеження активності під час занять спортом; – запис занять спортом та їх аналіз; – моніторинг сну; – вимірювання рівня стресу та пульсу відпочинку.

Кінець таблиці 1.1

Переваги	Безкоштовність та доступність - Google Fit доступний для завантаження безкоштовно для користувачів Android-пристроїв. Широкий спектр функцій - Google Fit має різноманітні функції для відстеження фізичної активності та здоров'я. Інтеграція з іншими застосунками - Google Fit інтегрується з іншими застосунками для спорту та фітнесу, такими як Strava, MyFitnessPal та інші.
Недоліки	Обмежена підтримка для інших операційних систем - Google Fit не має повної підтримки для інших операційних систем, таких як iOS. Не дуже інтуїтивний інтерфейс користувача - Деякі користувачі можуть вважати, що інтерфейс користувача Google Fit не дуже інтуїтивний. Потребує додаткового обладнання - Для вимірювання деяких параметрів здоров'я, таких як пульс та кисневий насиченість крові, потрібне додаткове обладнання.

Loop Habit Tracker - цей застосунок є безкоштовним та відкритим джерелом для Android. Він дозволяє користувачам створювати список звичок та відстежувати їх на щоденній основі. Застосунок має багато корисних функцій, таких як нагадування та графіки прогресу.

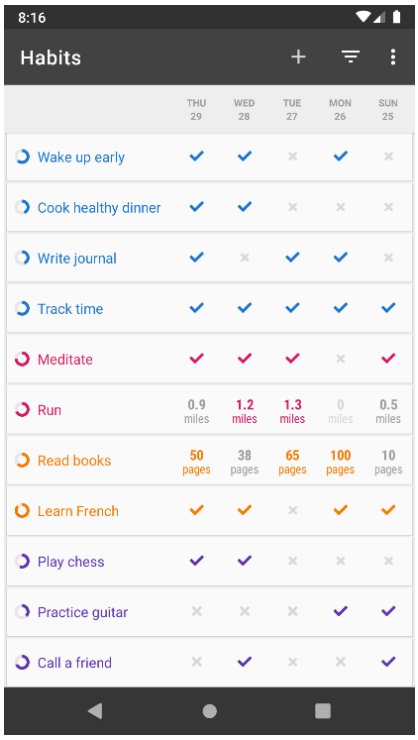

Рисунок 1.7 – Інтерфейс Loop Habit Tracker

Таблиця 1.2 – Характеристики аналогу 2

Назва	Loop Habit Tracker
Розробник	Álinson S Xavier
Мова реалізації	Java, Android Framework
Функції	– створення звичок для відстеження; – налаштування частоти та годин виконання звичок; – створення нагадувань про виконання звичок; – відстеження прогресу звичок за допомогою календаря та графіків.

Кінець таблиці 1.2

Переваги	– безкоштовний та відкритий вихідний код; – легкий та простий у використанні інтерфейс користувача; – можливість створювати необмежену кількість звичок для відстеження; – можливість налаштування нагадувань про виконання звичок; – гнучкість налаштування частоти та годин виконання звичок; – детальна статистика та аналіз даних про звички та їх прогресу; – можливість створення резервних копій даних та їх відновлення.
Недоліки	– застосунок доступний лише на платформі Android, тому користувачі інших платформ не можуть його використовувати; – застосунок не має вбудованих можливостей синхронізації даних між різними пристроями.

Todoist - це кросплатформений застосунок для керування завданнями, який дозволяє користувачам створювати список завдань та відстежувати їх виконання.

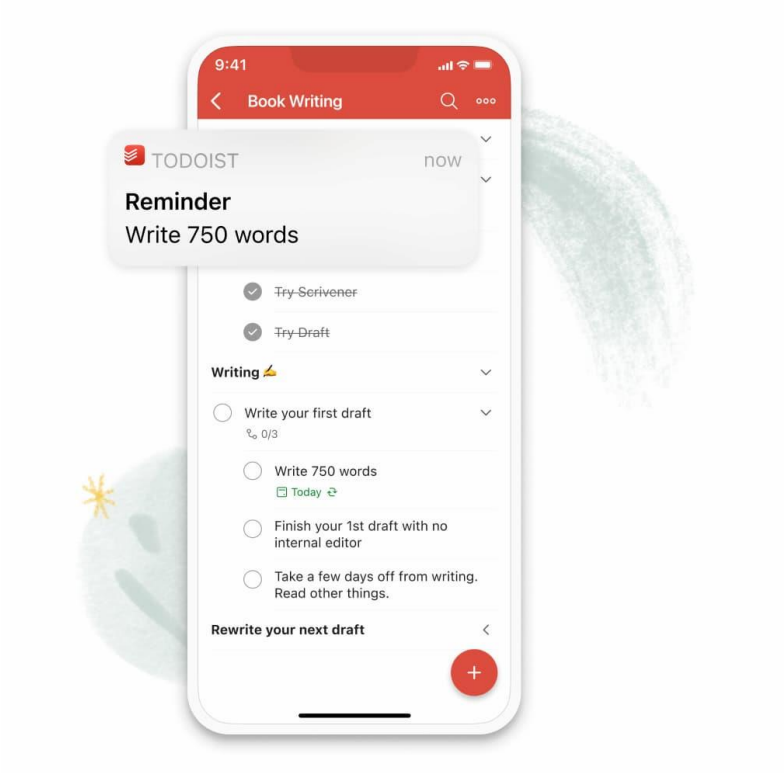

Рисунок 1.8 – Інтерфейс Todoist

Таблиця 1.3 – Характеристики аналогу 3

Назва	Todoist
Розробник	Doist
Мова реалізації	JavaScript, Node,js
Функції	– створення списку завдань та підзадач з можливістю надання пріоритетів, дедлайнів та міток;

Кінець таблиці 1.3

Функції	– нагадування про дедлайни та нагляд за термінами виконання завдань; – можливість розподілення завдань між різними проєктами та зберігання даних в хмарі; – керування завданнями з різних пристроїв через синхронізацію даних; – спільний доступ до списку завдань з іншими користувачами для спільної роботи над проєктами.
Переваги	– легкий та інтуїтивно зрозумілий інтерфейс; – можливість створення задач та списків у будь-якому порядку; – різноманітність функцій та можливостей налаштування; – широка інтеграція з іншими програмами та сервісами.
Недоліки	– деякі функції доступні тільки на платній версії; – обмежена можливість налаштування відображення списків та завдань.

1.3 Специфікація вимог до програмного забезпечення мобільного застосунку «Трекер звичок»

Проаналізувавши застосунки-аналоги, було визначено, що мобільний застосунок буде являти собою кросплатформений застосунок з можливістю виконання задач та переглядом статистики.

Застосунок має мати простий дизайн, щоб чітко відображати користувачеві його функціонал й підвищувати його особисту продуктивність. При цьому інтерфейс має відповідати вимогам кросплатформеності.

Було вирішено створити три основні вікна для навігації користувачем.

Daily Screen – екран на якому користувач бачить завдання необхідні для виконання в конкретний день тижня.

Flow Screen – екран, де користувач має можливість редагувати завдання, їх повторюваність.

Profile Screen – екран, який буде поєднувати статистику виконаних завдань та налаштування профілю користувача.

Унікальною особливістю застосунку буде можливість отримання нагороди за виконане завдання. Користувач сам має можливість оцінити складність завдання при його створенні (від 1 до 3 балів) й далі отримувати нагороду за виконання. В статистиці також буде відображатись кількість отриманих балів.

Застосунок матиме функції логіну й реєстрації, що надає можливість синхронізувати свої досягнення між різними девайсами.

Для досягнення мети кваліфікаційної роботи необхідно виконати наступні задачі:

1) вибір оптимальних технологій та інструментів для розробки кросплатформеного застосунку;

2) розробка дизайну інтерфейсу, що задовольняє потреби користувачів та

відповідає вимогам кросплатформеності;

3) створення основного функціоналу:

– функції реєстрації та авторизації;

– можливість додавання/редагування/видалення задач;

– редагувати повторюваність задач;

– змінювати активний список задач на інший;

– перегляд статистики.

4) розробка та тестування функціоналу застосунку, включаючи відстеження прогресу та повторення завдань;

5) оцінка ефективності та якості розробленого застосунку з використанням тестування та зібраних даних.

Даний проєкт буде розроблений з використанням технології React Native, що дозволяє створювати кросплатформені мобільні застосунки. Серверна частина застосунку буде розроблена на фреймворку Nest JS, який забезпечує швидку та ефективну розробку RESTful API. Застосунок буде забезпечувати безпечну збереження даних користувачів, а також надавати користувачам зручний та легкий у використанні інтерфейс.

Висновки до розділу 1

У даному розділі бакалаврської роботи було проведено аналіз предметної сфери розробки кросплатформених мобільних застосунків. Були розглянуті основні види мобільних застосунків, зокрема: соціальні мережі, бізнес-застосунки, ігри, трекери здоров'я та звичок. Також були детально розглянуті функції та особливості деяких конкретних застосунків - Google Fit та Loop Habit Tracker, а також надана інформація про Todoist.

Було виявлено, що кросплатформені мобільні застосунки набувають все більшої популярності, оскільки вони дозволяють розробникам створювати застосунки для різних платформ швидко та ефективно, що зменшує витрати на розробку та підтримку. Також важливою є можливість отримати доступ до широкого кола користувачів, що використовують різні платформи.

В цілому, аналіз предметної сфери розробки кросплатформених мобільних застосунків дозволив зробити висновок про важливість таких застосунків та їх популярність серед користувачів. Для подальшої розробки кросплатформеного застосунку трекер звичок необхідно було ретельно вивчити функціонал та особливості аналогів, що вже існують на ринку, та врахувати їх при розробці власного застосунку.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Моделювання даних є важливою частиною розробки кросплатформеного застосунку "Трекер звичок". Цей процес включає створення візуального представлення інформаційної системи або її складових. Мета моделювання даних полягає у визначенні типів даних, що використовуються та зберігаються в застосунку, встановленні зв'язків між ними, організації та групуванні даних, а також визначенні їхніх форматів та атрибутів [2].

Моделі даних розробляються з урахуванням бізнес-потреб. Правила та вимоги до моделі даних визначаються на підставі зворотного зв'язку з бізнесом і можуть бути включені до розробки нового застосунку або адаптовані до наявного.

Процес моделювання даних розпочинається зі збору вимог бізнесу від зацікавлених сторін та користувачів. Ці вимоги перетворюються в структури даних, що подібні до дорожньої карти або плану архітектора. Модель даних допомагає глибше зрозуміти те, що розробляється.

У моделюванні даних використовуються стандартизовані схеми та формальні методи, що забезпечують послідовний та передбачуваний підхід до керування даними всередині та поза організацією.

2.1 Діаграма використання

Діаграма варіантів використання є інструментом для узагальнення інформації про систему та користувачів, які взаємодіють з нею. Зазвичай вона представлена у вигляді графічного зображення, що відображає взаємодії між різними елементами системи. Діаграми варіантів використання надають інформацію про події, які відбуваються в системі, а також про те, як ці події протікають. Важливо зазначити, що діаграма варіантів використання не включає деталей щодо реалізації цих подій.

Випадок використання є підходом, що використовується в системному аналізі для визначення, пояснення та організації системних вимог. В цьому контексті термін "система" відноситься до розроблюваного або функціонуючого об'єкта, наприклад, веб-сайту для продажу та обслуговування товарів. Діаграми варіантів використання, які використовуються в UML (Unified Modeling Language), є стандартною нотацією для моделювання реальних об'єктів та систем. Діаграма варіантів використання має кілька переваг порівняно з іншими діаграмами, такими як блок-схеми. Деякі з цих переваг включають:

– відстеження реалізованих та перевірених варіантів використання, що дозволяє визначити функціональність, яка працює та ту, яка не працює;

– визначення вартості та складності системи шляхом ідентифікації функціональних вимог, які перейдуть до стадії розробки;

– використання природної мови в діаграмах варіантів використання сприяє легкому зрозумінню користувачами та сприяє ефективній комунікації з клієнтами;

– розбиття рішень на практичні функції дозволяє зменшити складність проблеми, яку система намагається вирішити. Логічне моделювання буде виконуватись за допомогою ER-методу (сутність-зв'язок), де логічна модель складатиметься з сутностей, які пов'язані між собою.

