

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до переддипломної практики

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ
РОЗПОДІЛЕНОЇ КОМАНДИ
APPLICATION DEVELOPMENT FOR ARRANGING DISTRIBUTED TEAM
WORK

Виконав: студент(ка) 4 курсу, групи КНТ-129Т

Спеціальності 121 Інженерія програмного

забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Селетков Є.А.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

?

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
« » 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Селєткова Євгена Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення застосунку для організації роботи розподіленої команди. Application Development for Arranging Distributed Team Work

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 07 червня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Аналіз інструментів. 3. Розробка програми. 4. Експлуатація та тестування програми. 5. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Каплієнко Т.І., к.н.т, доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 31 ” січня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 2
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Селєтков Є.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Каплієнко Т.І.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 185 с., 7 табл., 42 рис., 2 дод., 18 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, РОЗПОДІЛЕНА КОМАНДА, ANDROID, iOS, C#, XAMARIN.

Об'єкт дослідження – внутрішній устрій розподіленої команди та принцип її роботи.

Предмет дослідження – застосунки для організації роботи розподіленої команди.

Мета роботи – створення програмної системи для ефективної організації розподіленої команди.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування C#, смартфон з операційною системою Android або iOS.

Результати. Створено мобільний кросплатформовий застосунок для операційних систем Android та iOS, у якому можна ефективно організовувати роботу розподіленої команди.

Висновки. Розроблено мобільний застосунок для ефективної організації робочого процесу розподіленої команди.

Галузь використання – ІТ компанії з розподіленими командами.

ABSTRACT

Explanatory note to the bachelor's thesis: 185 pages, 7 tables, 42 figures, 2 appendices, 18 sources.

MOBILE APPLICATION, DISTRIBUTED TEAM, ANDROID, iOS, C#, XAMARIN.

Object of study is internal arrangement of the distributed team and the principle of its work.

Subject of study is applications for distributed team work organization.

The purpose of the work is creation of a software system for the effective distributed team organization.

Materials, methods and technical means: object-oriented programming, C# programming language, smartphone with Android or iOS operation system.

Results. The mobile cross-platform application for Android or iOS operation system has been created, which helps with effective divided team organization.

Conclusions. The mobile application for efficient distributed team workflow organization has been developed.

The field of use is IT-companies with distributed teams.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Опис об'єкта дослідження	11
1.2 Опис предмета дослідження	13
1.2.1 Atlassian Jira	13
1.2.2 Trello	15
1.2.3 Slack	16
1.2.4 GitHub	16
1.2.5 Hurma	17
1.3 Постановка завдання роботи	18
2 Аналіз інструментів	20
2.1 Порівняльний аналіз мов програмування	20
2.2 Порівняння середовищ розробки	27
2.3 Архітектура програми	30
2.4 Висновки з розділу 2	31
3 Розробка програми	32
3.1 Класи для роботи з даними	33
3.1.1 Models	33
3.1.2 Services	34
3.2 Інтерфейс	36
3.3 ViewModel	56
3.4 Висновки з розділу 3	57
4 Експлуатація та тестування програми	58
4.1 Мінімальні вимоги до апаратних та програмних засобів	58
4.2 Керівництво користувача	58
4.3 Тестування	60
Висновки	64

Перелік джерел посилання	65
Додаток А Текст програми	67
Додаток Б Слайди презентації	176

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

Android	– мобільна операційна система;
API	– Application Programming Interface;
DI	– Dependency injection.
IL	– Intermediate language;
iOS	– мобільна операційна система для пристроїв від компанії Apple;
IT	– інформаційні технології;
JIT	– Just-in-Time;
MVVM	– Model-View-ViewModel;
.NET	– багатомовна платформа для розробки програмного забезпечення;
NuGet	– система управління пакетами для платформ розробки Microsoft;
Xamarin	– платформа для розробки кросплатформових за стосунків на мові програмування C#;
XAML	– eXtensible Application Markup Language;
ОС	– операційна система;
БД	– база даних;
фреймворк	– програмна платформа, або комплексна бібліотека, яка спрощує розробку складних систем;
ПП	– програмний продукт;

ВСТУП

В той час коли для багатьох організацій вимушений перехід на роботу у дистанційному режимі став неабияким викликом, у світі вже багато років існують компанії без прив'язки до фізичних офісів [1].

Stack Overflow, Zapier, Stripe, Automattic, InVision, GitHub та багато інших компаній завжди працювали віддалено та мали розподілені команди. Це не стало на заваді розробити якісні програмні продукти та досягти успіху на ринку [1].

Ці компанії навіть не маючи прямого офлайн зв'язку в команді змогли організувати ефективний робочий процес та тримають усе під контролем. Управління розподіленими командами досягається не в останню чергу за допомогою спеціальних програмних засобів. Це й різні програми для персональних комп'ютерів, і мобільні застосунки, які можна встановити як на смартфони, так і на планшети з операційними системами (ОС) Android та iOS, і різні онлайн сервіси, які не потрібно встановлювати, а лише зайти на сторінку через браузер та авторизуватися. Всі ці програмні засоби допомагають спілкуватися у текстовому та відеочаті, створювати задачі для кожного члена команди та відстежувати процес їх виконання, надавати доступ до програмного коду, щоб кожен розробник в команді міг відстежити, що було написано, коли та ким, надавати відпустки та лікарняні, отримувати заробітну плату та багато чого іншого [1].

Але навіть маючи необхідне програмне забезпечення, може залишатися проблема комунікації. Найбільш поширеною формою спілкування є текстова. Але текст може не правильно передавати суть. Одну й ту саму фразу можна трактувати багатьма різними способами. Можливі непорозуміння, постійні правки, а також конфлікти тільки тому, що якийсь із співробітників не дочитав листа, або зрозумів його неправильно, або навіть пропустив його серед безлічі інших повідомлень у командному чаті. Тому в багатьох компаніях проводять відео конференції. Проводитися вони можуть

як кожен день, якщо це можливо, або хоча б раз в неділю. Але й тут можуть виникнути проблеми, особливо якщо в команді багато співробітників і всі вони знаходяться в різних країнах. Кожен живе в своєму ритмі. Кожен намагається організувати робочій день у найзручніший для себе спосіб. І тому колеги не можуть зв'язатися один з одним, а іноді навіть обрати спільний для цього час. Тому загальний темп проєкту може значно знизитися, а робочій процес затягнутися [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис об'єкта дослідження

Концепція віддаленої роботи була створена американським дослідником Джеком Ніллесом у 1973 році. Ним був запропонований термін «телеробота», який дав поштовх до розвитку віртуальних фірм, як нестандартної форми організації бізнесу. З розвитком інформаційних технологій, його ідеї отримали подальший розвиток. Подібну практику застосовують майже всі ІТ-компанії, серед яких American Express, Adobe, Airbus Industries, Siemens, Microsoft, Google, IBM, Dell, eBay, Hewlett Packard, а також тисячі інших компаній [3].

Віддалена робота та використання територіально розподілених проєктних команд в сучасних умовах є життєвою необхідністю. Особливо це стосується сфери ІТ-аутсорсингу [3].

Частіше всього територіально розподілена команда – це група співробітників компанії, які виділені для певного проєкту і від сумісної роботи яких залежить результат проєкту. Між собою співробітники, як однієї команди, так і різних, здійснюють комунікації шляхом спілкування за допомогою інформаційних технологій. При цьому вони можуть знаходитися як в одному будинку, так і за тисячі кілометрів один від одного [3].

Майже все, що стосується роботи та проєкту знаходиться у віртуальному просторі, а фізичний офіс є лише допоміжним пунктом. Тому для успішної роботи головним є якість комунікацій та взаємодії учасників команд. Але створити сприятливі умови для ефективної сумісної роботи вдається далеко не в кожній компанії [3].

Існує ряд проблем, що стосуються організації робочого процесу розподіленої команди у віртуальному середовищі: забезпечення сумісної роботи над проєктом, мотивація та управління членами територіально розподіленої команди, забезпечення згуртованості для досягнення цілей проєкту, впровадження стандартів корпоративної культури [3].

Зарубіжними та вітчизняними дослідниками такими як Джеймс Гербслеб, Ребекка Грінтер, Джек Ніллес проводилися дослідження в цій галузі. Особливо потрібно відмітити львівську школу вітчизняних дослідників під керівництвом професора Андрія Пелешішина. Недослідженою лишилася тільки проблема комунікаційної взаємодії, що забезпечує успішне розв'язання задач, які виникають при розробці програмного забезпечення [3].

Розподілені команди повинні функціонувати при зміні робочого середовища, в якому відстань, часові межі, різні мови, відмінності у корпоративній культурі, різні нормативні та законодавчі вимоги можуть перешкоджати ефективному виконанню роботи [3].

Для вирішення робочих проблем керівник повинен встановити чіткі правила дистанційної роботи, прописати внутрішні політики та обрати програмні інструменти для цього [1].

Перед тим, як починати роботу над проектом, розробники повинні отримати задачі від менеджера. Для управління задачами використовується певне програмне забезпечення, де вони можуть бути у вигляді списку або дощок kanban. Менеджер може встановити час, за який бажано встигнути виконати задачу. Всі члени команди можуть переглядати задачі один одного, витрачений час та прогрес виконання задачі.

Для комунікації в команді використовують різні застосунки-месенджери, в яких можна створювати різні чати, в яких співробітники спілкуються в текстовому форматі. Також потрібні застосунки для відеоконференцій, в яких проводяться так звані мітинги (meetings), для яких обирається певний час, та на яких обговорюється, що було зроблено, що планується робити сьогодні та деякі інші питання пов'язані з проектом. Для одночасного доступу до програмного коду використовуються системи контролю версій, за допомогою яких можна зберігати, переглядати та керувати програмним кодом.

1.2 Опис предмета дослідження

Розглянемо деякі програмні засоби для організації роботи розподіленої команди.

1.2.1 Atlassian Jira

Спочатку система Jira створювалася як рішення для відстеження завдань та помилок. Але сьогодні Jira – це потужний інструмент управління роботою, що підходить для різних випадків, від управління вимогами і сценаріями тестування, до agile-розробки програмного забезпечення. Була розроблена у 2002 році компанією Atlassian. Зараз доступна в двох версіях: «хмарній» та серверній. Сьогодні Jira є однією з найпопулярніших систем керування проблемами [4], [5].

Сьогодні все більше команд використовують ітеративну розробку із циклічним повторенням етапів, і Jira Software стає єдиним центром для написання коду, спільної роботи та випуску релізів. Для управління тестуванням Jira інтегрується з різними доповненнями, які дозволяють органічно включити тестування у цикл розробки програмного забезпечення. Команди можуть проводити тестування ефективно та ітеративно. Команди контролю якості використовують завдання Jira, екрани, поля і робочі процеси, що налаштовуються для управління ручним та автоматичним тестуванням [4].

Для команд, що практикують методики agile, у Jira Software існують дошки Scrum та Kanban. Дошки виконують роль центру управління, в якому завдання розподіляються по робочим процесам, що налаштовуються. Дошки роблять командну роботу більш відкритою і дозволяють бачити стан кожної робочої задачі. Інструменти для відстеження часу та звіти про продуктивність у режимі онлайн дозволяють командам відслідковувати свою ефективність у перспективі [4].

Jira Software можна налаштувати для проєкту будь-якого типу. У команд є можливість розпочати з готового шаблону проєкту або створити власний процес. Jira дозволяє відслідковувати всі задачі. Права доступу дозволяють адміністраторам визначати, яким користувачам будуть доступні певні дії. За наявності повної інформації про проєкт можна згенерувати докладні звіти про ефективність та хід виконання роботи [4].

Jira Software надає інструменти для планування та складання дорожніх карт, щоб команди з першого дня могли управляти зацікавленими особами, бюджетами та вимогами до функціональності. Jira підтримує інтеграцію з різними інструментами CI/CD, що дає змогу підвищувати прозорість протягом усього циклу розробки програмного забезпечення. Коли продукт буде готовий до розгортання, інформація про стан коду з'явиться на сторінці задачі Jira. Вбудована функція включення та відключення можливостей дозволить командам впроваджувати нову функціональність поступово та безпечно [4].

Для команд, які наслідують принципи DevOps, рішення Jira Software є основою відкритого інтегрованого комплексу інструментів Atlassian, відомого як Open DevOps. Jira Software можна інтегрувати з власними та сторонніми інструментами, включаючи рішення для управління кодом та версіями, наприклад Bitbucket, GitHub та Gitlab, інструменти управління документацією та знаннями, такі як Confluence, а також інструменти моніторингу та управління, включаючи Opsgenie, протягом усього життєвого циклу DevOps. Крім того, Jira Software можна інтегрувати з інструментами різних категорій і цим допомогти командам DevOps швидше постачати якісне програмне забезпечення [4].