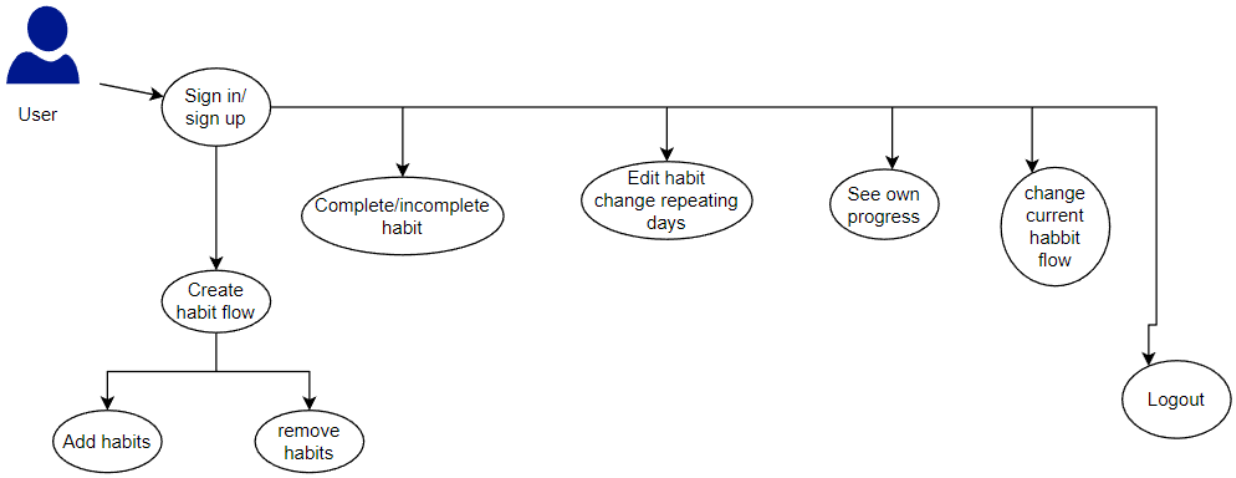

Рисунок 2.1 – Діаграма використання

Діаграма яскраво демонструє, що авторизований користувач є головним об’єктом взаємодії з застосунком. Майже кожний етап взаємодії є незалежним відповідно до інших, що дає користувачеві можливість власноруч пріоритезувати необхідний для нього функціонал.

2.2 Проєктування інтерфейсу мобільного застосунку

Розробка користувацького інтерфейсу планується з використанням готових компонентів з відкритої бібліотеки. Це рішення обумовлене ефективністю й зручністю кінцевого результату. Оскільки бібліотека має готовий набір рішень для створення кросплатформених компонент, що в свою чергу оптимізує час розробки та зменшує кількість потенційних помилок. Замість написання всіх компонентів з нуля, розробник може скористатися готовими рішеннями, що значно прискорює процес розробки. Також використання готових компонентів з відкритої бібліотеки дозволяє використовувати рішення, які підтримуються великою спільнотою розробників. Це забезпечує доступ до документації, прикладів використання та підтримки, що забезпечує якість розробки.

Головною метою при розробці дизайну є створення схематичного відображення для кожного з елементів, які взаємодіють з користувачем. Під час розробки дизайну слід зосередитися на забезпеченні доступності та зручного розташування компонентів, тоді як зовнішній вигляд може бути взятий з готової бібліотеки.

Дизайн інтерфейсу має бути привабливим, але ненав'язливим і непомітним, оскільки основною метою користувачів є отримання інформації. Інтерфейс є точкою взаємодії користувачів з мобільним застосунком. Хороший дизайн інтерфейсу поєднує візуальний дизайн, дизайн взаємодії та інформаційну архітектуру [3].

При плануванні дизайну застосунку було визначено, що основна взаємодія з користувачем буде проходити під час наступних трьох екранів:

– **daily screen** - основний екран, де користувач має можливість побачити список завдань необхідних для виконання на сьогоднішній день. Також на цьому екрані він має можливість позначити задачу як виконану/не виконану;

– **flow screen** - екран з активним списком задач. Користувач може змінювати опис задач та редагувати їх повторюваність;

– **profile screen** – екран, де буде зображена статистика користувача та швидкі налаштування.

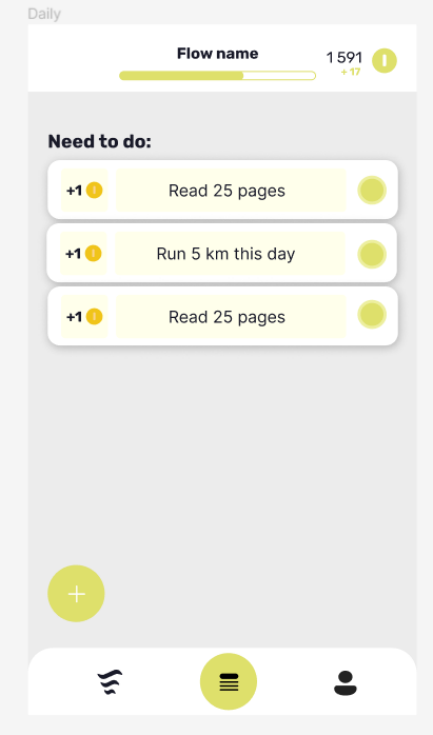

Рисунок 2.2 – Схематичний дизайн «Daily Screen»

Важливою особливістю першого екрану є наявність індикатора прогресу, який відображає співвідношення виконаних задач до наявних на активний день. Елемент буде розміщено в хедері екрану.

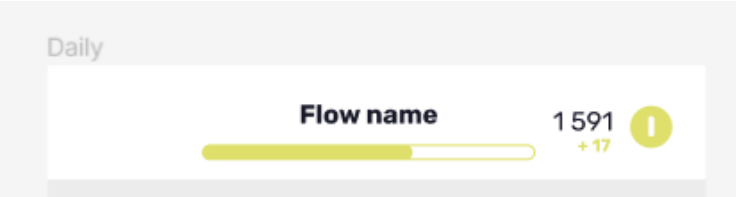

Рисунок 2.3 – Вигляд індикатора прогресу

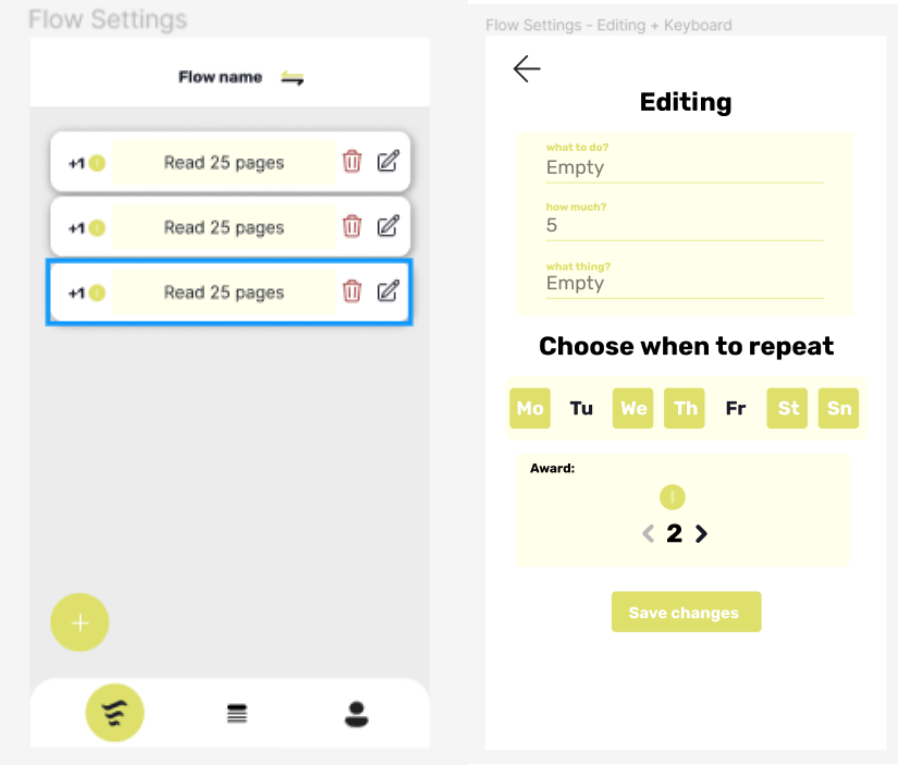

Рисунок 2.4 – Схематичний дизайн «Flow Screen»
та екрану редагування задачі

Як можна побачити з дизайну при редагуванні вибраної задачі має відкриватись вікно, яке надає користувачеві можливість змінити опис задачі її повторюваність та нагороду за її виконання. Також на дизайні зображено кнопку зі знаком «+». Вона матиме ключове значення, оскільки при натисканні на неї користувач матиме можливість додати нову задачу, або створити новий список задача «Flow». Кнопка та її функціонал повністю дублюється для першого екрану.

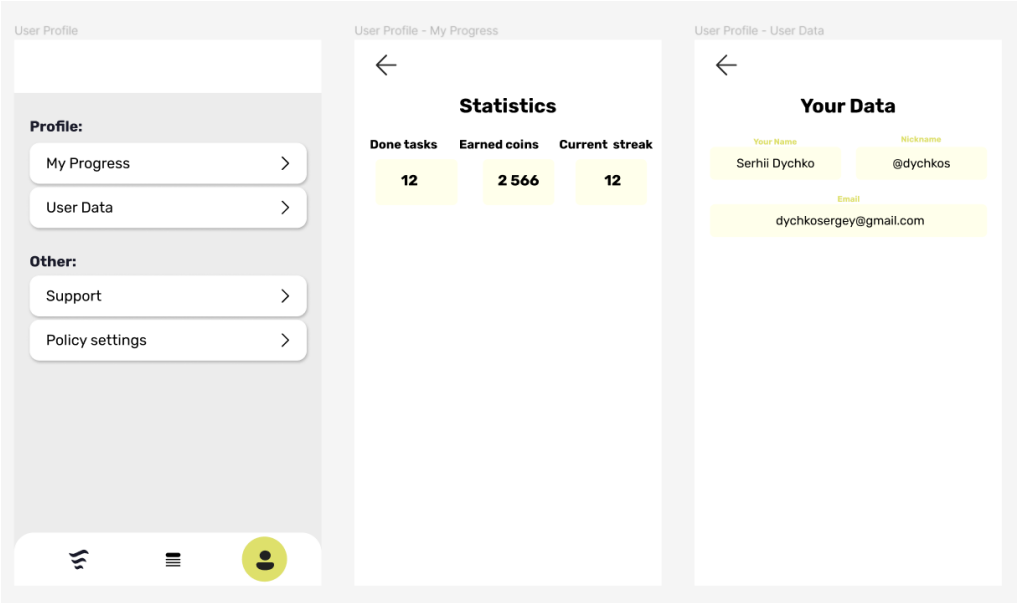

Рисунок 2.5 – Схематичний дизайн «Profile Screen»
та його внутрішніх екранів

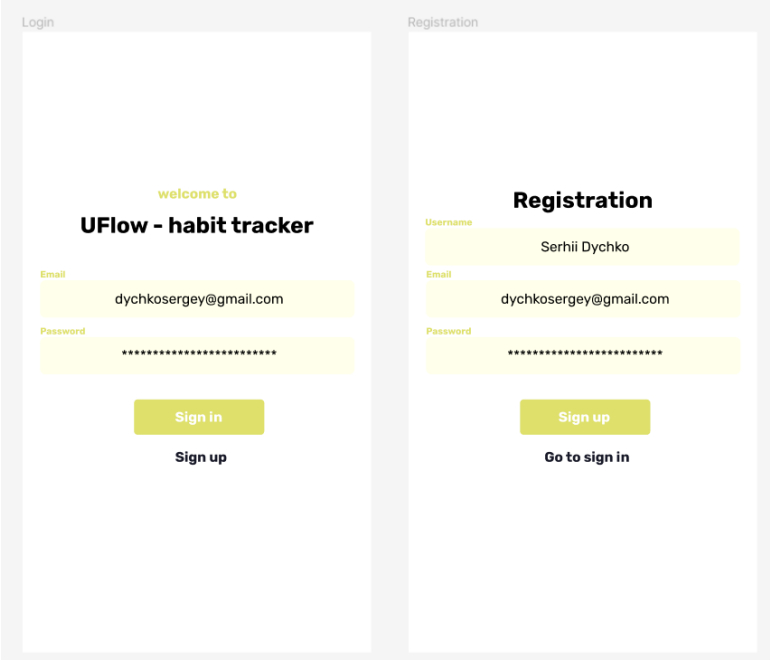

Рисунок 2.6 – Схематичний дизайн вікон
входу в систему та реєстрації

Описаний дизайн покриває більшість елементів взаємодії з користувачем. Додатковий функціонал (анімації, модальні вікна та ін. елементи) – будуть пере використані з бібліотеки готових компонентів та не вимагають додатково схематичного дизайну.

2.3 Візуальна карта мобільного застосунку

Мобільний застосунок може бути представлений у вигляді візуальної карти, яка наглядно відображає структуру та навігацію у застосунку. Вона включає основні сторінки та їх взаємозв'язки, допомагаючи користувачам легко орієнтуватися та переходити між різними частинами застосунку.

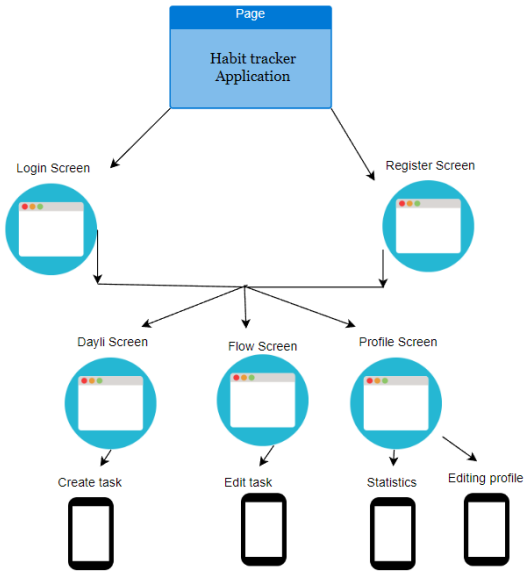

Рисунок 2.8 – Візуальна картка веб застосунку

Візуальна карта веб-застосунку зазвичай має ієрархічну структуру, де головна сторінка або дашборд є центральною точкою, а наступні сторінки відображаються як вкладені елементи.

Користувачі можуть використовувати цю карту для швидкого доступу до потрібних сторінок та отримання загального уявлення про структуру застосунку.

2.4 Проєктування бази даних мобільного застосунку

База даних - набір електронних даних, що зберігаються і надаються доступ до них. Вона має вбудовані обмеження та перевірки для забезпечення точності інформації. Бази даних також забезпечують безпеку даних за допомогою авторизації користувачів і специфікаторів доступу.

Таблиця 2.1 – Атрибути таблиць у базі даних

Таблиця	Атрибут	Опис атрибуту
users	id	Ключ
	email	Електронна пошта користувача
	username	Ім’я користувача
	fullname	Повне ім’я користувача
	password	Зашифрований пароль
	doneTasks	Кількість виконаних задач
	globalCoins	Кількість отриманих нагород
	earnedCoins	Кількість отриманих нагород за актуальний день
	createdAt	Дата реєстрації користувача
	updatedAt	Дата останнього оновлення користувача
flows	id	Ключ
	title	Назва списку задач

Продовження таблиці 2.1

flows	userId	Зовнішній ключ до таблиці користувачів
	chosen	Флаг, який вказує чи є список задач обраним користувачем
tasks	id	Ключ
	flowId	Зовнішній ключ, який вказує на список задач
	action	Що зробити в контексті задачі?
	how_many	Скільки зробити в контексті задачі?
	unit	Одиниця вимірювання дії в контексті задачі (сторінки, кілометри та ін.)
	reward	Нагорода за виконання задачі
	done	Флаг, який вказує на стан виконання задачі. Оновлюється зі зміною доби.
	createdAt	Дата створення задачі
	updatedAt	Дата останнього оновлення задачі

Кінець таблиці 2.1

days	id	Ключ
	value	Значення дня тижня
daysOnTasks	id	Ключ
	taskId	Зовнішній ключ, який вказує на список задач
	dayId	Зовнішній ключ, який вказує на день тижня

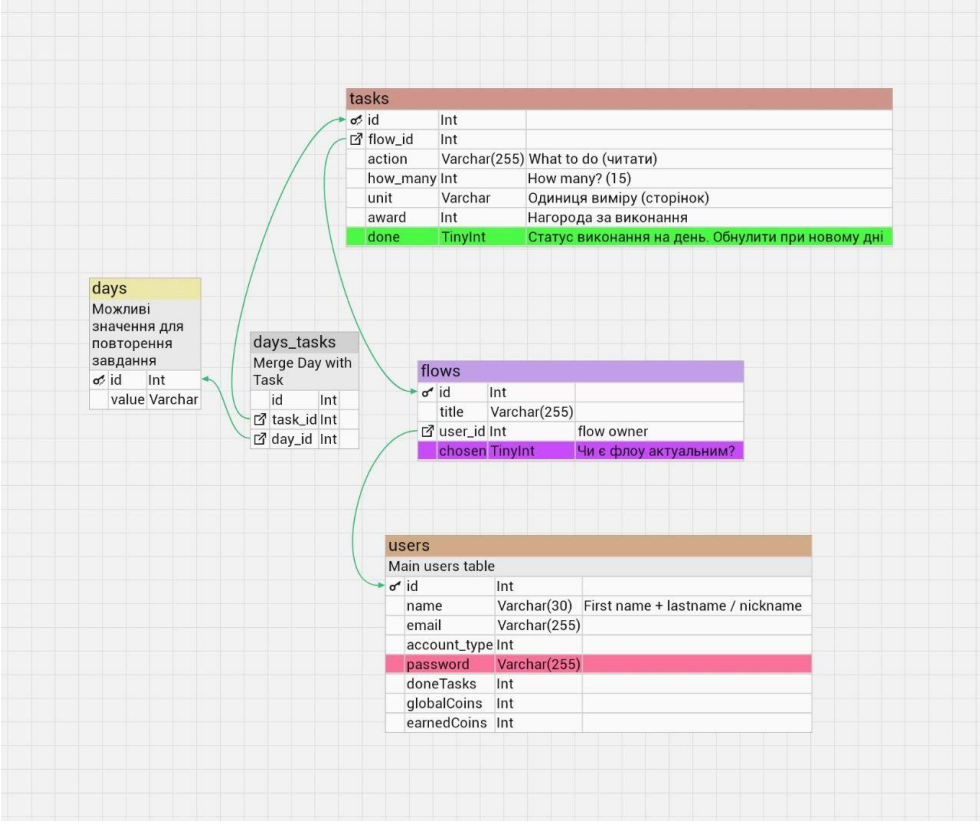

Рисунок 2.9 – Фізична модель бази даних

Фізична модель бази даних включає всі необхідні деталі про структури таблиць, такі як назви стовпців, типи даних, обмеження, первинні та зовнішні ключі, а також зв'язки між таблицями.

Реляційні бази даних широко використовуються як гнучкі рішення загального призначення. Їх можливість обробки складних запитів дозволяє використовувати одну базу даних для різних програм і сценаріїв використання [4].

Висновки до розділу 2

У цьому розділі було розглянуто моделювання та проєктування програмного забезпечення для мобільних застосунків.

Перш за все, було розглянуто важливість дизайну інтерфейсу користувача. візуальний дизайн та інтерактивний дизайн були представлені як ключові компоненти успішного мобільного застосунку. Візуальний дизайн забезпечує привабливість та естетичність мобільного застосунку, в той час як інтерактивний дизайн гарантує зручну та ефективну взаємодію користувача з системою.

Також було розглянуто важливість фізичної моделі бази даних. Фізична модель бази даних визначає структуру таблиць, обмеження, ключі та зв'язки між ними.

Розділ також підкреслив важливість реляційних баз даних як гнучких рішень для мобільного застосунків. Реляційні бази даних дозволяють обслуговувати різні програми та сценарії використання, забезпечуючи гнучкість та узгодженість даних.

У цілому, розділ моделювання та проєктування програмного забезпечення для мобільного застосунків наголосив на важливості розробки ефективного та добро збалансованого рішення, яке поєднує зручний інтерфейс, оптимальну структуру бази даних та гнучкість системи.

3 ВИБІР ЗАСОБІВ РОЗРОБКИ КРОСПЛАТФОРМНОГО МОБІЛЬНОГО ЗАСТОСУНКУ «ТРЕКЕР ЗВИЧОК»

3.1 Вибір технології розробки клієнтської частини мобільного застосунку

З урахуванням основних вимог проєкту, які включають кросплатформеність, продуктивність, доступність та розширюваність, було розглянуто різні можливі варіанти технологій для розробки мобільних застосунків.

Один з основних критеріїв вибору технології була кросплатформеність, що дозволить створити мобільний застосунок, який буде працювати на різних операційних системах, таких як iOS та Android. Для досягнення цієї мети було проаналізовано різні фреймворки та інструменти, зокрема React Native, Xamarin та Flutter.

Flutter

Flutter - це відкрите програмне забезпечення для розробки кросплатформених мобільних застосунків. Він розроблений компанією Google і використовує мову програмування Dart.

Переваги Flutter:

– кросплатформеність: За допомогою Flutter можна розробляти додатки, які працюють на різних платформах, таких як iOS та Android, з одним кодом. Це дозволяє економити час і зусилля при розробці для різних платформ;

– швидкість і продуктивність: Flutter використовує власний движок рендерингу, що дозволяє створювати швидкі та відзивчиві інтерфейси користувача. Це особливо корисно для графічно інтенсивних застосунків;

– гарний зовнішній вигляд: Завдяки своїй власній системі відображення інтерфейсу, відомій як "віджети", Flutter дозволяє створювати привабливі та сучасні

дизайни. Він надає розробникам велику свободу при налаштуванні вигляду та анімацій;

– широкі можливості: Flutter має багатий набір вбудованих віджетів та бібліотек для різних функціональних можливостей, таких як робота з мультимедіа, мережеві запити, геолокація тощо. Також існує активна спільнота розробників, яка надає різноманітні розширення та додатковий функціонал.

Недоліки Flutter:

– розмір застосунків: Застосунки, розроблені на Flutter, можуть мати більший розмір, оскільки вони потребують включення двигуна Flutter у вихідний код застосунку;

– обмеження доступу до платформених функцій: Хоча Flutter надає доступ до багатьох функцій платформи, можуть бути деякі обмеження або потреба в додаткових налаштуваннях для отримання повного доступу до всіх функцій пристрою;

– недостатня зрілість: Flutter - це відносно нова технологія, і в порівнянні з іншими фреймворками, вона може мати меншу кількість документації та ресурсів. Це може ускладнити пошук відповідей на питання та вирішення проблем під час розробки.

Xamarin

Xamarin - це платформа для розробки кросплатформених мобільних застосунків, яка дозволяє використовувати мову програмування C# і фреймворк .NET.