У Jira Software команди можуть скласти дорожню карту, пов'язану з усіма проєктами. Дорожня карта дозволяє командам планувати роботу у довгостроковій перспективі, відстежувати прогрес та ділитися інформацією про нього [4].

Jira дозволяє створювати задачі для себе та учасників команди, додавати до них подробиці, терміни виконання та нагадування, розділяти великі задачі на кілька дрібних підзадач, спостерігати за задачами іншим користувачам, щоб вони могли відстежувати його виконання та отримували повідомлення після вирішення задачі, створювати підзадачі всередині основної задачі, щоб розділити весь обсяг роботи для окремих членів команди, переглядати всі задачі на дошці, щоб легко визначити стан кожної з них [4].

Баги – це просто назва задач, які виникають, коли в програмному забезпеченні, що розробляється, виявляються проблеми. Командам важливо бачити всі задачі та баги в беклогу, щоб мати можливість визначити пріоритетні. Ефективна система управління робочими процесами в Jira забезпечує автоматичне призначення та визначення пріоритетів багів одразу після їх виявлення. Потім команди можуть відслідковувати баг до його виправлення [4].

1.2.2 Trello

Trello – це візуальний інструмент керування робочим процесом, який дає змогу командам формувати ідеї, планувати роботу та організовано й продуктивно керувати нею [6].

Trello пристосовується до будь-якого проєкту. Цей інструмент допомагає спростити й стандартизувати робочий процес команди в інтуїтивно зрозумілий спосіб. Trello дуже простий та зручний у використанні засіб, але він здатен впоратися з найскладнішими проєктами команди [6].

Інтерфейс Trello складається з кабан дошок, які використовуються для організації роботи з задачами [7].

Trello підтримує розширення, за допомогою яких можна збільшити функціонал. Але в безкоштовній версії можна встановити лише три розширення. Також в безкоштовній версії можна прикріплювати файли

розміром не більше 10 МБ. Але навіть у такому тарифі Trello дозволяє гнучко взаємодіяти невеликим командам та контролювати головні процеси та завдання. Це особливо важливо за умов віддаленої роботи [7].

1.2.3 Slack

Slack – корпоративний месенджер. Був запущений в тестовому режимі в серпні 2013 року. Публічний реліз відбувся 12 лютого 2014 року. За роки існування показав найбільший ріст серед бізнес-застосунків [8].

У Slack можна створювати різні канали для обговорення загальних робочих тем, приватні групи, а також писати особисті повідомлення. Застосунок має власний хостинг та підтримує інтеграцію з більше ніж 100 сторонніх сервісів: Dropbox, Google Drive, GitHub, Bitbucket, Google Docs, Trello та інші. У безкоштовній версії підтримує необмежену кількість користувачів, інтеграції з 5 сервісами й пошук до 10 тис повідомлень [8].

Великим недоліком безкоштовної версії є обмежена кількість повідомлень. Всі повідомлення які були написані пізніше 14 днів видаляються. Також є обмеження зберігання файлів [8].

1.2.4 GitHub

GitHub це система управління проєктами та версіями коду, а також платформа соціальних мереж, створена для розробників. Він дозволяє працювати разом з іншими людьми по всьому світу, планувати свої проєкти і відстежувати свою роботу [9].

GitHub також є одним із найбільших онлайн-сховищ для спільної роботи по всьому світу [9].

GitHub заснований на системі контролю версій Git, яка була розроблена Лінусом Торвальдсом, людиною, яка створила ОС Linux [9].

Система контролю версій допомагає відстежувати зміни, внесені до бази коду. Більше того, вона записує, хто вніс зміни та може відновити стертий або змінений код [9].

Окрім участі у певному проєкті, GitHub дозволяє користувачам спілкуватися з однодумцями. Можна стежити за людьми та дивитися, що вони роблять або з ким вони спілкуються [9].

Репозиторій або сховище – це каталог, де зберігаються файли проєкту. Він може бути розташований у сховищі GitHub або локальному сховищі на комп'ютері. Git дозволяє зберігати файли кодів, зображення, аудіо або все, що пов'язане з проєктом у сховищі [9].

Гілка – це копія репозиторію. Гілка використовується, коли потрібно зробити розробку ізольовано [9].

Робота з гілкою не вплине на центральне сховище чи інші гілки. За потребою можна поєднати свою гілку з іншими гілками та центральним репозиторієм, використовуючи запит на вилучення. Цей запит може бути переглянутий іншим більш кваліфікованим розробником, який вкаже на помилки і тільки після виправлення прийме запит. Це запобігає більшості з проблем в коді [9].

GitHub – це не тільки платформа для розробників. Кожен, хто хоче ефективно керувати своїм проєктом і працювати спільно, також може використовувати GitHub [9].

Окрім GitHub є й інші не менш популярні альтернативи, такі як BitBucket та GitLab [9].

1.2.5 Hurma

Hurma (скорочено від Human Resource Management) – продукт для автоматизації задач HR-відділу [10].

Hurma підходить для маленьких стартапів та компаній, у яких потрібно керувати великим штатом співробітників та наймати нових. Сфера

може бути будь-якою – ІТ-компанії, заводи, об'єкти інфраструктури, держпідприємства, будівельні концерни, ритейлери, фінанси, маркетинг [10].

Hurma System надає такі функції:

- створення та управління вакансіями компанії;
- створення та управління профілями кандидатів;
- створення та управління профілями співробітників;
- візуалізація організаційної структури компанії у зручному дереві команд;
- побудова ієрархії цілей компанії у системі OKR;
- нарахування та управління відпустками співробітників;
- робота із запитами (1:1, Performance review, лікарняні, робота з дому тощо);
- бронювання приміщень під зустрічі;
- робота з шаблонними повідомленнями для оповіщення учасників зустрічей та заходів;
- обробка статистики з рекрутингу, OKR та HR;
- управління заходами компанії (мітинги, інтерв'ю, зустрічі з клієнтами, корпоративні свята, навчання співробітників тощо);
- створення та управління опитуваннями;
- побудова програм внутрішнього навчання та менторства;
- моніторинг настрою працівників [10].

1.3 Постановка завдання роботи

В рамках дипломної роботи необхідно розробити мобільний застосунок для організації роботи розподіленої команди.

Найменування продукту – «TeamConnect».

Програмний продукт (ПП) повинен містити наступний функціонал: створення власного профілю, створення команди та додавання до неї

учасників, перегляд списку учасників команди, відмічування відпусток та лікарняних і перегляд списку відсутніх, перегляд списку спільних проміжків часу для командних зустрічей та подій.

Застосунок має добре працювати як на ОС Android, так і на ОС iOS. Через відсутність серверної частини, окрім деяких сервісів, які пов'язані з картою, всі дані повинні зберігатися у локальній базі даних (БД).

2 АНАЛІЗ ІНСТРУМЕНТІВ

2.1 Порівняльний аналіз мов програмування

На сьогоднішній день існує дуже багато мов програмування для розробки застосунків, але вибір конкретного інструмента залежить від цілей, можливостей, операційної системи, типу застосунка та вимог до нього [11].

Для написання мобільних застосунків сьогодні частіше всього використовують мови, які були створені під конкретну мобільну платформу. Такими мовами є: Kotlin або Java для Android і Swift або Objective C для iOS. Також популярними мовами для розробки мобільних застосунків є C#, Dart та JavaScript. Нижче в таблиці 2.1 наведено порівняльний аналіз мов програмування для розробки мобільних застосунків.

Таблиця 2.1 – Порівняння мов програмування Kotlin, Java, Swift , Objective C та C#

Мови програмування	ОС	Кросплатформовість	ОС для розробки	Простота вивчення
Kotlin	Android та iOS	+	Windows, Linux, MacOS	Просто вчити
Java	Android	-	Windows, Linux, MacOS	Відносно просто вчити
Swift	iOS	-	MacOS	Просто вчити
Objective C	iOS	-	MacOS	Складно вчити
C#	Android та iOS	+	Windows, Linux, MacOS	Просто вчити

Kotlin – мова програмування, рекомендована Google для розробки застосунків під ОС Android. Вона була створена у 2010 році міжнародною компанією JetBrains, яка була заснована програмістами з Петербурга для подолання недоліків Java, і використовується для написання коду майже всіх нових програм на Android [11].

Kotlin за кілька років завоював довіру програмістів та став галузевим стандартом в Android-розробці. У 2019 році компанія Google оголосила Kotlin мовою вартою переваги для розробки програм під її мобільну операційну систему, що означає пріоритетну підтримку з боку компанії у всіх інструментах, компонентах та API порівняно з іншими мовами [11].

Kotlin знайшов застосування і в серверній розробці, де почав тіснити позиції Java. В останні роки також розвивається Kotlin Multiplatform Mobile (KMM) – кросплатформова версія мови Kotlin, що дозволяє створювати спільну бізнес-логіку iOS та Android-застосунків [11].

Kotlin взаємозамінний з Java. Це дозволяє створювати застосунки на різних мовах та використовувати бібліотеки та фреймворки, які написані на Java у застосунках написаних на Kotlin. Хоча Kotlin обходиться написанням меншої кількості коду ніж на Java, але швидкість складання застосунків трохи нижча [11].

Swift – мова програмування від Apple, на якій можна створювати програми для iOS, Apple Watch та Apple TV. Swift не тільки перейняв усі можливості Objective C, але й був наділений новими функціями, які спрощують написання та реалізацію коду.

Переваги:

- висока швидкість – досягає рівня C++;
- простий для читання синтаксис та код;
- підвищена безпека, якщо порівнювати з Objective C;
- спрощений спосіб виправлення помилок у коді;
- стабільність за рахунок бібліотек, які автоматично зв'язуються з оновленою версією та приєднуються до програми;
- забезпечує безпечне керування пам'яттю [11].

Недоліки:

- Swift використовується для розробки під пристрої Apple. Інших застосувань мови практично немає, за дуже рідкісними винятками. Це обмежує його популярність;

- погана сумісність із попередніми версіями мови;
- для розробки обов'язково потрібна техніка Apple [11].

До 2018 року Java була основною мовою для створення програм під Android, але і в 2022 її продовжують використовувати розробники для підтримки старих проєктів. Важливо відзначити, що сама по собі мова не втратила популярності в ІТ, але набагато частіше зараз її використовують для Back-end розробки. Нові програми для Android цією мовою пишуть рідко. Але багато компаній при прийманні на роботу вимагають знання як Kotlin так і Java [11].

Переваги:

- природний код для Android;
- сама ОС частково теж написана на Java, а ядро становлять Linux та власна віртуальна машина Virtual Machine;
- дозволяє легко масштабувати та оновлювати проєкти за рахунок об'єктно-орієнтованого коду;
- багато готових інструментів, які за замовчуванням сумісні з Java, що також збільшує швидкість розробки;
- на мові Java можна розробляти не лише мобільні, а й серверні, настільні, системні та інші програми;
- мову Java знає більше програмістів [11].

Недоліки:

- програми на Java більше схильні до помилок, ніж на Kotlin;
- програми виходять досить багатослівними, що ускладнює їхнє читання;
- нові бібліотеки для Android-розробки насамперед орієнтовані на Kotlin, а не Java [11].

Objective C почали використовувати в 80-х роках 20 століття. Він був створений на основі C і Smalltalk, а в 2008 Apple випустила iPhone SDK 2.0, що дозволяє створювати програми для iOS. Спочатку його вважали

надбудовою до C, але коли його ліцензували NextStep і Apple, Objective C став офіційною мовою всіх інтерфейсів під iOS.

У 2014 році вийшов потужніший Swift, який взяв собі все краще від Objective C, але був позбавлений його недоліків. Зараз більшість програмістів обирають Swift, але Objective C все ще використовується для підтримки проєктів Legacy [11].

Переваги:

- існує багато документації, що спрощує роботу;
- сумісний зі Swift [11].

Недоліки:

- невисока продуктивність проти Swift;
- складний синтаксис [11].

Мова програмування C# – об'єктно-орієнтована мова програмування (ООП). Була розроблена в 1998 – 2001 роках інженерами Microsoft під керівництвом Андерса Хейлсберга та Скота Вільтамоута [12].

C# відноситься до сімейства мов з C-подібним синтаксисом [12].

Хоча є мови більш популярні, але C# теж використовують дуже багато програмістів. Ця мова по різних рейтингах входить в першу п'ятірку найпопулярніших мов програмування.