Переваги Xamarin:

– зручна інтеграція з сервісами Microsoft;

– доступ до багатьох платформених API і сервісів, таких як камера, геолокація, мережеві запити тощо.

Недоліки Xamarin:

– великий розмір застосунків;

– обмеження безкоштовної версії: Деякі функції та сервіси Xamarin доступні лише у комерційних версіях.

React Native

React Native є популярним фреймворком для розробки мобільних застосунків, який базується на React - відомій технології розробки веб-інтерфейсів. Основною перевагою React Native є можливість створення кросплатформених застосунків, що працюють як на платформі iOS, так і на Android. Це дозволяє заощадити час та зусилля, оскільки розробка відбувається одноразово для обох платформ [5].

Переваги React Native:

– швидкість розробки: за допомогою React Native можна швидко розробляти кросплатформені застосунки, використовуючи один спільний код на JavaScript для різних платформ. Це дозволяє економити час та зусилля при створенні застосунків для iOS та Android;

– гнучкість: React Native надає зручні інструменти для створення користувацького інтерфейсу, що підтримує нативні компоненти для кожної платформи. Це дозволяє розробникам створювати застосунки з високою продуктивністю та виглядом, що наближений до нативних застосунків;

– широке співробітництво та підтримка: React Native має велику активну спільноту розробників, що приносить внесок у розвиток фреймворку та надає підтримку у вирішенні проблем та пошукові рішення. Крім того, багато компаній та проєктів використовують React Native, що створює ефективну екосистему для розробки;

– гармонійне поєднання з веб-технологіями: розробка на React Native базується на розумінні React та веб-технологій, що робить процес вивчення та перехід

на цей фреймворк легшим для веб-розробників. Існує можливість використовувати спільні компоненти та логіку між веб та мобільними застосунками.

Недоліки React Native:

– обмеження нативного доступу: хоча React Native надає багато нативних компонентів, деякі платформено-специфічні функції можуть бути обмеженими або потребувати створення додаткових модулів;

– продуктивність: Порівняно з нативно розробленими застосункамим, React Native може мати меншу продуктивність, особливо при важких обчисленнях або роботі з графікою.

Отже, в результаті проведеного аналізу, було виявлено, що розробка з використанням фреймворку React Native буде більш доцільною в рамках теми кваліфікаційної бакалаврської роботи. Оскільки обраний фреймворк надає можливість отримати швидке кросплатформне рішення, яке легко підтримується й масштабується.

Вибір React Native також обумовлений його активною спільнотою розробників, яка забезпечує велику кількість готових компонентів та рішень, що спрощують розробку. Крім того, React Native володіє широким спектром можливостей для розробки інтерактивного та швидкого інтерфейсу, забезпечуючи плавну роботу застосунку.

Також варто зазначити, що вибір React Native дає можливість перевикористовувати код та ресурси між різними проєктами, що сприяє швидкості розробки та зменшенню трудомісткості.

З урахуванням всіх цих факторів, React Native був обраний як оптимальна технологія для розробки клієнтської частини мобільного застосунку "Трекер Звичок".

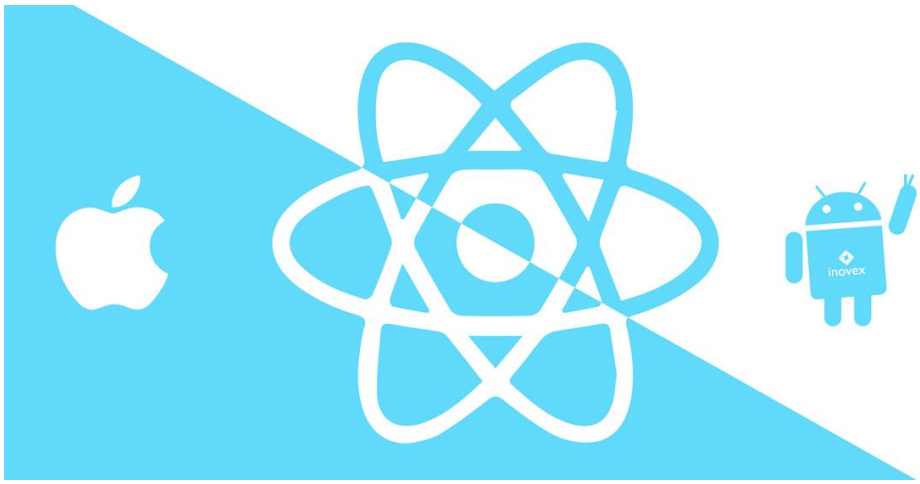

Рисунок 3.1 – Логотип React Native

Для створення інтерфейсів в проєкті були використані додаткові технології, такі як Expo, CSS, JS та бібліотека компонентів UI Kitten.

Expo: Expo є набором інструментів та сервісів, які допомагають спростити розробку мобільних застосунків. Він надає широкий спектр функціональності, включаючи можливість швидко побудувати та перевірити застосунок без необхідності налаштовування середовища розробки для кожної платформи окремо.

CSS (Cascading Style Sheets) використовується для опису зовнішнього вигляду елементів інтерфейсу. За допомогою CSS можна задавати розміри, кольори, шрифти, відступи та інші стилістичні властивості для кожного елемента.

JS (JavaScript) є основною мовою програмування для розробки веб-застосунків і мобільних застосунків. В контексті розробки інтерфейсу, JavaScript використовується для програмування логіки та взаємодії з елементами інтерфейсу.

UI Kitten - це бібліотека компонентів, яка надає набір готових і стилізованих компонентів інтерфейсу, які можна використовувати для швидкого та простого побудови зовнішнього вигляду застосунку. Ця бібліотека забезпечує однорідний та сучасний дизайн, спрощуючи розробку інтерфейсу [6].

3.2 Вибір технологій розробки серверної частини мобільного застосунку

Серверна частина мобільного застосунку є одним з найважливіших компонентів і виступає як його основний скелет, тоді як зовнішній інтерфейс - це просто зовнішній вигляд, створений для користувачів.

Серверна частина - це місце, де зберігаються всі дані та відповідна інформація, яка повинна бути показана користувачам через мобільний застосунок. Вона обробляє всі запити користувачів через свій інтерфейс. Вибір правильної технології для серверної частини є дуже важливим, оскільки це дозволить створити надійну основу для мобільного застосунку.

У контексті теми роботи було обрано фреймворк Nest.js для розробки серверної частини мобільного застосунку. Nest.js є популярним фреймворком, побудованим на основі мови програмування TypeScript, який надає потужні можливості для створення масштабованих та ефективних застосунків.

Вибір Nest.js для бекенду має кілька переваг. По-перше, він пропонує модульну архітектуру, яка дозволяє легко організовувати код та забезпечує високу структурованість проєкту. По-друге, Nest.js надає вбудовану підтримку для розробки API з використанням популярних стандартів, таких як REST та GraphQL. Це спрощує процес створення і взаємодії з API в мобільному застосунку. Крім того, Nest.js підтримує TypeScript, що дозволяє розробникам використовувати типізацію та отримати переваги статичного аналізу коду.

Незважаючи на свої переваги, Nest.js також має певні недоліки. Один з них - вимога до знання TS, що може стати перешкодою для розробників, які не мають досвіду з цією мовою програмування. Крім того, Nest.js може бути важким для вивчення, оскільки він має власні концепції та підходи до розробки [7].

Також варто враховувати, що деякі функціональності Nest.js можуть бути перевищеними для простих мобільних застосунків, що може призвести до зайвого складнощів у розробці.

Рисунок 3.2 – Логотип Nest.js

Загалом, вибір фреймворку Nest.js для розробки бекенду мобільного застосунку зумовлений його модульною архітектурою, підтримкою стандартів API та підтримкою TypeScript.

3.3 Вибір засобів реалізації бази даних мобільного застосунку

Реляційна база даних використовує таблиці для зберігання даних і є найефективнішим способом зберігання складних даних для розробників програмного забезпечення. Деякі популярні реляційні бази даних, які використовуються зараз, включають MySQL, SQL Server, PostgreSQL, Microsoft Access, MariaDB і Azure SQL.

PostgreSQL є системою керування базами даних з відкритим вихідним кодом, яка належить до корпоративного класу. Вона підтримує як мову SQL, так і JSON для реляційних і нереляційних запитів, що дозволяє розширювати можливості і відповідати SQL-стандартам [8].

Переваги PostgreSQL включають:

– можливість запускати динамічні веб-сайти та веб-програми в якості стеку LAMP;

– вільний доступ до вихідного коду під ліцензією з відкритим вихідним кодом, що дозволяє внести зміни та адаптувати базу даних під потреби проєкту;

– низький рівень обслуговування та адміністрування як для вбудованого, так і для корпоративного використання. Це означає менше зусиль, необхідних для підтримки та управління базою даних.

Таким чином, PostgreSQL забезпечує гнучкість, доступність вихідного коду, розширені можливості географічного моделювання та ефективне управління базою даних.

Висновки до розділу 3

В даному розділі було проаналізовано та досліджено актуальні технології для розробки клієнтської та серверної частини застосунку.

В результаті проведеного аналізу, було вирішено використовувати фреймворк React Native для розробки кросплатформеного мобільного застосунку "Трекер звичок". React Native демонструє багато переваг, таких як швидкий розробка, кросплатформеність, широкі можливості налаштування і розширення, а також велика спільнота розробників, що забезпечує підтримку та розвиток фреймворку.

Для реалізації серверної частини було обрано фреймворк Nest Js. Фреймворк володіє надійною та універсальною структурою, пропонує просту і зрозумілу архітектуру, підтримку впровадження залежностей та маршрутизацію за допомогою декораторів. Використання фреймворка дозволить створити міцний ґрунт для бекенду мобільного застосунку.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ КРОСПЛАТФОРМЕНОГО МОБІЛЬНОГО ЗАСТОСУНКУ «ТРЕКЕР ЗВИЧОК»

4.1 Реалізація серверної частини

Розробку серверної частини варто почати з інсталювання всіх необхідних залежностей для проєкту. А саме необхідно інсталювати бібліотеки для коректної роботи застосунку:

– Prisma - це ORM (об'єктно-реляційне відображення), що надає зручні методи взаємодії з базами даних в програмах;

– Passport package - це популярний модуль для аутентифікації та авторизації користувачів розроблений для Node.js. Він надає широкий спектр стратегій аутентифікації, включаючи локальну аутентифікацію, аутентифікацію через соціальні мережі (Facebook, Google, Twitter і т.д.), аутентифікацію на основі токенів та багато іншого.

– class-transformer і class-validator є пакетами для роботи з валідацією та трансформацією об'єктів в TypeScript.

Наступним етапом розробки є генерація всіх необхідним ресурсів для сутностей, які будуть застосовані на проєкті: User, Flow, Task. В Nest CLI [9] є дуже зручний спосіб автоматично створити необхідні класи за допомогою консольної команди:

– nest generate controller <name> - генерація контролеру;

– nest generate service <name> - генерація сервісу;

– nest generate module <name> - генерація модуля.

Таким чином директорія проєкту набуває наступного вигляду:

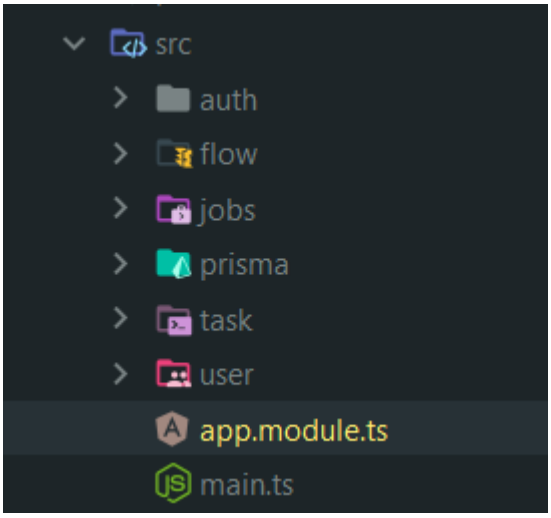

Рисунок 4.1 – Директорія проєкту після генерації необхідних файлів

Після генерації всіх необхідних модулей варто приєднати їх до корневого модуля застосунку:

```
4 usages  Serhii Dychko
@Module( metadata: {
  imports: [
    ConfigModule.forRoot( options: {
      isGlobal: true,
    }),
    PrismaModule,
    AuthModule,
    UserModule,
    TaskModule,
    FlowModule,
    ScheduleModule.forRoot(),
  ],
  providers: [DailyJobService],
})
export class AppModule {}
```

Рисунок 4.2 – Вміст файлу app.module.ts

Далі необхідно описати таблиці бази даних, які будуть застосовані в застосунку. В Prisma ORM існує спеціальний синтаксис для цієї задачі. Скориставшись документацією ORM [10] було створено файл **schema.prisma** в якому описані всі взаємозв'язки між таблицями (рис. 4.3).

```
model User {
    id          String      @id @default(uuid())

    email       String @unique
    password    String

    username    String
    fullname    String?

    flows       Flow[]

    doneTasks   Int @default(0)
    globalCoins Int @default(0)
    earnedCoins Int @default(0)

    createdAt   DateTime @default(now())
    updatedAt   DateTime @updatedAt

    @@map("users")
}
```

Рисунок 4.3 – Приклад опису таблиці для Prisma ORM

Для розробки функціоналу авторизації користувача було вирішено використати підхід JWT (Json Web Token) - це відкритий стандарт, що використовується для передачі безпечної та автентифікованої інформації між сторонами у форматі JSON. JWT складається з трьох частин: заголовка (header), заяви (payload) та підпису (signature) [11]. Заголовок містить тип токена та алгоритм шифрування, заява містить корисну інформацію, а підпис використовується для перевірки цілісності та автентичності токена.

Тобто при логіні користувача необхідно генерувати токен, який він зможе використовувати в майбутньому для виконання авторизованих запитів.

Для того, щоб позначити запити, які вимагають авторизації від користувача можемо використати Auth Guard.

Guard в Nest Js - це механізм, який дозволяє контролювати доступ до маршрутів (routes) та виконувати логіку авторизації для запитів до сервера. Гарди виконуються перед обробкою запиту та можуть перевіряти різні умови, наприклад, наявність прав доступу, наявність валідного токена аутентифікації тощо. Якщо гард повертає false або викидає виключення, запит не буде продовжено до обробки.

При успішному виконанні методу validate (рис. 4.4) буде повернено модель користувача, а в іншому випадку пусте значення, це означає що авторизація є хибною.

```
@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy, name: 'jwt') {
  no usages  Serhii Dychko
  constructor(config: ConfigService, private prisma: PrismaService) {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      secretOrKey: config.get('JWT_SECRET'),
    });
  }

  no usages  Serhii Dychko
  async validate(payload: { sub: string; email: string }) : Promise<User>  {
    const user : User  = await this.prisma.user.findUnique({
      where: {
        id: payload.sub,
      },
    });
    delete user.password;
    return user;
  }
}
```

Рисунок 4.4 – Реалізація JWT Auth Guard

Далі, щоб позначити маршрути, які вимагають авторизації, для контролера потрібно вказати декоратор [12] @UseGuards(JwtGuard). Nest Js автоматично буде перевіряти авторизацію користувача й повертати помилку в разі її хибності.

Наступним етапом було створено методи для реєстрації та входу в систему:

```
1 usage  Serhii Dychko
async signIn(dto: SignInDto) : Promise<{access_token: string}...  {
  // пошук User по його даним
  const user : User  = await this.prisma.user.findFirst({
    where: { email: dto.email },
  });

  // якщо User не був знайдений генеруємо помилку
  if (!user) {
    throw new ForbiddenException( objectOrError: 'Credentials are not correct');
  }

  const isVerified : boolean  = await argon.verify(user.password, dto.password);

  // якщо User увів неправильні дані генеруємо помилку
  if (!isVerified) {
    throw new ForbiddenException( objectOrError: 'Credentials are not correct');
  }

  return this.signToken(user.id, user.email);
}
```

Рисунок 4.5 – Метод, який реалізовує функціонал логіну

Варто зазначити, що виконання цього методу повертає access_token, що і є JWT токеном для користувача. Далі на клієнтській частині цей токен можна помістити в заголовок запиту й автоматично зробити всі наступні запити авторизованими.

Основний функціонал закладений в серверну частину застосунку, це реалізація CRUD операцій для всіх моделей. CRUD означає створення (Create), читання (Read), оновлення (Update) та видалення (Delete). Це набір базових операцій, які використовуються в багатьох базах даних і системах керування даними для маніпулювання інформацією. CRUD надає можливість створювати нові записи, читати та відображати існуючі, оновлювати їх зміст і видаляти записи, що більше не потрібні. На прикладі моделі задачі (рис. 4.6) було відображено принцип взаємодії.

```
@UseGuards(JwtGuard)
@Controller('flows/:flow_id/tasks')
export class TaskController {
  constructor(private service: TaskService) {}

  @Post()
  create(    @GetUser('id') userId: string,    @Param('flow_id') flowId: string,    @Body() dto: CreateTaskDto,  ): Promise<TaskWithDays> {
    return this.service.createTask(flowId, userId, dto);
  }

  @Get(':id')
  findOne(    @GetUser('id') userId: string,    @Param('flow_id') flowId: string,    @Param('id') id: string,  ) {
    return this.service.findOne(id, flowId, userId);
  }

  @Get()
  findAll(@GetUser('id') userId: string, @Param('flow_id') flowId: string) {
    return this.service.findAll(flowId, userId);
  }

  @Patch(':id')
  update(    @GetUser('id') userId: string,    @Param('flow_id') flowId: string,    @Param('id') id: string,    @Body() dto: UpdateTaskDto,  ) {
    return this.service.update(id, flowId, userId, dto);
  }

  @Delete(':id')
  remove(    @GetUser('id') userId: string,    @Param('flow_id') flowId: string,    @Param('id') id: string,  ) {
    return this.service.remove(id, flowId, userId);
  }
```

Рисунок 4.6 – CRUD операції для моделі Task

Всі методи для операцій над моделю Task приймають вхідні параметри userId, який випливає з інформації зашифрованої в JWT, та flowId, який вказує на список задач до якого вносяться зміни.

За схожим принципом CRUD була побудована й для інших моделей, а саме Flow та User.

Останній функціонал, який варто реалізувати на стороні серверу – це оновлення стану для користувача та його завдань. Тобто з початком нового дня потрібно для всіх задач змінити статус виконання на **не виконано** та оновити список виконаних завдань для користувача. Для реалізації цього функціоналу Nest Js пропонує список асинхроних задач, які можуть виконуватись з певною періодичністю:

```
@Cron( cronTime: '0 0 * * *') // Run at 12:00 AM every day
async handleCron() : Promise<void>  {
  console.log('Running midnight job!');
  await this.userService.refresh();
  await this.taskService.refresh();
}
}
```

Рисунок 4.7 – Реалізації асинхроної задачі

Вищевказана задача буде виконуватись щодня опівночі й забезпечить коректну зміну доби для користувача.

4.2 Розробка клієнтської частини застосунку

Встановлення Expo CLI [13] на комп'ютер за допомогою команди:

npm install -g expo-cli.

Було створено новий проєкт React Native за допомогою Expo, виконавши команду **expo init UflowTracker**, де "UflowTracker" назва проєкту. Для запуску середовища розробки існує команда **expo start.** Під час запуску Expo відкриється веб-інтерфейс, де буде відображено QR-код. Далі необхідно завантажити та встановити Expo Go застосунок на свій мобільний пристрій з App Store або Play Store. При скануванні QR-коду з мобільного девайсу маємо отримати базовий застосунок на React Native (рис. 4.8).

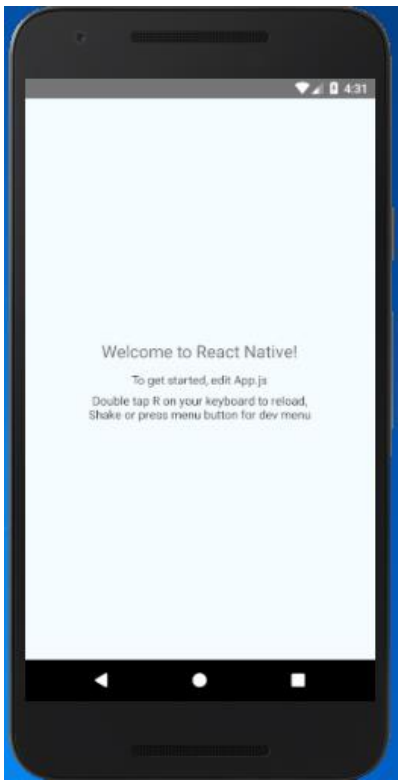

Рисунок 4.8 – Базовий застосунок на React Native

Наступним етапом необхідно інтегрувати бібліотеку з компонентами інтерфейсу - UI Kitten. Виконавши команду **npm install @ui-kitten/components @eva-design/eva** до проєкту буде інстальовано всі необхідні залежності.

Головний файл App.jsx набуває наступного вигляду:

```
no usages  Serhii Dychko *
function App() {
  const isAuth = useUser((state) => state.isAuth);

  return (
    <>
      <IconRegistry icons={EvaIconsPack} />
      <ApplicationProvider {...eva} theme={eva.light}>
        <AppNavigator isAuth={isAuth} />
      </ApplicationProvider>
    </>
  );
}
```

Рисунок 4.9 – Вигляд файлу App.jsx

IconRegistry – компонент, який дозволить використовувати іконки з пакету EvaIcons.

ApplicationProvider – компонент з бібліотеки UI Kitten, необхідний для коректної інтеграції елементів інтерфейсу.

AppNavigator – головний компонент проєкту, в якому відбувається налаштування всіх видових екранів. Як залежність він приймає булеве значення **isAuth.** Ця зміна вказує чи є користувач авторизованим й впливає на доступну навігацію. При успішній авторизації користувач має можливість використовувати

функціонал застосунку, а в іншому разі буде перенаправлений на вікно входу в систему.

Навігація по застосунку відповідно до дизайну матиме вигляд табів. Для цього в React Native є гарне вбудоване рішення – пакет react-navigation. Для його роботи потрібно використати компонент Navigator й в середині помістити необхідні екрани – таби (рис. 4.10).

```
const TabNavigator = () => (
  <Tab.Navigator tabBar={(props) => <BottomTab {...props} />} screenOptions={{ headerShown: false }}>
    <Tab.Screen name="Daily Tap" component={DailyTab} />
    <Tab.Screen name="Flow Tap" component={FlowTab} />
    <Tab.Screen name="Profile" component={ProfileTab} />
  </Tab.Navigator>
);

2 usages  Serhii Dychko
export default TabNavigator;
```

Рисунок 4.10 – Компонент навігації по застосунку

Коли навігація повністю готова, можна почати створювати логіку роботи під кожен видовий екран.

Написання всієї логіки в середині видового екрана є небажаною практикою і суперечить принципу розробки SOLID [14], який ставить за мету забезпечити гнучкість, розширюваність і підтримуваність коду. Тому була організована структура проєкту, де кожна частина відповідає за один конкретний аспект функціональності.

У такій структурі проєкту використовується підхід розділення на шари (layered architecture) або модульну архітектуру (module architecture).