Багато програмістів обирають C# для розроблення програм за рахунок простоти синтаксису, зрівнюючи з такими ж C-подібними мовами як C++ або Java.

Також C# є мовою, яка працює на платформі .NET, яка є міжмовною платформою для розробки. Це означає що код написаний на C# або іншій мові платформи .NET транслюється в код на мові IL – об'єктно-орієнтована мова, яка схожа на Assembler. Цей код потім за допомогою JIT-компіляції буде переводиться у машинні команди під час роботи програми.

Для C# розробляється багато фреймворків та бібліотек, що спрощує розроблення. Ще одною перевагою є наявність спільного репозиторію для

бібліотек, який називається NuGet і через який можна встановити майже кожен бібліотеку, обравши будь-яку версію.

На мові C# можна написати будь-яке програмне забезпечення. Для комп'ютерів під ОС Windows можна розробляти як консольні, так і віконні програми. Для розробки вебзастосунків використовується платформа ASP.NET. Після виходу .NET Core з'явилася можливість створювати програми для ОС Linux. C# сьогодні є найпопулярнішою мовою для розробки відеоігор. Для цього є найпопулярніший движок Unity. Також можна розробляти застосунки з штучним інтелектом та чат-боти для різних месенджерів. Ну і звісно за допомогою фреймворка Xamarin можна розробляти застосунки одразу для Android, iOS, Windows та MacOS.

Звичайно у C# є й свої недоліки. Одним з них є те що програми написані на C# достатньо легко дизасимблюються. Також через JIT-компіляцію застосунки можуть працювати швидше ніж написані на інших мовах, але під час першого запуску можуть бути проблеми з продуктивністю.

Не дивлячись на те, що більшість компаній використовують Kotlin та Swift, багатьом компаніям зручніше використовувати кросплатформові мови для цього. Простіше вивчити одну мову програмування, ніж декілька різних під окрему платформу. І для цього наприклад можна використовувати Kotlin, але поки що KMM не отримала широкого поширення. Тому для цього використовують мови програмування, для яких були створені спеціальні фреймворки, які дозволяють створювати застосунки одразу для декількох платформ. Такими мовами є: C# (Xamarin), Dart (Flutter), JavaScript (React Native) та деякі інші менш популярні мови. Нижче у таблиці 2.2 наведено порівняльний аналіз найпопулярніших кросплатформових фреймворків для розробки мобільних застосунків.

Таблиця 2.2 – Порівняння фреймворків Xamarin, Flutter та React Native

Фреймворк	Популярність	Платформи	Можливі архітектурні шаблони
Xamarin	Не дуже популярний	Android, iOS, Windows, MacOS	MVVM, MVC
Flutter	Дуже популярний	Android, iOS	MVVM, MVC, MVP, Redux
React Native	Популярний	Android, iOS, Windows, MacOS, вебзастосунки	MVVM, MVC, MVP, MVI, Redux

React Native – кросплатформовий фреймворк для розробки мобільних застосунків на мові програмування JavaScript, а також TypeScript.

JavaScript – одна з найпопулярніших мов програмування у світі. Він використовується для створення інтерактивних вебсторінок, мобільних програм і навіть у серверній розробці [11].

React Native підтримує такі платформи як Android, iOS, MacOS, Web, Windows та UWP, дозволяючи розробникам використовувати можливості бібліотеки React поза браузером для створення програм, що мають повний доступ до системних API платформ [11].

Переваги:

- React Native легко освоїти за рахунок JavaScript;
- велика поширеність: до половини кросплатформовий застосунків розробляються на React Native;
- React Native підтримує інтеграцію з вже існуючими програмами – наприклад, частина інтерфейсу мобільного застосунку може бути реалізована на React Native, а частина – за допомогою суто платформових засобів [11].

Недоліки:

- продуктивність програм на React Native поступається програмам на Xamarin та Flutter;
- простота мови розробки негативно позначається на захищеності від помилок, порівняно з більш строгими мовами [11].

Dart є мовою програмування загального призначення від компанії Google, що розробляється з 2011 року. Спочатку він призначався для розроблення вебзастосунків [11].

Все змінилося в 2017 році, коли вийшла перша версія Flutter – комплекту засобів розробки та фреймворку з відкритим кодом для створення мобільних застосунків під Android та iOS, вебзастосунків, а також настільних застосунків під Windows, macOS та Linux. В якості мови програмування у Flutter використовується Dart [11].

Flutter та мова Dart активно розвиваються, переймаючи найкращі ідеї з Kotlin, Swift та інших мов програмування. Сьогодні Flutter є одним із найпопулярніших фреймворків у розробників кросплатформових застосунків [11].

Переваги:

- проста в освоєнні мова програмування, однаково зручна для програмістів, які раніше писали нативний код для Android, iOS або вебзастосунків;
- висока продуктивність щодо більшості інших кросплатформових фреймворків завдяки компіляції в нативний код для цільової платформи;
- можливість писати єдиний код для будь-якої платформи;
- незважаючи на свою молодість, Flutter та Dart вже завоювали велику популярність серед програмістів, тому для даного фреймворку адаптовано безліч бібліотек та інструментів, можна легко знайти документацію та приклади [11].

Недоліки:

- число фахівців, які знають Flutter, зростає, але поки що поступається кількістю нативних програмістів;
- мова Dart поки що поступається в гнучкості мови Kotlin, хоча і тут відставання скорочується [11].

Xamarin – найстаріший з фреймворків. Був випущений у 2011 році. Отримав свою популярність через свою високу продуктивність. З 2016 року Xamarin належить Microsoft [13].

Xamarin підтримує два типи розробки: Xamarin Native та Xamarin Forms.

Xamarin.Android та Xamarin.iOS часто називають Xamarin.Native або Xamarin Traditional. Це кросплатформовий набір інструментів, який дозволяє розробникам створювати мобільні застосунки з використанням C# та .NET Framework. Незалежний від платформи код програм Xamarin Native зберігається або в бібліотеці класів (PCL), що переноситься, або в загальному проєкті. Такі інструменти, як Xamarin.Mac, Xamarin.iOS і Xamarin.Android, допомагають реалізувати функції, що залежать від платформи, і власний інтерфейс користувача [14].

Xamarin.Forms – це розширена версія Xamarin Native, що дозволяє писати один XAML код для одночасної роботи з кількома платформами. Програми, розроблені з використанням Xamarin.Forms, успадковують усі функції платформи Xamarin. За допомогою Xamarin.Forms розробники можуть писати бізнес-логіку на C#, а інтерфейс користувача або C# кодом, або XAML кодом, щоб його можна було використовувати на всіх платформах [14].

З огляду на порівняльну характеристику мов програмування та фреймворків, а також всі переваги та недоліки мови програмування C# та фреймворка Xamarin, прийнято рішення про використання останніх в якості інструментів для розробки.

2.2 Порівняння середовищ розробки

Для розробки програм на будь-якій мові програмування необхідно обрати середовище розробки. Це може бути як велика програма, яка включає в себе і редактор коду, і компілятор, і багато інших інструментів для

налагодження та тестування програм, так і звичайний текстовий редактор, але тоді додатково потрібно встановлювати необхідні сторонні інструменти. Найпопулярнішими середовищами для розробки програм на C# є: Visual Studio, Rider та Visual Studio Code. Нижче у таблиці 2.3 наведено порівняльний аналіз середовищ розробки для створення програм на мові C#.

Таблиці 2.3 – Порівняння середовищ розробки Visual Studio, Rider та Visual Studio Code

Середовище розробки	ОС	IDE або редактор коду	Мови програмування	Наявність безкоштовної версії
Visual Studio	Windows та MacOS	IDE	C#, C++, F#, TypeScript, JavaScript, Iron Python та Visual Basic	+
Rider	Windows, Linux та MacOS	IDE	C#, F# та Visual Basic	-
VS Code	Windows, Linux та MacOS	Редактор коду	Будь-яка мова програмування	+

Visual Studio – інтегроване середовище розробки (IDE), створена компанією Microsoft для розробки різних типів програм на мовах: C#, C++, F#, TypeScript, JavaScript, Iron Python та Visual Basic. Крім типових функцій будь-якої IDE, а також деяких спеціальних редакторів коду, Visual Studio має багато корисних інструментів для розроблення, які прискорюють процес створення програм. Такими інструментами є IntelliCode – інструмент для автоматичного завершення коду, інструмент для керування версіями Git, а також різні зручні інструменти для тестування та аналітики коду та спільної роботи. А якщо цих інструментів буде недостатньо, то у Visual Studio є можливість додати нові, або покращити існуючі, для цього є розширення, які можна знайти та встановити в самій IDE.

Існує три версії Visual Studio: Community, Professional та Enterprise. З них тільки Community версія є безкоштовною, за інші необхідно платити.

Visual Studio є безумовно чудовою IDE, але так як вона була спочатку розроблена для Windows, тому краще всього вона працює саме на цій ОС. На версії для MacOS деякі можливості відсутні і працює ця версія значно гірше ніж версія для Windows. Версії для Linux у Visual Studio зовсім немає.

Rider – кросплатформова IDE для .NET розробників, заснована на платформі IntelliJ та ReSharper [15].

Основною перевагою Rider є ReSharper – інструмент, створений для збільшення продуктивності роботи та автоматизації рефакторингу. Він здійснює миттєвий статичний аналіз коду, спрощує модульне тестування, має додаткові інструменти автозаповнення, пошуку, навігації, виділення синтаксису, генерації коду, форматування та оптимізації. Хоча у Rider є дуже зручний ReSharper, але ця IDE є платною програмою і має лише пробну безкоштовну версію на 30 днів. Також ReSharper можна встановити як розширення для Visual Studio, але воно також є платним [16].

Visual Studio Code – безкоштовний редактор коду від компанії Microsoft.

VS Code не звичайний текстовий редактор. У нього є можливості, які притаманні майже всім сучасним IDE: виділення синтаксису, налагодження коду, автоматичне завершення коду та деякі інші. І хоча цього достатньо для майже будь-якої розробки, але її не можна назвати повноцінною IDE, так як у VS Code можна писати на будь-якій мові програмування, вона не має конкретної направленості, а також не має за замовчуванням інструментів, які необхідні для розробки на деяких мовах. Але за рахунок розширень, як і в вище описаних IDE, можна збільшити кількість можливостей. Можна додати інструменти, які необхідні для розробки на потрібній мові, можна розширити можливості VS Code, можна покращити вже існуючі інструменти, а також змінити зовнішній вигляд середовища розробки.

VS Code частіше всього використовують для веброзробки і редактор для цього має найбільшу кількість інструментів, як вбудованих, так і з розширеннями. Але у цього середовища є безперечні переваги. По-перше VS Code безкоштовний, а по-друге його можна встановити на будь-яку з популярних ОС: Windows, Linux та MacOS.

З огляду на характеристику кожного з середовищ, а також переваги та недоліки кожної, було обрано Visual Studio в якості середовища розробки на мові C#.

2.3 Архітектура програми

Для реалізації застосунку було обрано шаблон проєктування MVVM. Це один з доступних архітектурних шаблонів проєктування для розробки програм з використанням фреймворка Xamarin.

MVVM – спосіб організації коду, який допомагає відділити інтерфейс користувача від логіки. Програмний код умовно ділиться на три частини [17]:

- View містить інтерфейс користувача. У Xamarin.Forms представлений у вигляді коду на мові розмітки XAML;
- Model – класи з даними, а також логіка пов'язана з ними. В той же час моделі не можуть мати логіку для відображення цих даних, а також мати доступ до елементів управління;
- ViewModel використовується для зв'язування Model та View. Має логіку для отримання даних із Model та передавання їх у View, а також оновлення даних у Model при взаємодії з View [17].

ViewModel у Xamarin.Forms реалізується за допомогою інтерфейса INotifyPropertyChanged, який додає подію OnPropertyChanged, за допомогою якої оновлюються View та Model.

Для того щоб відділити інтерфейс від логіки була застосована бібліотека Prism.Unity.Forms. Це фреймворк для створення слабо пов'язаних застосунків, які добре підтримуються та добре тестуються. Prism підтримує

реалізацію MVVM, впровадження залежностей (DI), команди та декілька інших корисних інструментів. Також фреймворк додає зручну навігацію.

Model у застосунку ділиться на дві частини:

- перша частина включає в себе усі класи з даними;
- друга частина – це класи-сервіси, які потрібні для прямої роботи з даними: зберігання в базу даних, зберігання у файли, вилучення даних, оновлення, видалення тощо.