Зазвичай ці шари або модулі включають:

– представлення (Presentation Layer): Відповідає за відображення і взаємодію з користувачем. Сюди входять компоненти і екрани, в яких відбувається відображення даних та взаємодія з користувачем. Однак, логіка, пов'язана зі станом компонентів, винесена в окремий шар;

– логіка (Business Logic Layer): Цей шар містить бізнес-логіку вашого застосунку. Він відповідає за обробку даних, взаємодію з сервером або базою даних, а також за правила обробки даних. Логіка виконується незалежно від представлення, що дозволяє її перевикористовувати і тестувати окремо від інтерфейсу;

– доступ до даних (Data Access Layer): Відповідає за отримання даних з різних джерел, таких як сервер або локальна база даних. В цьому шарі реалізовані класи або модулі, що забезпечують доступ до даних.

Ця структура допомагає забезпечити принцип SOLID, де кожна частина відповідає за свою конкретну відповідальність, що полегшує розширення і тестування проєкту. В результаті було отримано наступну структуру проєкту:

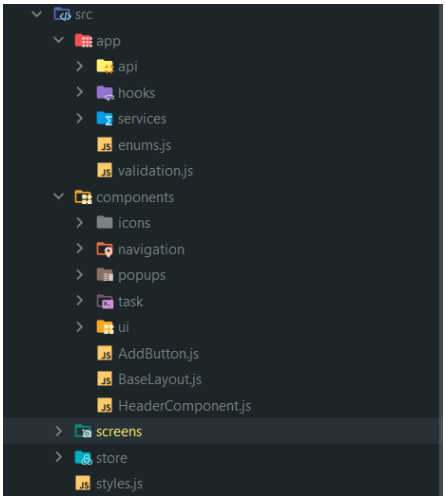

Рисунок 4.11 – Структура проєкту відповідно до принципів SOLID

Основна структура директорій та файлів у проєкті має організовані елементи відповідно до принципів SOLID. Розглянемо кожен з них:

– **app**: головна директорія проєкту, що містить усі компоненти і функціональні блоки застосунку;

– **api**: директорія, яка містить модулі або класи, що відповідають за взаємодію з веб-сервером або зовнішніми API;

– **hooks**: директорія, де розташовуються кастомні хуки, які використовуються для спільного використання логіки між компонентами;

– **services**: директорія, в якій можуть знаходитися різноманітні сервіси або утилітарні класи, які виконують спеціалізовані функції, такі як обробка даних або розрахунки;

– **enums.js:** файл, що містить перерахування (enumerations), які використовуються для представлення іменованих станів або варіантів у вашому застосунку;

– **validation.js:** файл, який містить функції або класи для валідації даних, що вводяться або передаються в застосунку;

– **components**: директорія, де знаходяться універсальні компоненти, які можуть бути використані в різних частинах застосунку;

– **icons**: директорія, де знаходяться компоненти для відображення та використання іконок;

– **navigation**: директорія, в якій містяться компоненти та утиліти, що відповідають за навігацію між екранами застосунку;

– **popups**: директорія, що містить компоненти або модулі, пов'язані з вікнами спливаючих повідомлень або сповіщень;

– **task**: директорія, де знаходяться компоненти, що відповідають за відображення та управління окремими завданнями або функціональними блоками;

– **ui**: директорія, де знаходяться різні компоненти, які використовуються для побудови інтерфейсу користувача, такі як кнопки, поля вводу, тощо;

– **screens**: директорія, що містить компоненти або модулі, пов'язані з окремими екранами застосунку;

– **store**: директорія, де знаходиться логіка стану застосунку.

Ця структура директорій допомагає зберігати відокремлені елементи проєкту відповідно до SOLID, де кожна частина відповідає за свою конкретну відповідальність та може бути легко розширена, підтримана і перевикористана у майбутньому.

4.3 Демонстрація використання та тестування функціоналу мобільного застосунку

Для користування застосунком необхідно мати мобільний пристрій з операційною системою Android версії 6 і вище, або IOS версії 13 і вище. Крім того, використання клієнтської частини застосунку передбачає наявність інтернет-підключення, оскільки вона потребує доступу до серверу. Завдяки функціоналу Expo, було створено два файли для запуску застосунку на відповідних операційних системах. Це дозволяє забезпечити сумісність з різними мобільними платформами і спрощує процес розгортання та використання застосунку для користувачів (рис. 4.12).

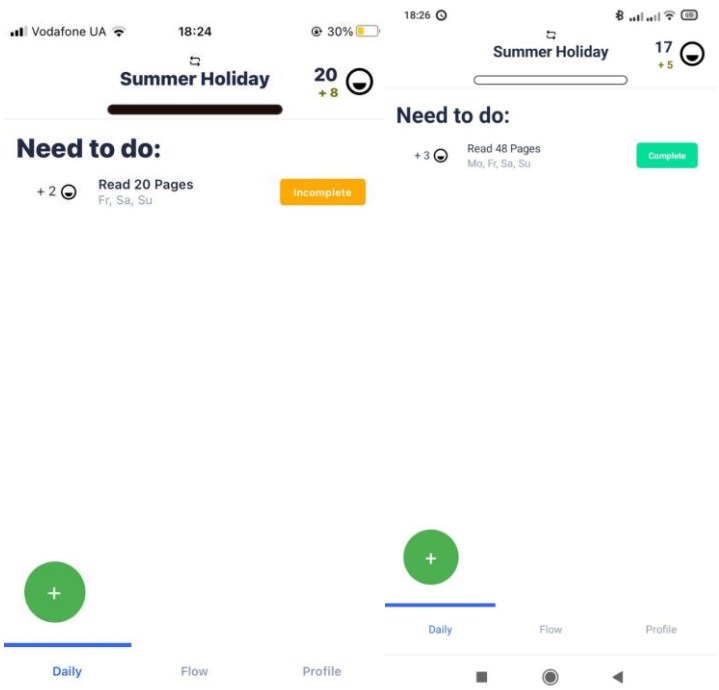

Рисунок 4.12 – Демонстрація кросплатформеної роботи застосунку (ліворуч – IOS, праворуч - Android)

Зображення нагадує про те, як авторизація на одному акаунті користувача дозволяє синхронізувати його дані між різними пристроями. Це означає, що користувач може отримати доступ до своїх персоналізованих налаштувань, даних та взаємодіяти зі своїм обліковим записом з будь-якого пристрою з встановленою програмою.

Надалі в демонстрації показано основний функціонал застосунку. Користувач може виконувати різні дії, такі як створення, редагування та видалення записів, взаємодію зі списками, аналізувати власний прогрес та використання інших доступних функцій. Все це робить застосунок повноцінним і зручним інструментом для досягнення своїх цілей та організації своїх звичок.

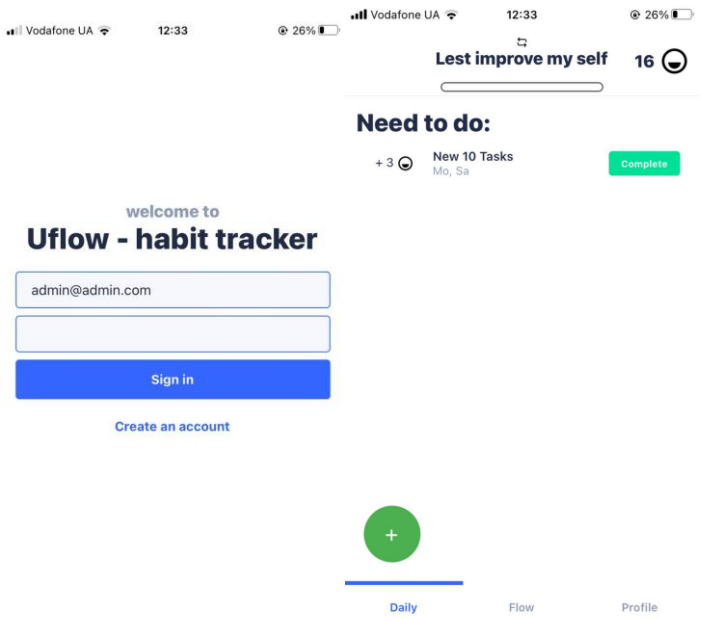

Рисунок 4.13 – Демонстрація процесу входу в систему

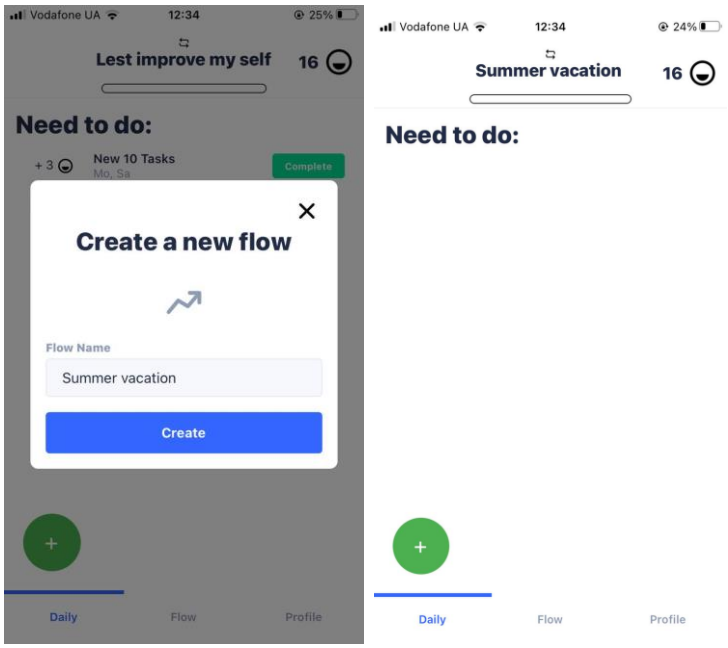

Рисунок 4.14 – Демонстрація процесу створення нового списку задач

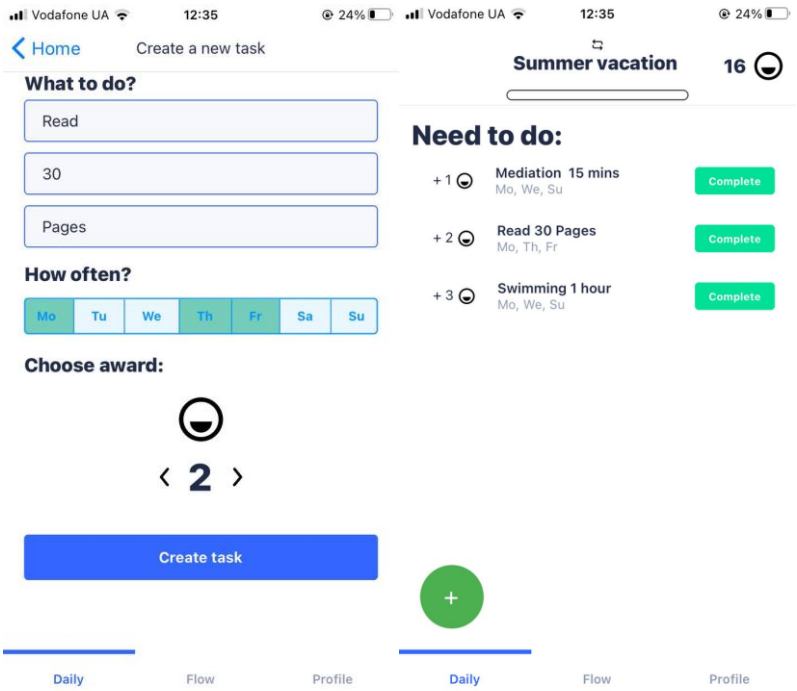

Рисунок 4.15 – Демонстрація додавання задачі

При натисканні на елемент «Додати», відбувається перехід на видовий екран, який надає можливість детально налаштувати звичку та створити її.

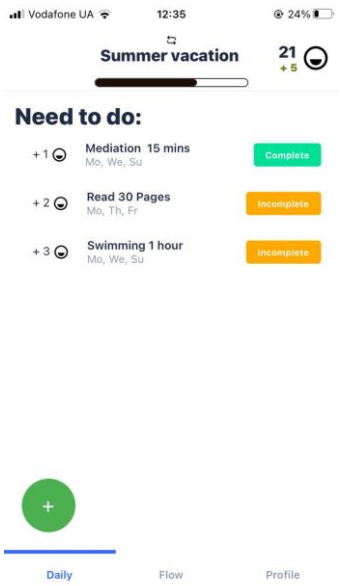

Рисунок 4.16 – Демонстрація позначення задачі як виконаної

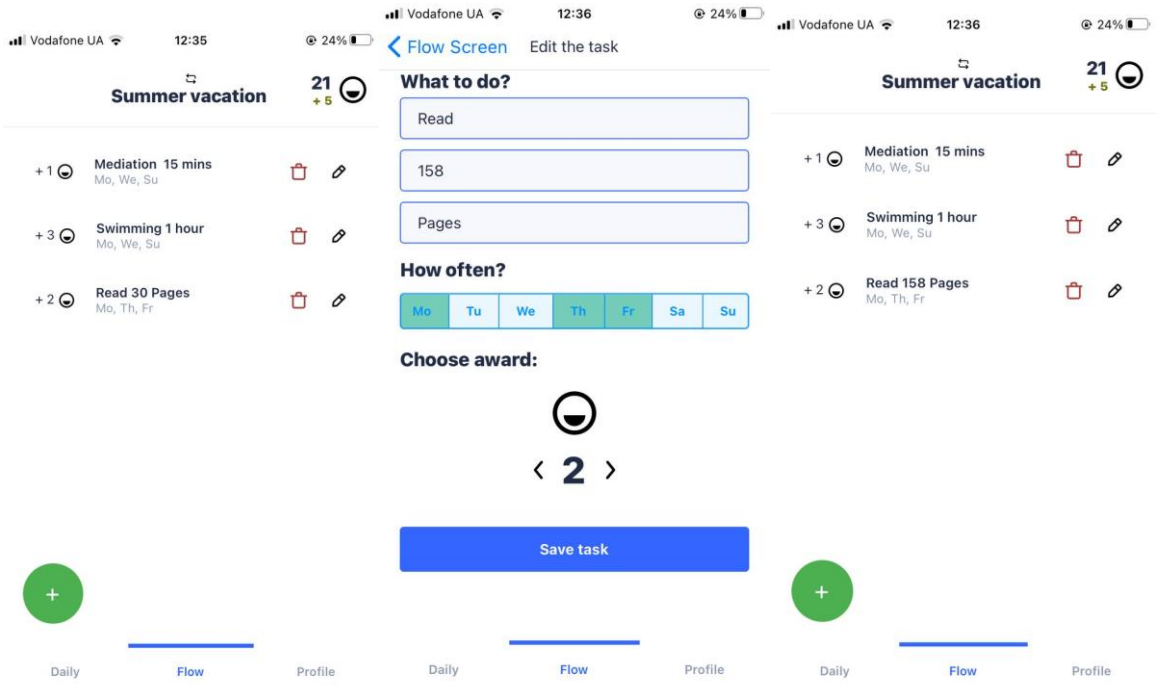

Рисунок 4.17 – Демонстрація редагування задачі

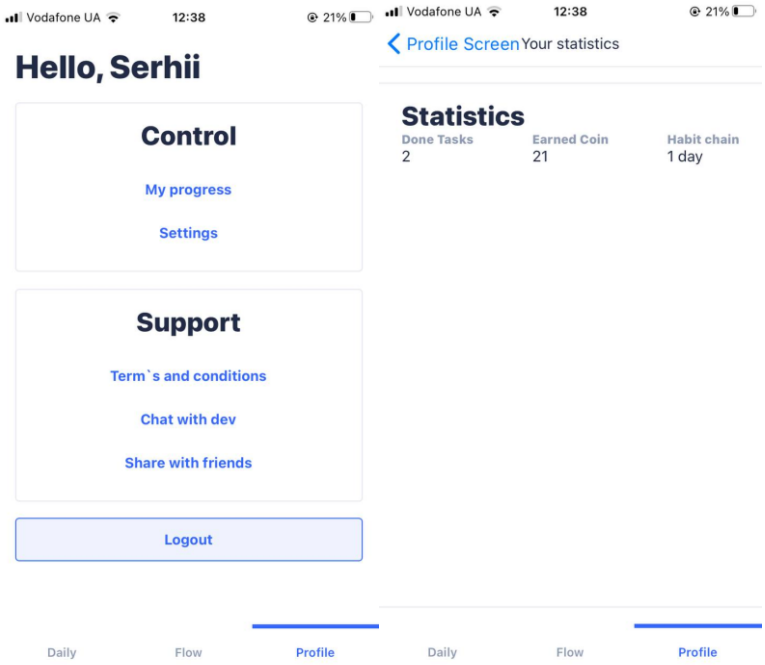

Рисунок 4.18 – Демонстрація перегляду статистики

Демонстрація використання та тестування функціоналу мобільного застосунку підтверджує його ефективність та готовність до використання користувачами. Через можливість синхронізації даних та широкий спектр доступних опцій, застосунок є потужним інструментом для досягнення мети та покращення звичок користувача.

Висновки до розділу 4

У цьому розділі було реалізовано застосунок, з використанням технологій для серверної та клієнтської частини, дотримуючись вимог до застосування, спроєктованих моделям бази даних та користувальницьких сценаріїв використання системи.

Програмна реалізація мобільного застосунку «Трекер звичок» показує успішну інтеграцію між клієнтською та серверною частинами. Застосунок забезпечує зручний інтерфейс для користувачів, дозволяє створювати, відстежувати та аналізувати свої звички.

Технологічний стек, що використовується у реалізації застосунку, дає можливість побудувати масштабований та надійний застосунок з кросплатформеною підтримкою. React Native і NestJS є потужними інструментами для розробки мобільних застосунків та серверних застосунків відповідно.

Усе це дозволяє користувачам насолоджуватися функціональністю та зручним інтерфейсом мобільного застосунку «Трекер звичок» на різних платформах, забезпечуючи зручну інструмент для досягнення своїх цілей та формування позитивних звичок.

ВИСНОВКИ

У результаті виконання роботи була досягнута мета покращення доступності та зручності користування застосунок "Трекер звичок" для мобільних пристроїв. Робота передбачала розробку функціоналу, що дозволяє користувачам стежити за своїми звичками та досягати своїх цілей.

В процесі дослідження та аналізу актуальних технологій були вибрані відповідні засоби для реалізації клієнтської та серверної частини застосунку. З використанням фреймворку React Native було розроблено зручний та ефективний інтерфейс для мобільних пристроїв. Бекенд застосунку був побудований з використанням фреймворку NestJS, що забезпечило потужний та надійний сервер для обробки запитів та збереження даних.

У роботі були визначені основні функціональні вимоги до застосунку, зокрема планування дизайну, розробка інтерфейсу, збереження та аналіз даних про звички. За допомогою відповідно спроєктованої архітектури та використання сучасних патернів розробки, було реалізовано функціональність, що дозволяє користувачам створювати, відстежувати та аналізувати свої звички.

Кросплатформений застосунок "Трекер звичок" відкриває нові можливості для користувачів, допомагаючи їм досягати поставлених цілей та формувати позитивні звички.

Виконана робота підтверджує важливість кросплатформеного розробки та використання сучасних технологій для створення мобільних застосунків, що задовольняють потреби користувачів. Результати роботи можуть бути використані для подальшого вдосконалення та розширення функціональності застосунку "Трекер звичок".

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dawn Griffiths. Head First Android Development: A Brain-Friendly Guide 1st Edition, 2015. 736 p.

2. Steve Hoberman. Data Modeling Made Simple, 2nd Edition: A Practical Guide for Business and IT Professionals, 2009. 244 p.

3. Стівен Хеллер, Мирко Іліч. Анатомія дизайну. Приховані джерела сучасного графічного дизайну, 2008. 104 с.

4. C. J. Date. An Introduction to Database Systems, 2003. 375 p.

5. The Benefits of React Native for Mobile App Development: online. URL: https://www.cleart.com/the-benefits-of-react-native-for-mobile-app-development.html (Date accessed 1 May 2023).

6. UI Kitten Docs: online. URL: https://akveo.github.io/react-native-ui-kitten/docs/getting-started/what-is-ui-kitten (Date accessed 1 May 2023).

7. Nest.js — Architectural Pattern, Controllers, Providers, and Modules: online. URL: https://medium.com/geekculture/nest-js-architectural-pattern-controllers-providers-and-modules-406d9b192a3a (Date accessed 5 May 2023).

8. Advantages of PostgreSQL: online. URL: https://www.cybertec-postgresql.com/en/postgresql-overview/advantages-of-postgresql/ (Date accessed: 6 May 2023).

9. Nest Js Documentation: online. URL: https://docs.nestjs.com/ (Date accessed 6 May 2023).

10. Prisma Documentation: online. URL: https://www.prisma.io/docs (Date accessed 9 May 2023).

11. JSON Web Token: online. URL: https://en.wikipedia.org/wiki/JSON_Web_Token (Date accessed 21 May 2023).

12. TypeScript. Decorators: online. URL: https://www.typescriptlang.org/docs/handbook/decorators.html (Date accessed 15 May 2023).

13. Expo CLI Documentation: online. URL: https://docs.expo.dev/more/expo-cli/ (Date accessed 17 May 2023).

14. Robert C. Martin. Getting a SOLID start, 2000. 280 p.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

Охорона праці

на тему:

«КРОСПЛАТФОРМЕНИЙ ЗАСТОСУНОК «ТРЕКЕР ЗВИЧОК» ДЛЯ МОБІЛЬНИХ ДЕВАЙСІВ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910110

Виконав студент 4-го курсу, групи 401

_____ С. А. Дичко

«18» червня 2023 р.

Консультант: канд. техн. наук, доцент

_____А. О. Алексєєва

«18» червня 2023 р.

Миколаїв – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

дБА – децибел

ПК – персональний комп’ютер

ДСН – державні санітарні норми

ЛК – одиниця виміру люкс

МОЗ – міністерство охорони здоров’я

ДСН – державні санітарні норми

МЮУ – Міністерство юстиції України

ВСТУП

У зв'язку з постійним розвитком інформаційних технологій, питання охорони праці стає все більш актуальним у сфері розробки мобільних застосунків. Використання технічних пристроїв, таких як мобільні пристрої та комп'ютери, сприяє автоматизації робочих процесів, але водночас ставить під загрозу здоров'я та безпеку працівників.

Охорона праці є необхідною системою на будь-якому підприємстві, незалежно від його форми власності. Дотримання правил безпечного виконання робіт та вимог охорони праці може значно знизити ризик виробничого травматизму. Однак, лише врахування технічних пристроїв і інформаційних технологій недостатньо. Необхідно також забезпечити сприятливі умови праці та запобігти негативному впливу комп'ютерів на здоров'я користувачів.

Для забезпечення безпеки та здоров'я працівників у сфері розробки мобільних застосунків, слід дотримуватися відповідних законодавчих норм України, таких як Закон про охорону праці, Кодекс законів про працю та інші нормативно-правові акти. Крім того, важливо розробити і впровадити заходи для усунення або обмеження ризиків на робочому місці програміста.

Метою даного розділу є аналіз умов праці, технологічних процесів, апаратури та обладнання на робочому місці програміста. На основі цього аналізу визначаються небезпечні ділянки, оцінюється відповідність умов праці нормативним вимогам та розробляються заходи для запобігання аварійним ситуаціям та зменшення їх наслідків.

Дотримання вимог охорони праці у сфері розробки кросплатформених мобільних застосунків допоможе забезпечити безпеку та здоров'я працівників, знизити ризики виробничого травматизму та створити сприятливі умови для ефективної роботи.