Головними класами-сервісами є Repository, SettingsManager та RestService:

- Repository – сервіс для роботи за базою даних SQLite. Для цього використовується бібліотека sqlite-net-pcl;
- SettingsManager – сервіс для збереження простих даних. Ці дані зберігаються в звичайний xml-файл на конкретному пристрої;
- RestService – сервіс для роботи з обміном даними з сервером.

Ці сервіси не працюють з конкретними даними. Для цього існують усі інші сервіси, які для роботи використовують методи головних класів-сервісів.

2.4 Висновки з розділу 2

У даному розділі проведено аналіз мов програмування, а також кросплатформових фреймворків для розробки мобільних застосунків. У результаті аналізу було обрано мову програмування C# та фреймворк Xamarin.Forms, через свою кросплатформовість та зручність розробки.

Також було проведено порівняння середовищ розробки. І в ході порівняння, дивлячись на всі переваги та недоліки, було обрано Visual Studio.

Для реалізації застосунку було обрано архітектурний шаблон MVVM. Сама архітектура була описана у розділі 2.3.

3 РОЗРОБКА ПРОГРАМИ

Застосунок умовно можна поділити на три частини:

- Model – класи для роботи з даними;
- View – інтерфейс;
- ViewModel – класи для взаємодії між Model та View.

У застосунку була використана карта, для користування якою на ОС Android потрібен API ключ, який прописаний у файлі AndroidManifest.xml. Крім карти був застосований ще TimezoneAPI від Google для визначення часового поясу певної країни. Для взаємодії з цим API був створений RestService.

Також до проєкту було підключено декілька бібліотек та фреймворків:

- Prism.Unity.Forms – MVVM, DI та зручна навігація;
- sqlite-net-pcl – робота з базою даних SQLite;
- Newtonsoft.Json – робота з json;
- Xamarin.Forms.Maps – робота з картою Apple для iOS та картою Google для Android;
- Nager.Date – визначення святкових днів та вихідних для певної країни;
- Xamarin.CommunityToolkit – різні інструменти для спрощення розробки на Xamarin.Forms;
- Xamarin.FFImageLoading та Xamarin.FFImageLoading.Forms – елемент керування для взаємодії з зображеннями;
- Acr.UserDialogs – стандартні для Android, iOS та Windows спливаючі вікна;
- Rg.Plugin.Popup – створення власних спливаючих сторінок;
- Prism.Plugin.Porups – взаємодія Rg.Plugin.Popup з навігацією Prism.Unity.Forms.

На рисунку 3.1 зображено структуру проєкту.

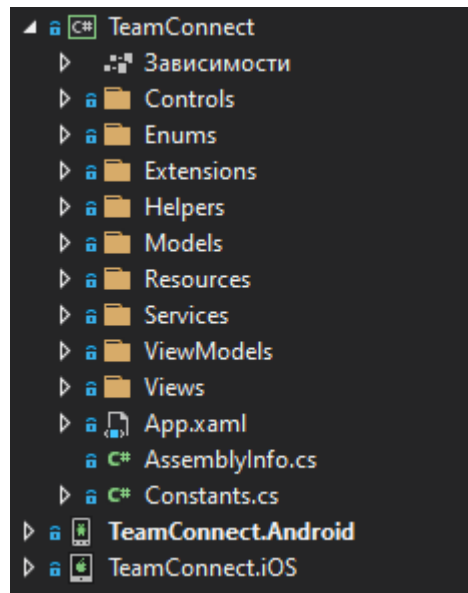


Рисунок 3.1 – Структура проєкту

3.1 Класи для роботи з даними

Класи для роботи з даними (Model) умовно діляться на класи-моделі (Models) та класи-сервіси (Services).

3.1.1 Models

У проєкті є декілька різних видів моделей:

- моделі (Models) – класи, дані в яких зберігаються у БД. Вони реалізують базовий інтерфейс `IEntityBase`, в якому є лише одна властивість `Id`, але за потреби можна додати й інші необхідні властивості. Моделі містять в собі лише властивості;
- моделі DTO (Data Transfer Object) – моделі для даних, які приходять з сервера;
- вьюмоделі (ViewModels) – моделі, які потрібні для відображення у View. Ці моделі можуть вже містити в собі за потреби методи, приватні поля та інші члени класу;

– груповані вьюмоделі – вьюмоделі для відображення даних у групованих списках.

В таблиці 3.1 наведені всі моделі, що містяться в проєкті.

Таблиця 3.1 – Моделі

Модель	Опис
Leave	Містить в собі такі класи: LeaveModel, LeaveViewModel та LeaveGroupViewModel. Зберігає дані про відпустки.
Team	Містить в собі такі класи: TeamModel та TeamViewModel. Зберігає дані команд.
TimeZone	Містить в собі такі класи: TimeZoneModel та TimeZoneModelDTO. Зберігає дані про часовий пояс в обраній країні. Не містить вьюмоделі, так як дані, що отримуються з серверу, не відображаються на екрані, а застосовуються лише у сервісах.
User	Містить в собі такі класи: UserModel, UserViewModel та UserGroupViewModel. Зберігає дані користувачів.

3.1.2 Services

В таблиці 3.2 наведені всі сервіси, що містяться в проєкті.

Таблиця 3.2 – Сервіси

Сервіс	Опис
1	2
AuthorizationService	Сервіс, який містить методи для створення аккаунту, входу та виходу з нього.
LeaveService	Сервіс для роботи з відпустками.
MapService	Сервіс для роботи з картою та часовими поясами.

Продовження таблиці 3.2

1	2
Repository	Сервіс для прямої роботи з методами БД SQLite.
RestService	Сервіс для роботи з запитам до сервера.
SettingsManager	Сервіс для зберігання даних простих типів: строкових, числових та деяких інших.
TeamService	Сервіс для роботи з даними команд.
UserService	Сервіс для роботи з даними користувачів.

Класи Repository, RestService та SettingsManager – сервіси низького рівня. Вони працюють на пряму з методами БД, запитам до серверу та методами зберігання простих даних у файл xml. Усі інші класи – сервіси високого рівня, що застосовують методи та властивості сервісів низького рівня.

Методи Repository:

- InsertAsync – метод для збереження даних нового об’єкта;
- DeleteAsync – метод для видалення даних певного об’єкта;
- UpdateAsync – метод для відновлення даних вже існуючого об’єкта;
- GetAllAsync – метод для отримання списку всіх об’єктів певного типу та їх даних;
- FindAsync – метод для отримання даних певного об’єкта.

Методи RestService:

- GetAsync – метод, який робить запит на отримання даних з сервера.

Властивості та методи SettingsManager:

- UserId – властивість для зберігання та отримання Id користувача;
- TeamId – властивість для зберігання та отримання Id команди;

– ClearSettings – метод для встановлення стандартних значень UserId та TeamId.

3.2 Інтерфейс

Робота в застосунку починається зі сторінки «Login» (Вхід). На рисунку 3.1 можна побачити як виглядає ця сторінка на ОС Android та ОС iOS. На цій сторінці знаходяться два текстових поля: «Email» та «Password» (Пароль), а також дві кнопки: «Login» та «Sign up» (Зареєструватися). Кнопка «Login» буде недоступна для натискання поки в текстові поля не будуть введені дані.

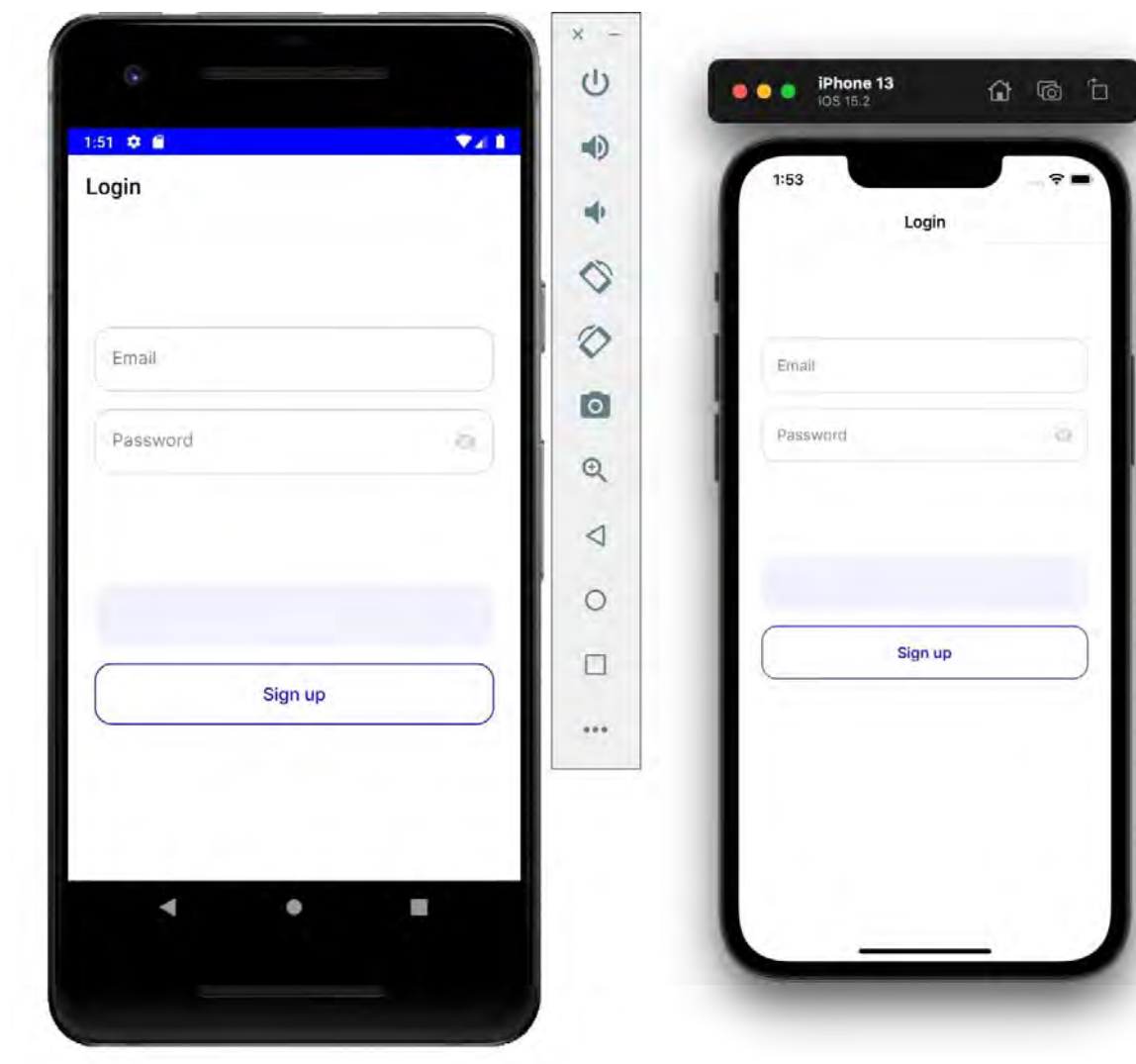


Рисунок 3.1 – Сторінка Login

Якщо ввести неправильні дані та натиснути кнопку «Login», під текстовими полями з'являться повідомлення про помилку «Invalid email» (Недійсний email) та «Invalid password» (Недійсний пароль). На рисунку 3.2 наведено сторінку «Login» з повідомленнями про помилку.