1 Техніка безпеки під час роботи з ПК

Під час виконання робіт на комп'ютерах необхідно дотримуватись вимог загальних правил та інструкцій з охорони праці. Для самостійної роботи на комп'ютерах допускаються лише особи, які відповідають наступним вимогам: пройшли медичний огляд, отримали навчання з професії, пройшли вступний інструктаж з охорони праці та отримали первинні інструктажі з охорони праці на робочому місці. Після цього проводяться регулярні повторні інструктажі з охорони праці на робочому місці один раз на півріччя, а також періодичні медичні огляди один раз на два роки.

Основне обладнання робочого місця користувача комп'ютера включає монітор, системний блок і клавіатуру. Розміщення робочих місць має відповідати певним вимогам: відстань між робочим місцем і стіною з вікнами повинна бути не менше 1,5 метра, відстань від інших стін - 1 метр, а між робочими місцями - не менше 1,5 метра. Рекомендується розташовувати робоче місце відносно вікон таким чином, щоб природне світло падало на нього збоку, переважно зліва [1].

Уникайте використання фільтрів з металевою або нейлоновою сіткою, оскільки вони можуть спотворити зображення через інтерференцію світла. Найкращою якістю зображення забезпечують скляні поляризаційні фільтри, які усувають більшість відблисків і роблять зображення чітким і контрастним.

При роботі з текстовою інформацією (введення даних, редагування тексту, читання з екрану) найбільш фізіологічним і правильним є зображення чорних знаків на світлому (чорному) фоні.

Монітор повинен бути розташований на робочому місці так, щоб поверхня екрана знаходилась в центрі поля зору на відстані 400-700 мм від очей користувача. Рекомендується розташовувати елементи робочого місця таким чином, щоб відстань між очима та екраном, клавіатурою та текстом була однаковою.

Для нейтралізації статичної електрики в приміщенні, де працюють з комп'ютерами, включаючи лазерні та світлодіодні принтери, рекомендується збільшувати вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити синтетичний одяг, оскільки він сприяє накопиченню статичної електрики.

Згідно статті 18 Закону України «Про охорону праці» працівник зобов'язаний:

– знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;

– дотримуватись зобов'язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства;

– співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров'ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу.

Вимоги безпеки перед початком роботи [2]:

– увімкнути систему кондиціювання в приміщенні;

– перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран - під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;

– перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;

– відрегулювати освітленість робочого місця;

– відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;

– приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

– ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

– відрегулювати яскравість свічення монітора, мінімальний розмір світної точки, фокусування, контрастність. Не слід робити зображення надто яскравим, щоб не втомлювати очей.

Забороняється самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера, 1 раз на півроку повинні відкривати процесор і вилучати пилососом пил і бруд, що накопичилися; класти будь-яку предмети на апаратуру комп'ютера; закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

2 Аналіз умов роботи за шумом

Будівельні процеси виробництва часто включають шум, який виникає від роботи машин і енергетичних установок. Шум – це неприємні звуки різної інтенсивності і частоти, які заважають працювати, сприймати звукові сигнали та відпочивати. Відомо, що шум негативно впливає на здоров'я людей, знижує працездатність, може спричиняти захворювання органів слуху, ендокринної, нервової та серцево-судинної систем. Він також може збільшувати кров'яний

тиск, спричиняти головні болі, погіршувати пам'ять і сприйняття звукових та світлових сигналів небезпеки, що призводить до більш високого ризику травм.

Шум є формою фізичного забруднення природного середовища, і його важко адаптуватися. Тому шум вважається серйозним забруднювачем, який повинен бути контрольований та обмежений законодавством. Вплив шуму включає зниження працездатності робітників, зниження уваги до небезпеки та виклик втоми. Шум також може стати джерелом стресу, який негативно впливає на всі органи і системи організму. При тривалому впливі шуму можуть виникати проблеми зі слухом, центральною нервовою та серцево-судинною системами. Вплив шуму поділяється на специфічний, що впливає на органи слуху, і неспецифічний, що впливає на інші органи і системи. Зниження слуху є однією з найбільш поширених проблем, пов'язаних з шумом.

Результати розрахунку рівня шуму порівнюються з допустимим значенням для даного робочого місця. Якщо рівень шуму перевищує допустиме значення, необхідно вжити спеціальних заходів для його зниження. Рівні звукового тиску джерел шуму, що впливають на оператора на його робочому місці, можна знайти в таблиці 1.1.

Таблиця 1.1 – рівні звукового тиску різних джерел [3].

Джерело шуму	Рівень шуму, дБА
Жорсткий диск	40
Вентилятор	45
Клавіатура	10
Принтер	42
Сканер	45

3 Аналіз умов праці за освітленням

Для кожного виду діяльності, який вимагає розрізнення об'єктів, необхідний певний рівень освітленості на робочій ділянці. Зазвичай, чим складніше завдання, пов'язане зі зоровим сприйняттям, тим вище має бути середнє освітлення. Недостатня освітленість та недостатня контрастність робочої зони можуть призвести до напруження зорового аналізатора, що в свою чергу може спричинити проблеми зі зором. В ситуаціях, коли загальне освітлення відсутнє, роботу може бути неможливо виконати без індивідуальних світильників на голові або в руках.

Для освітлення виробничих приміщень та робочих майданчиків застосовують природне світло від небосхилу (пряме або відбите), штучне світло або їх комбінацію. Залежно від джерела освітлення, конструкції та функціонального призначення виділяють різні типи освітлення.

Видиме випромінювання – це діапазон електромагнітних хвиль від 380 до 770 нм, які людське око сприймає [4].

Штучне освітлення застосовується у приміщеннях, де не вистачає природного світла, а також для освітлення приміщень у години, коли природне освітлення недоступне.

Робочі приміщення, де працюють робітники з персональними комп'ютерами або ноутбуками, повинні відповідати певним вимогам, які перевіряють відповідні служби для забезпечення дотримання вимог щодо охорони праці.

Нижче наведено частину вимог з документу про правила охорони праці під час роботи на підприємствах [5]:

- площа приміщення не повинна бути менша ніж 6 метрів на 1 робоче місце;

− місця робітників повинні бути розташовані не менше 1 метра від стіни з вікном і 1,4 метра від звичайної стіни;

− відстань між комп'ютерами має бути не менша ніж 1,2 метри;

− відстань між тильною стороною комп'ютера та екранним пристроєм має бути не менша 2,5 метрів;

− робочі місця заборонено облаштовувати у підвальних приміщенням або цокольних приміщеннях будинків;

− під час обладнання приміщень забороняється використовувати полімерні матеріали, що виділяють у повітря шкідливі хімічні речовини;

− підлога має бути з рівною, матовою, неслизькою, з антистатичними властивостями [6].

Існують і вимоги до виробничого приміщення щодо параметрів мікроклімату та вібрації. Ці вимоги допомагають зменшити вплив негативних факторів на здоров'я працівника.

Температурні показники повітря в робочій зоні не повинні перевищувати нормальних значень оптимальної температури для даної категорії робіт, зазначених у таблиці 1, по висоті і по горизонталі.

Таблиця 1.2 – Норми мікроклімату для приміщень [7]

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

4 Пожежна безпека

Основними спрямуваннями забезпечення безпеки від пожежі є усунення умов, що можуть спричинити виникнення пожежі, а також зменшення наслідків в разі її виникнення. Пожежа виникає, коли присутні одночасно горюча речовина, джерело запалення та окисник (кисень, повітря), що створюють горюче середовище. Якщо будь-який з цих чинників буде вилучений або заблокований, пожежі не станеться. На цьому базуються основні принципи запобігання пожежам та методи їх гасіння.

Близько 90% пожеж виникають через: недбале поводження з вогнем; порушення правил монтажу та експлуатації електроустаткування і побутових електроприладів; порушення правил монтажу та експлуатації систем опалення і теплогенеруючих установок; підпали; недбалість дітей щодо вогню; несправність промислового обладнання. Основні нормативні документи, які регулюють роботу щодо забезпечення пожежної безпеки об'єктів, включають Закон України "Про пожежну безпеку", стандарти, будівельні норми та правила, правила пожежної безпеки та інші. Керівники підприємств та інші посадові особи мають обов'язок забезпечити пожежну безпеку об'єкту та окремих ділянок виробництва.

Встановлено порядок створення та функціонування пожежно-технічних комісій та добровільних пожежних дружин. Затверджені типові положення про пожежно-технічні комісії та добровільні пожежні дружини. Визначені обов'язки членів добровільних пожежних дружин щодо запобігання пожежам та гасіння. Також передбачені пільги та заохочення для членів добровільних пожежних дружин.

Громадяни, посадові особи та юридичні особи несуть кримінальну, адміністративну, матеріальну та дисциплінарну відповідальність за порушення вимог пожежної безпеки та спричинення пожежі.

Технологічний процес, сировина, готова продукція, агрегати, установки та інші елементи виробництва мають свою пожежну небезпеку. Надано коротку характеристику виробництва та пожежної небезпеки цих елементів.

Заходи пожежної безпеки, які необхідно дотримуватися перед початком роботи, під час роботи та після її завершення, мають на меті запобігання пожежам.

Територія підприємства, протипожежні прогалини, джерела протипожежного водопостачання та протипожежний режим на об'єкті мають відповідати вимогам пожежної безпеки.

Будівлі та приміщення, електрообладнання, опалювальні прилади, системи вентиляції, вогневі роботи, фарбування, знежирення та миття виробів та обладнання мають відповідати основним вимогам пожежної безпеки.

ВИСНОВКИ

Під час написання спеціальної частини з охорони праці були проаналізовані умови праці людей на підприємствах та в установах, а також особливості роботи з електронними пристроями.

У процесі нашої роботи було виявлено сприятливі умови для здійснення роботи з мобільним застосунком, забезпечуючи комфортне самопочуття та збереження стану здоров'я. При цьому був проведений аналіз щодо пожежної безпеки та використання безпекових технік при роботі з персональними комп'ютерами.

Розглянуті аспекти безпеки, пов'язані з експлуатацією комп'ютерної техніки, а також виявлені основні чинники, що впливають на пожежну безпеку при використанні ПК. Ця інформація допоможе визначити необхідні заходи та рекомендації для забезпечення безпеки працівників у процесі роботи з електронними пристроями та використання веб-застосунків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Закон України за 25 квітня 2018р №508/31960 про затвердження вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 22.05.2023).

2. Типовий тематичний план і прикладна програма навчання посадних осіб і фахівців з питань охорони праці. URL: https://forca.com.ua/instrukcii/ohorona-praci/obuchenie-po-voprosam-ohrany-truda-pozharnoi-bezopasnosti-i-tehnicheskoi-ekspluatacii_8.html (Дата звернення: 22.05.2023).

3. Яковенко, М. Ю. Охорона праці в Україні : підручник / М. Ю. Яковенко, Г. Ю. Яковенко. Київ : Центр учбової літератури, 2017. 720 с.

4. Значення освітлення для успішної трудової діяльності. URL: https://subj.ukr-lit.com/osnovi-oxoroni-praci-zhideckij-v-c-2-6-1-znachennya-osvitlennya-dlya-uspishno%D1%97-trudovo%D1%97-diyalnosti/ (Дата звернення: 23.05.2023).

5. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями: вебсайт URL: https://www.sop.com.ua/article/758-vimogi-shchodo-bezpeki-ta-zahistu-zdorovya-pratsvnikv-pd-chas-roboti-z-ekrannimi-pristroyami (дата звернення: 23.05.2023).

6. Порядок проведення медичних оглядів працівників певних категорій: вебсайт. URL: https://zakon.rada.gov.ua/laws/show/z0846-07 (дата звернення: 25.05.2023).

7. Санітарні норми мікроклімату виробничих приміщень: вебсайт. URL: https://www.sop.com.ua/article/ru/225-qqq-16-m4-13-04-2016-sanitarnye-normy-mikroklimata-proizvodstvennyh-pomeshcheniy (Дата звернення: 25.05.2023).