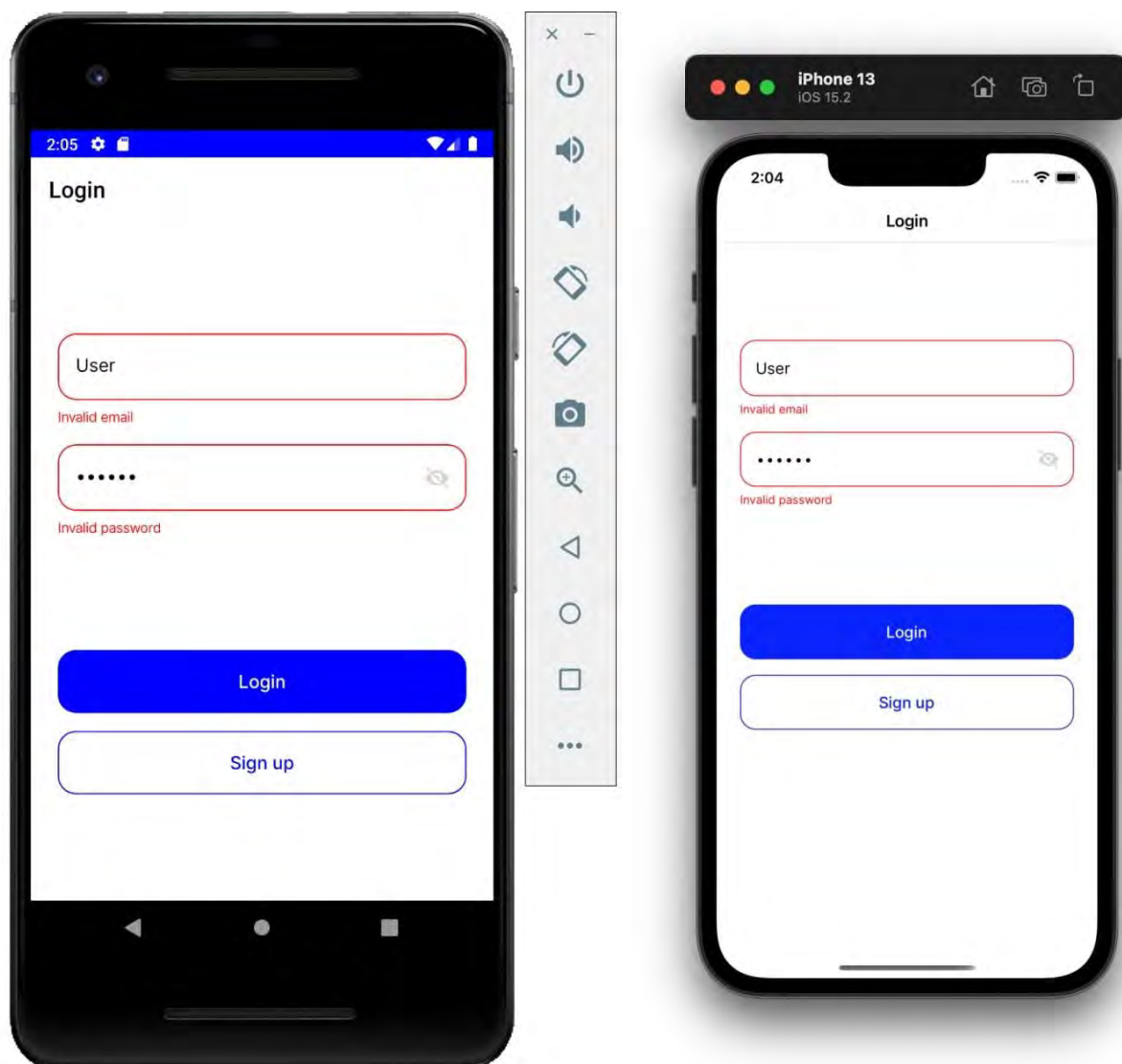


Рисунок 3.2 – Login сторінка з повідомленнями про помилку

Щоб зареєструватися потрібно натиснути кнопку «Sign up», після чого відкриється перша сторінка «Create account» (Створити акаунт). На рисунку 3.3 показано як виглядає ця сторінка. На цій сторінці є два текстових поля: «Name» (Ім'я) та «Surname» (Прізвище), а також кнопка «Next» (Далі). Кнопка «Next» буде недоступна доки не будуть введені дані в текстові поля.

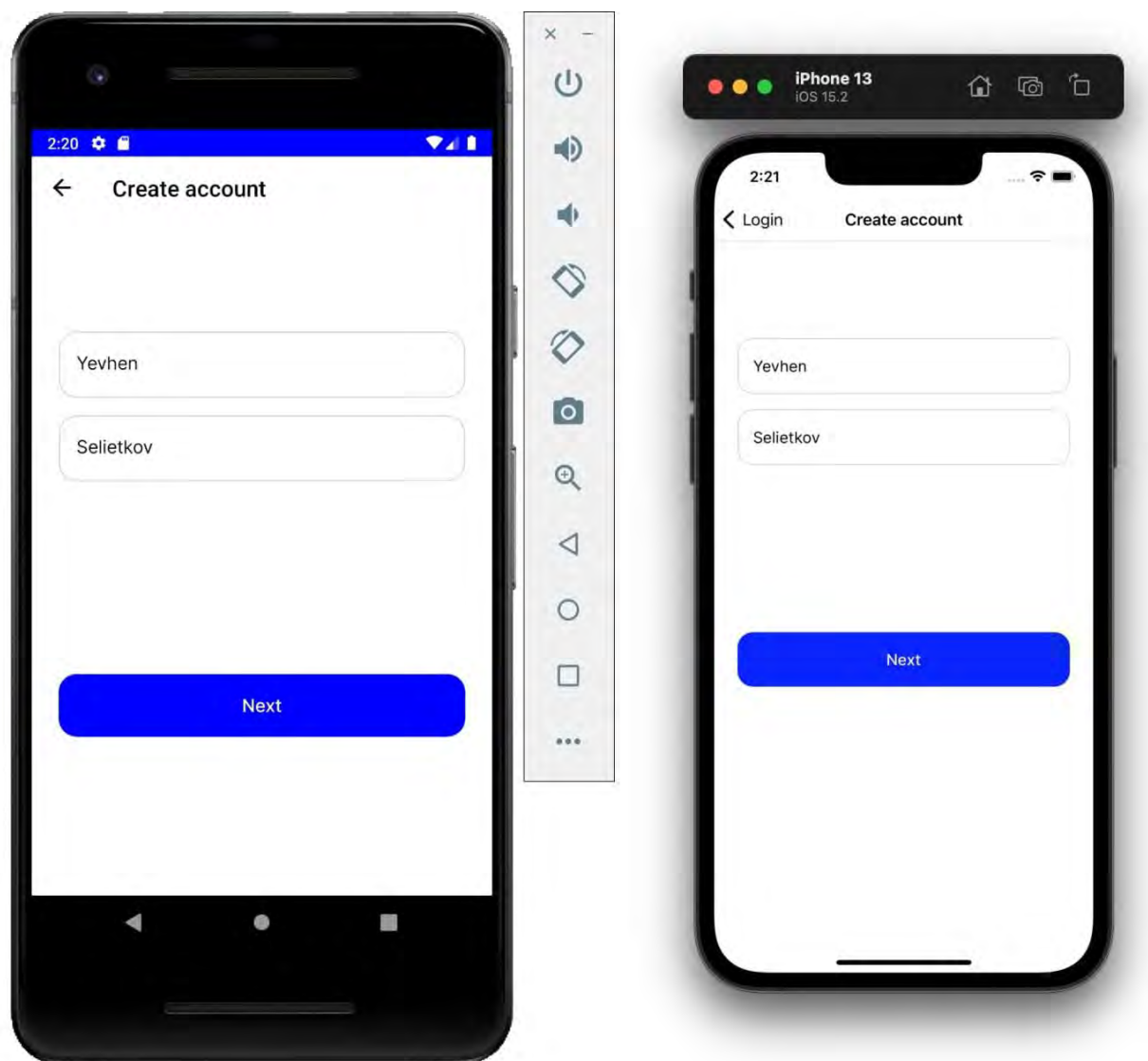


Рисунок 3.3 – Перша сторінка Create account

Якщо натиснути кнопку «Next», відкриється друга сторінка «Create account». На рисунку 3.4 показано як виглядає ця сторінка. На цій сторінці є три текстових поля: «Email», «Password» та «Confirm password» (Підтвердити пароль), а також кнопка «Create account», яка буде недоступна поки не ввести дані у текстові поля.

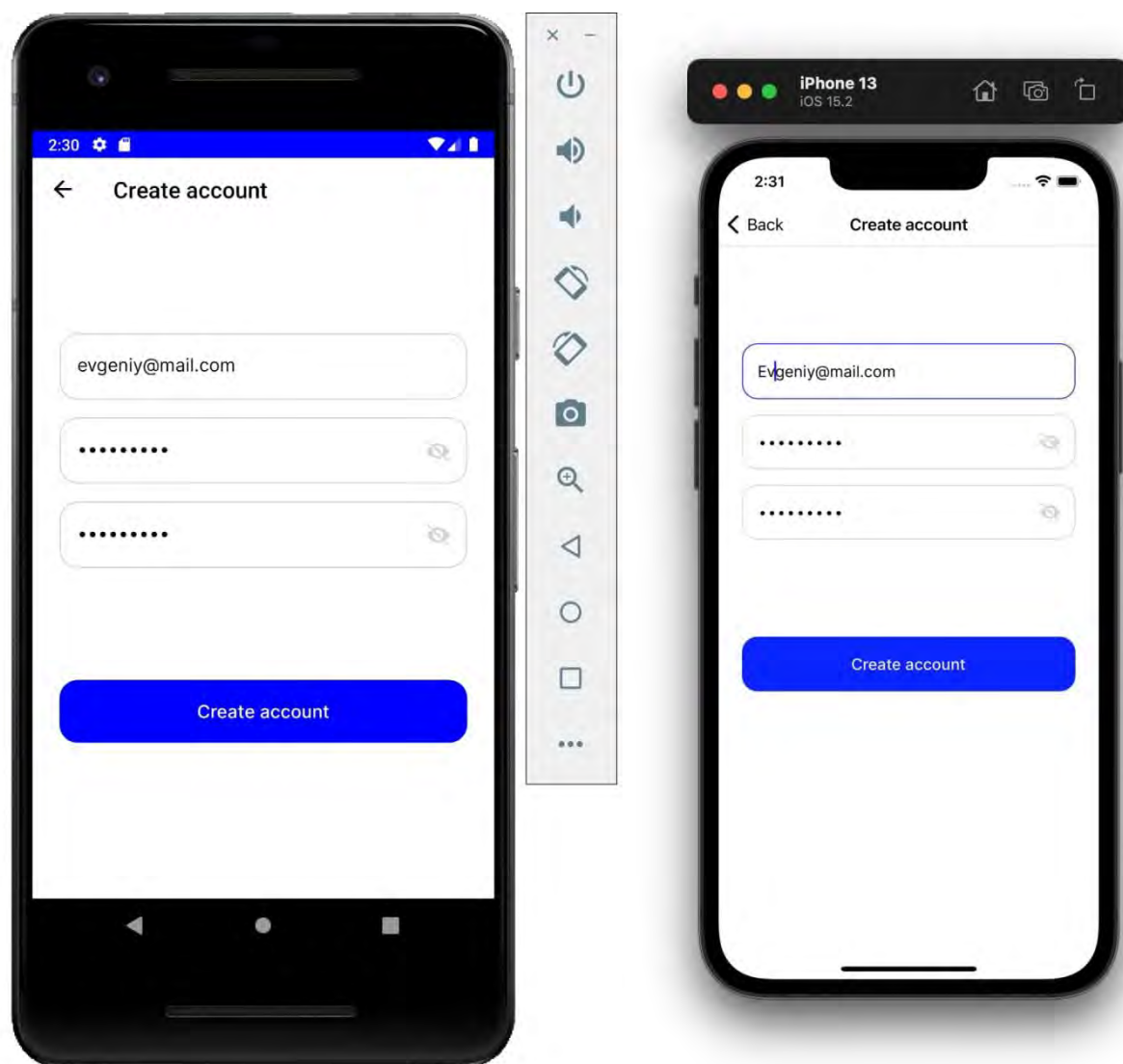


Рисунок 3.4 – Друга сторінка Create account

Якщо ввести email та пароль у невірному форматі, під текстовими полями з'являться повідомлення: «Wrong email» (Неправильний email) або «Wrong password» (Неправильний пароль). Якщо ввести email, на який вже зареєстровано іншого користувача, під текстовим полем Email з'явиться повідомлення «Please type your email» (Будь ласка введіть ваш email). Якщо ввести в поле «Confirm password» пароль, який не співпадає з тим, який був введений у «Password», з'явиться повідомлення «Please confirm password» (Будь ласка підтвердіть пароль). Якщо ж всі дані введені вірно, після натиснення кнопки «Create account» створиться новий аккаунт та знову

відкриється сторінка «Login», де в полі «Email» буде автоматично введено email нового користувача.

Після натиснення кнопки «Login» якщо email та пароль були введені вірно, а також якщо користувач тільки пройшов реєстрацію відкриється перша сторінка «Complete registration» (Завершити реєстрацію). Якщо ж користувач вже був зареєстрований раніше та вже користувався застосунком відкриється сторінка «Team» (Команда). На рисунку 3.5 показано як виглядає сторінка «Complete registration». На сторінці зверху знаходиться текстове поле «Address» (Адреса), нижче знаходиться карта, яка займає значну частину сторінки, внизу сторінки, поверх карти знаходиться кнопка «Next».

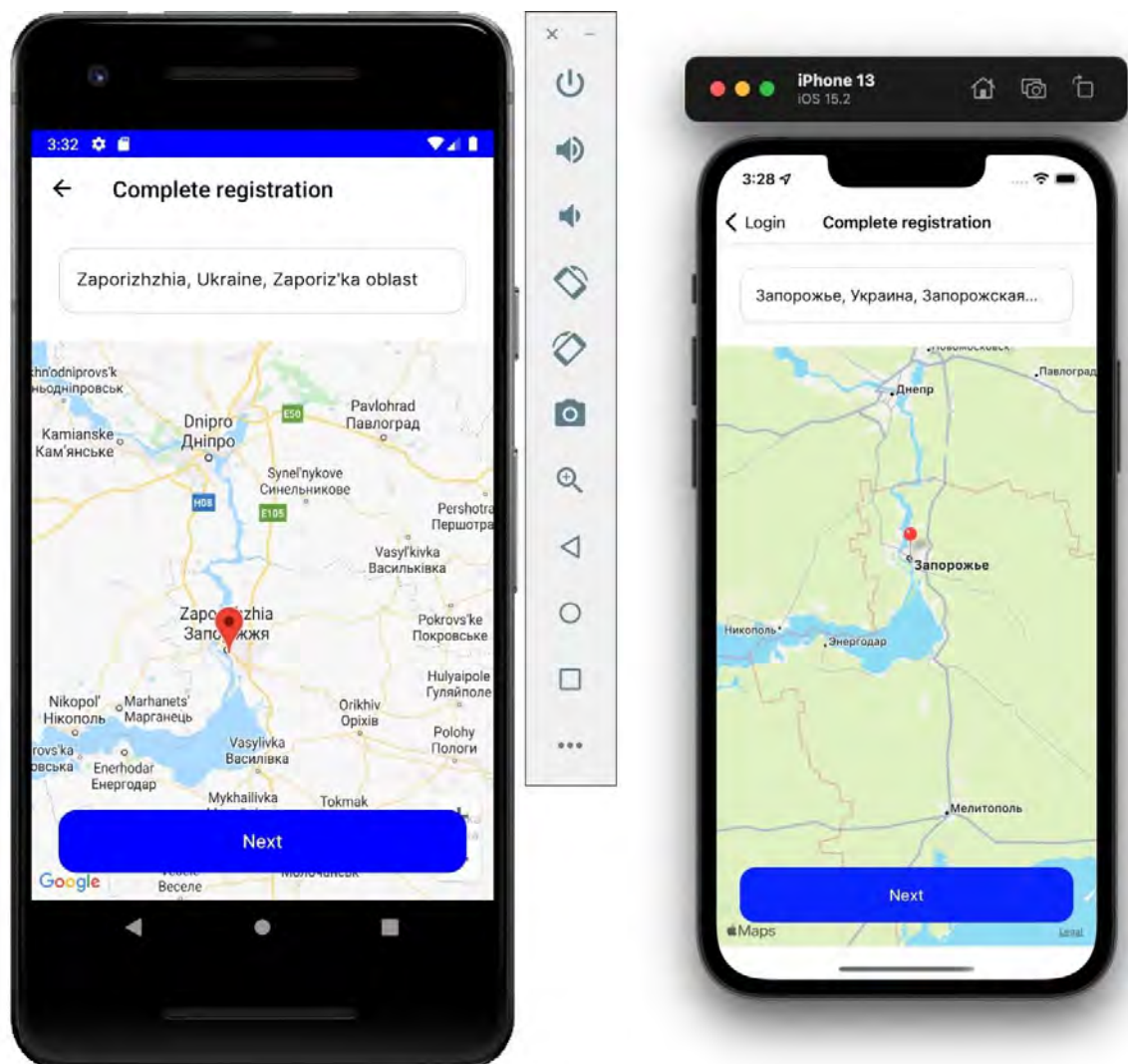


Рисунок 3.5 – Перша сторінка Complete registration

Натиснувши на карту в певному місці, з'явиться маркер, а в текстовому полі буде вказано адресу обраної місцевості. Якщо була обрана недоступна місцевість, з'явиться спливаюче вікно з текстом «Wrong location» (Неправильна локація). Це вікно наведено на рисунку 3.6.

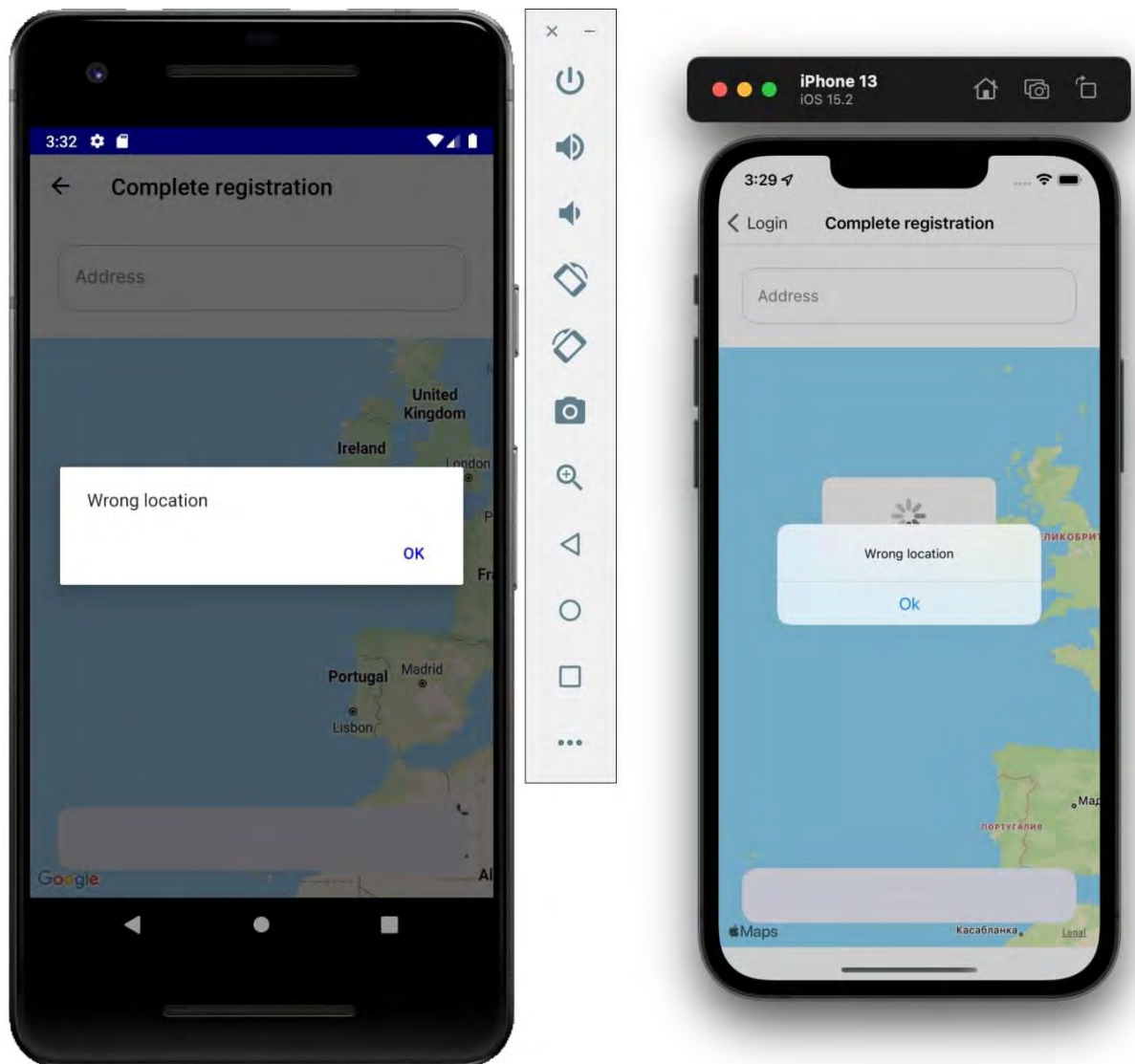


Рисунок 3.6 – Спливаюче вікно з текстом «Wrong location»

Якщо обрана локація була вірна, після натиснення на кнопку «Next» відкриється друга сторінка «Complete registration». Ця сторінка зображена на рисунку 3.7. Сторінка містить три поля: поле «Select position» (Обрати посаду) та два поля «From» (З) та «To» (До) для вибору проміжку робочого часу.

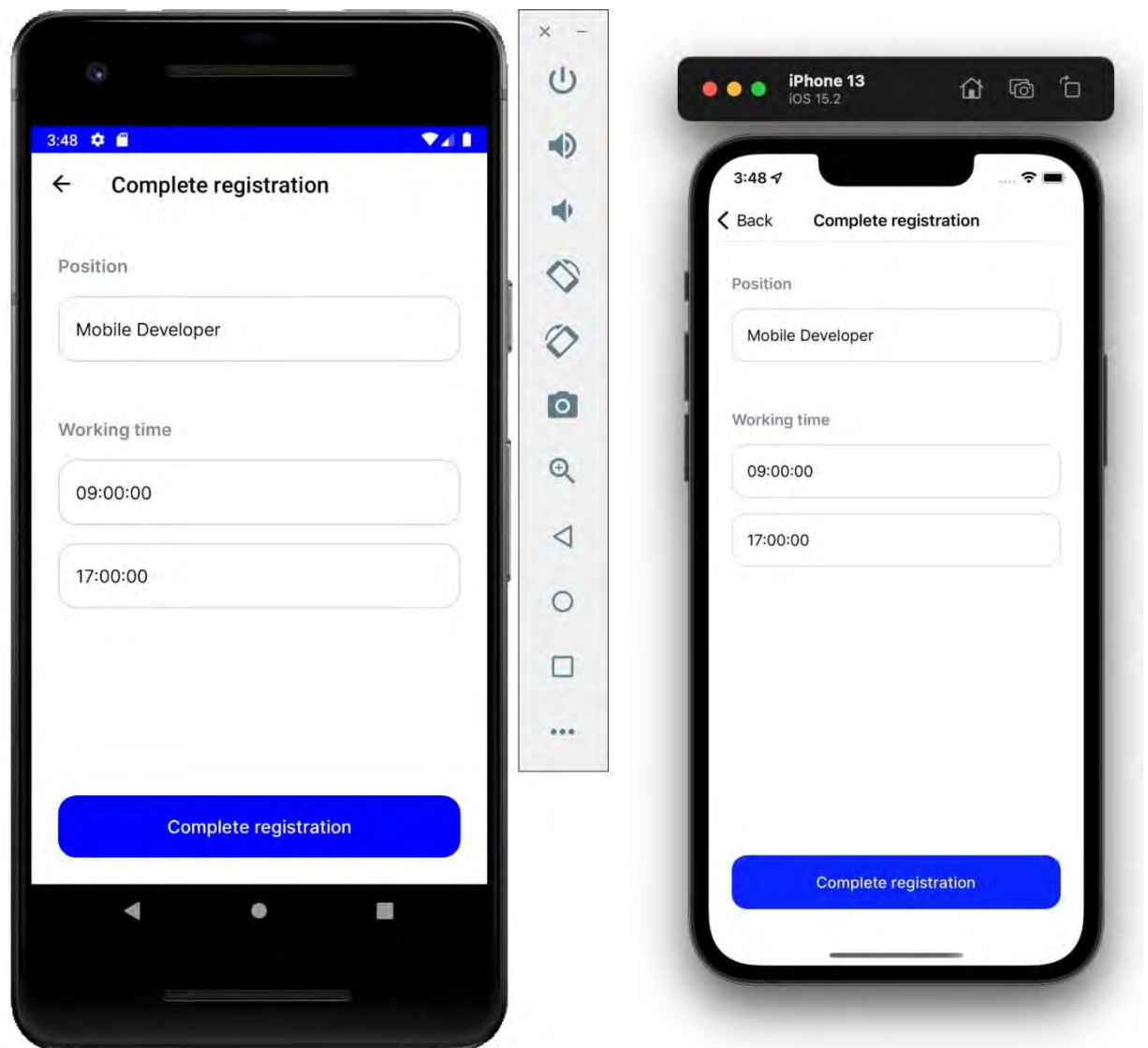


Рисунок 3.7 – Друга сторінка «Complete registration»

Щоб обрати посаду, потрібно натиснути на поле «Select position», після чого відкриється спливаюче вікно зі списком посад. Це спливаюче вікно наведено на рисунку 3.8.

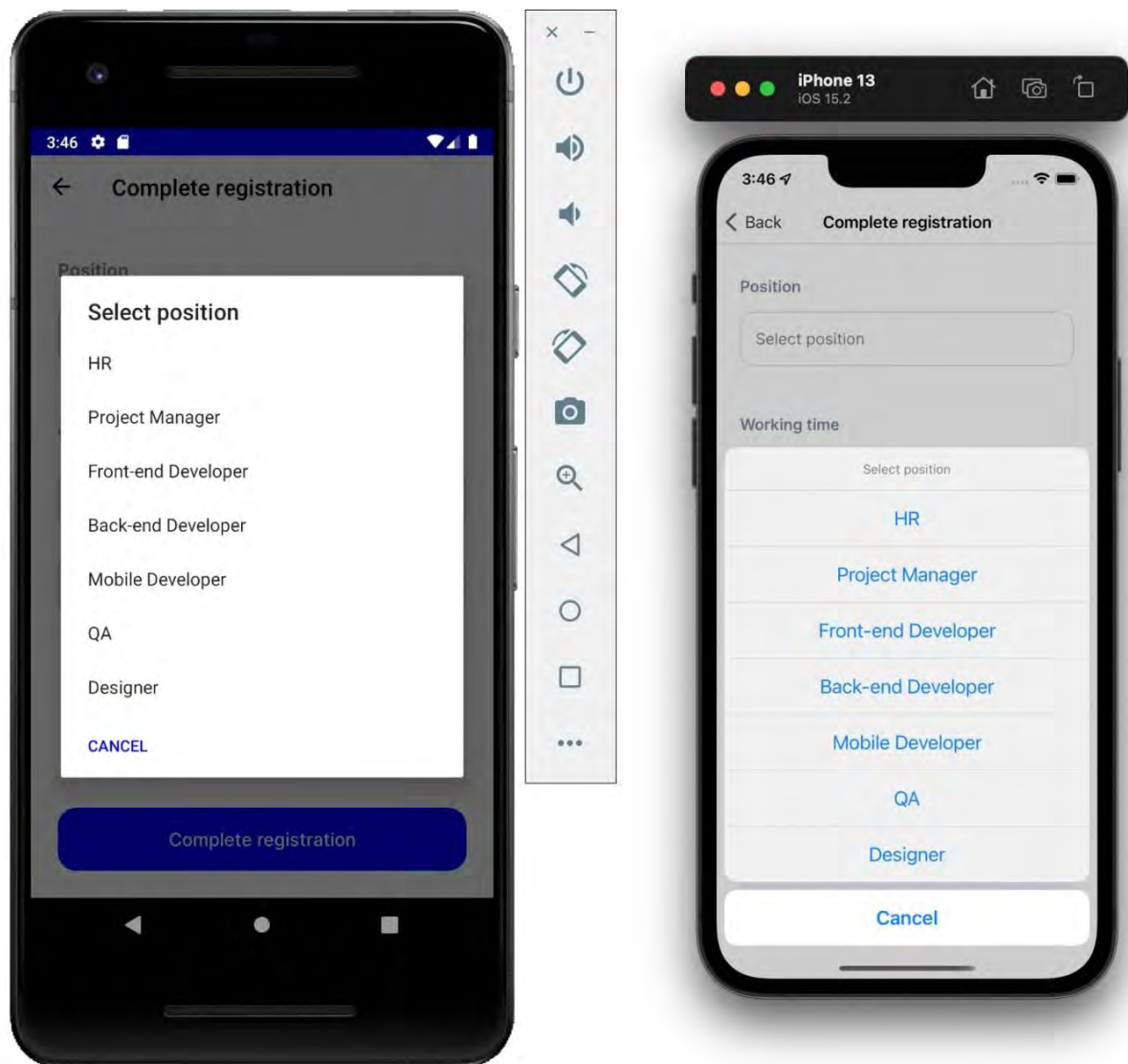


Рисунок 3.8 – Спливаюче вікно зі списком посад

Щоб обрати проміжок робочого часу, потрібно натиснути на поле «From» або «To», після чого відкриється спливаюче вікно з вибором часу. Це вікно наведено на рисунку 3.9.

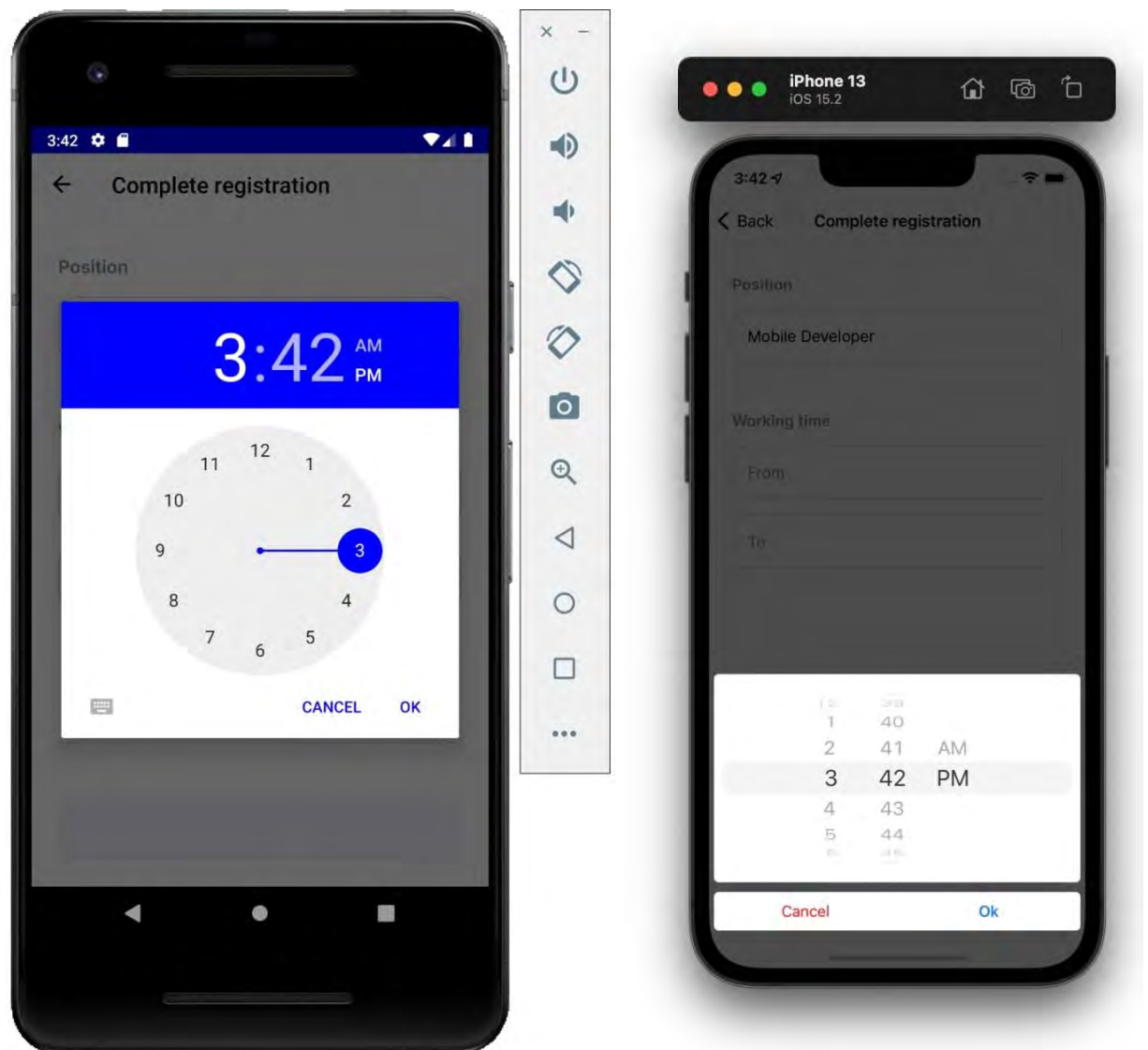


Рисунок 3.9 – Спливаюче вікно з вибором часу

Якщо були обрані посада та проміжок робочого часу, кнопка «Complete registration» стане доступною. Після натиснення на неї, завершиться створення аккаунта та відкриється сторінка «Team». На рисунку 3.10 показано як виглядає ця сторінка. Якщо користувач не є частиною команди, то на сторінці в середині будуть розташовані текстове повідомлення «No team» (Немає команди) та кнопка «Create a new team» (Створити нову команду).

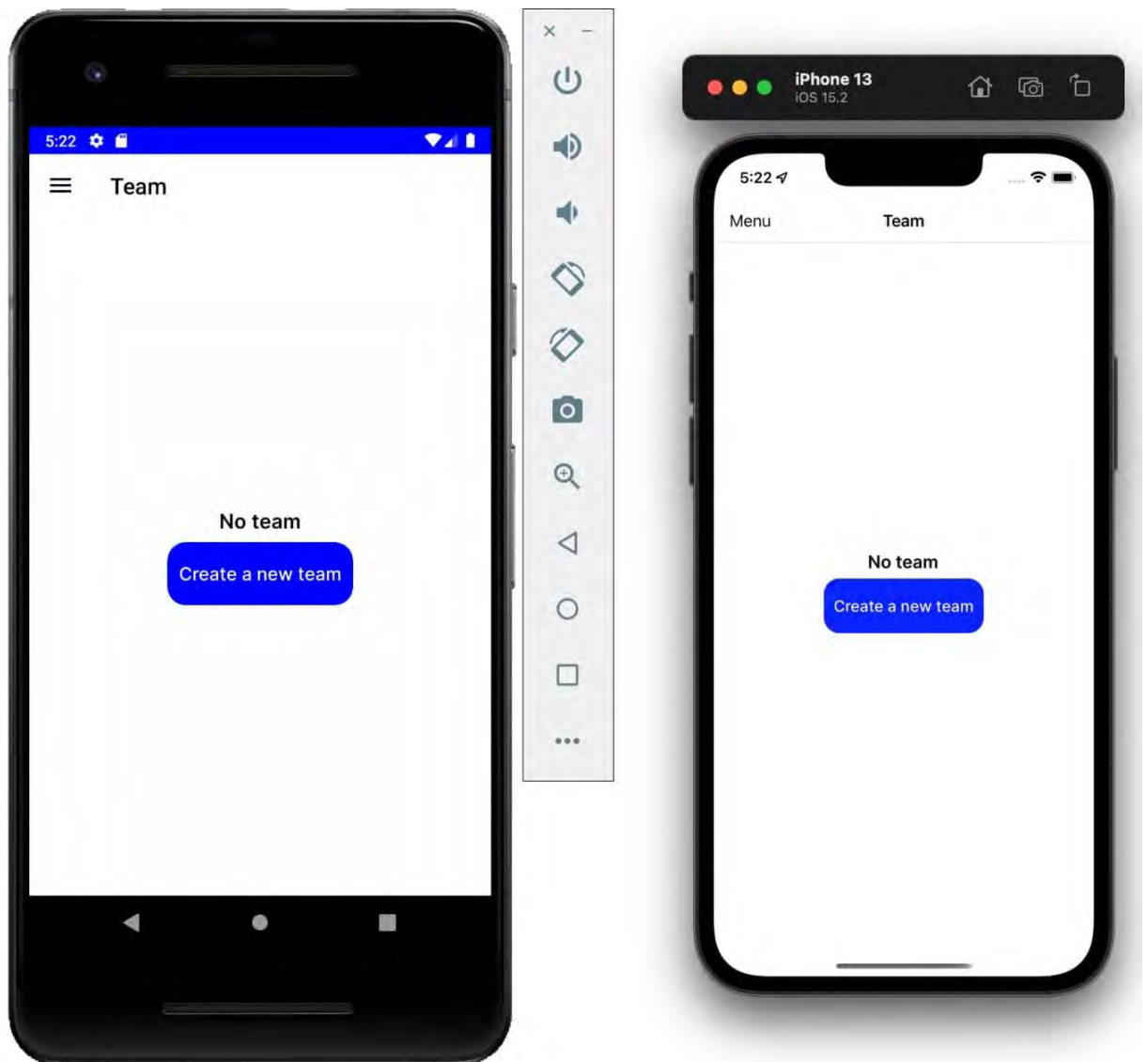


Рисунок 3.10 – Сторінка «Team»

Якщо натиснути кнопку «Create a new team», відкриється спливаюча сторінка. Ця сторінка зображена на рисунку 3.11. Сторінка містить текстове поле «Team name» (Назва команди) та дві кнопки «Cancel» (Відмінити) та «Create» (Створити).

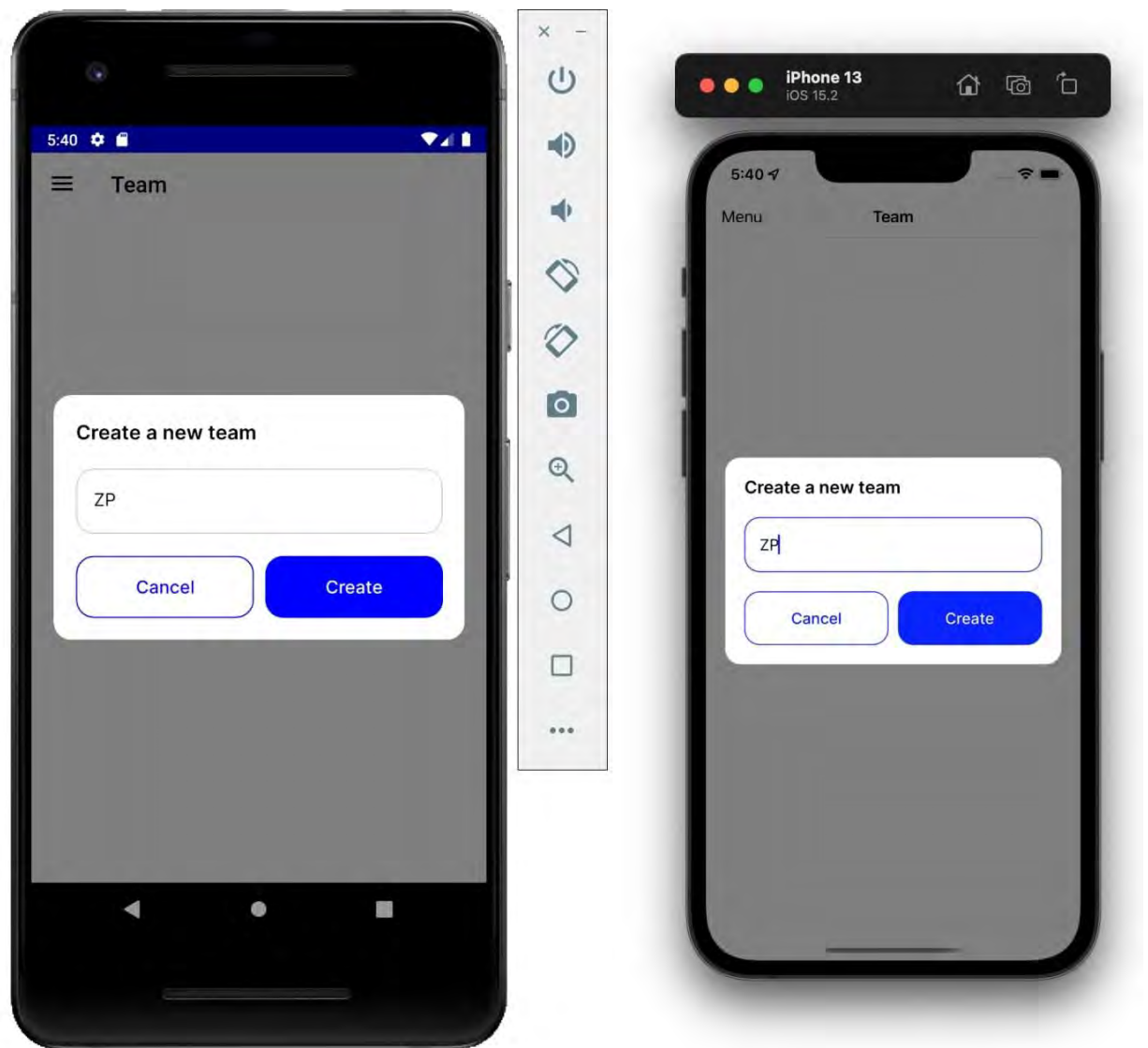


Рисунок 3.11 – Спливаюча сторінка «Create a new team»

Після натиснення на кнопку «Create», створиться команда з назвою, яка була введена у текстове поле. Тепер на сторінці «Team» буде знаходитися список членів команди, а також в правому нижньому куті з'явиться кругла кнопка з іконкою «плюса». Сторінка «Team» зі списком членів команди наведена на рисунку 3.12.

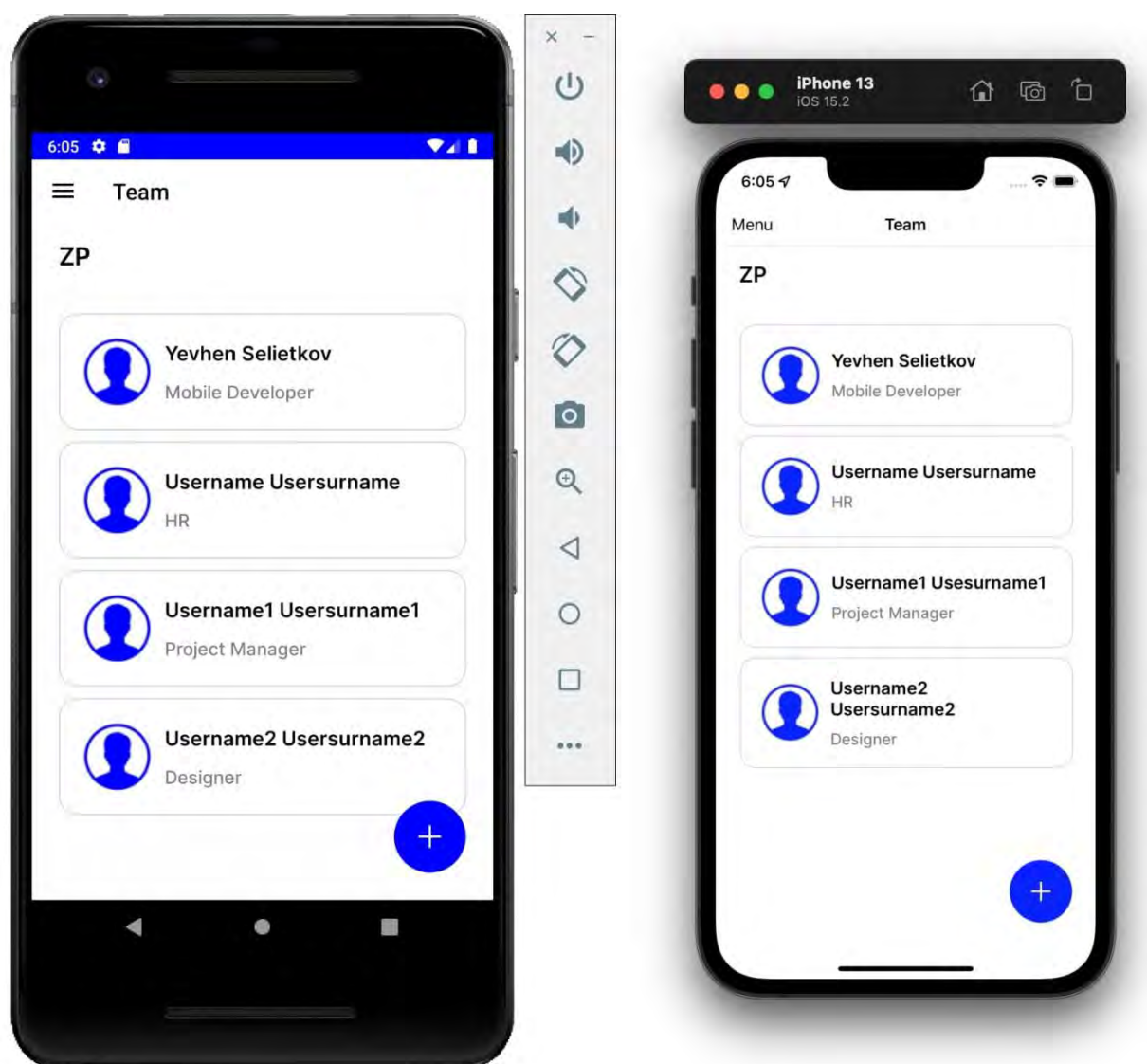


Рисунок 3.12 – Сторінка «Team» зі списком членів команди

Якщо натиснути кнопку з іконкою «плюса», відкриється спливаюча сторінка «Add member» (Додати учасника). Ця сторінка зображена на рисунку 3.13. На сторінці є текстове поле «Search by name, surname or email» (Знайти по імені, прізвищу або email) та дві кнопки «Cancel» та «Add» (Додати). Якщо почати вводити текст у поле, з'явиться список користувачів, які відповідають введеному тексту та яких немає у команді.

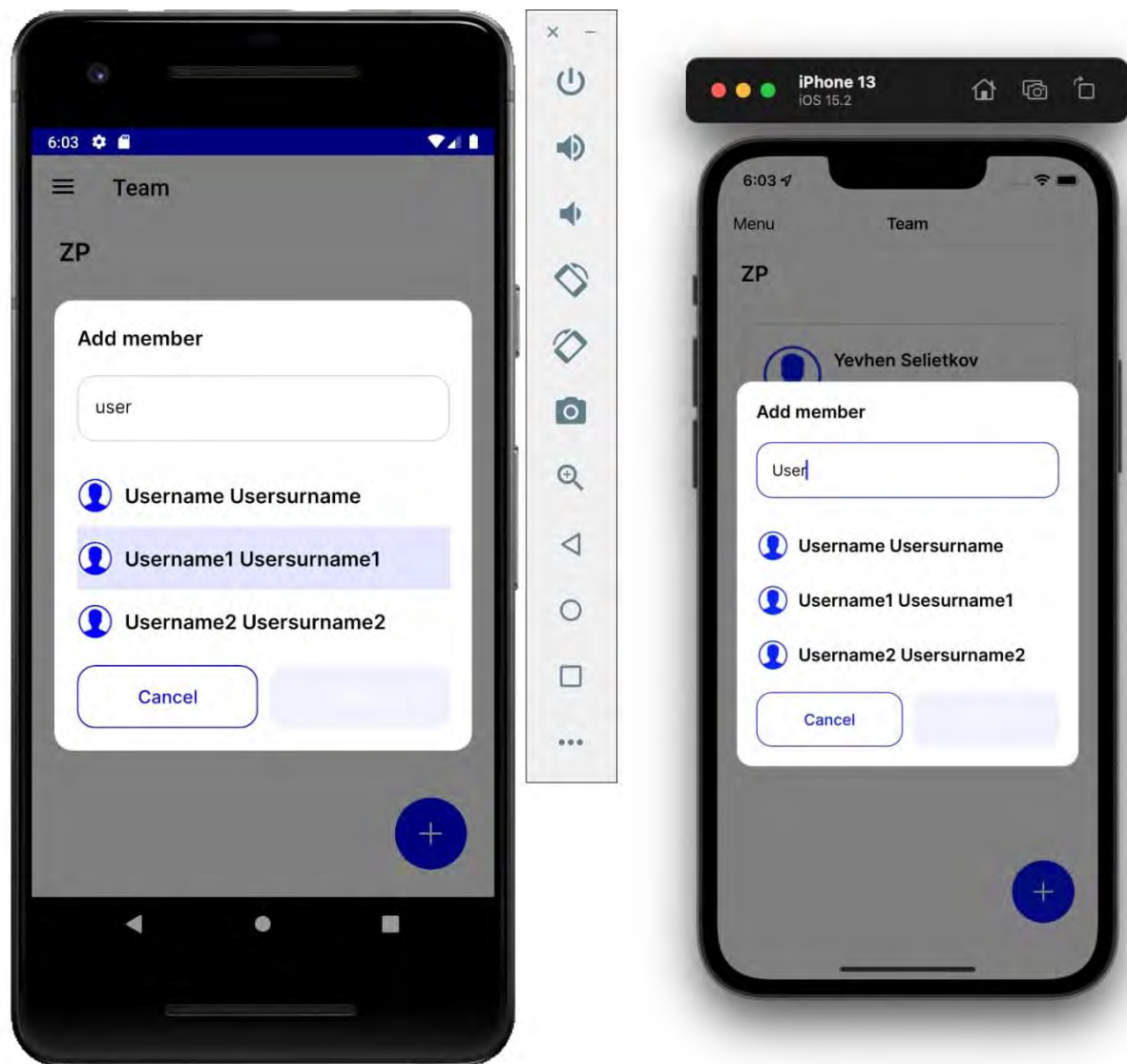


Рисунок 3.13 – Сторінка «Add member»

Натиснувши на одного з користувачів списку, його ім'я, прізвище та фотографія з'являться у текстовому полі. Також справа з'явиться кнопка з іконкою «хрестика», натиснувши на яку, поле буде очищене. На рисунку 3.14 наведено спливаюче вікно «Add member» з обраним користувачем.

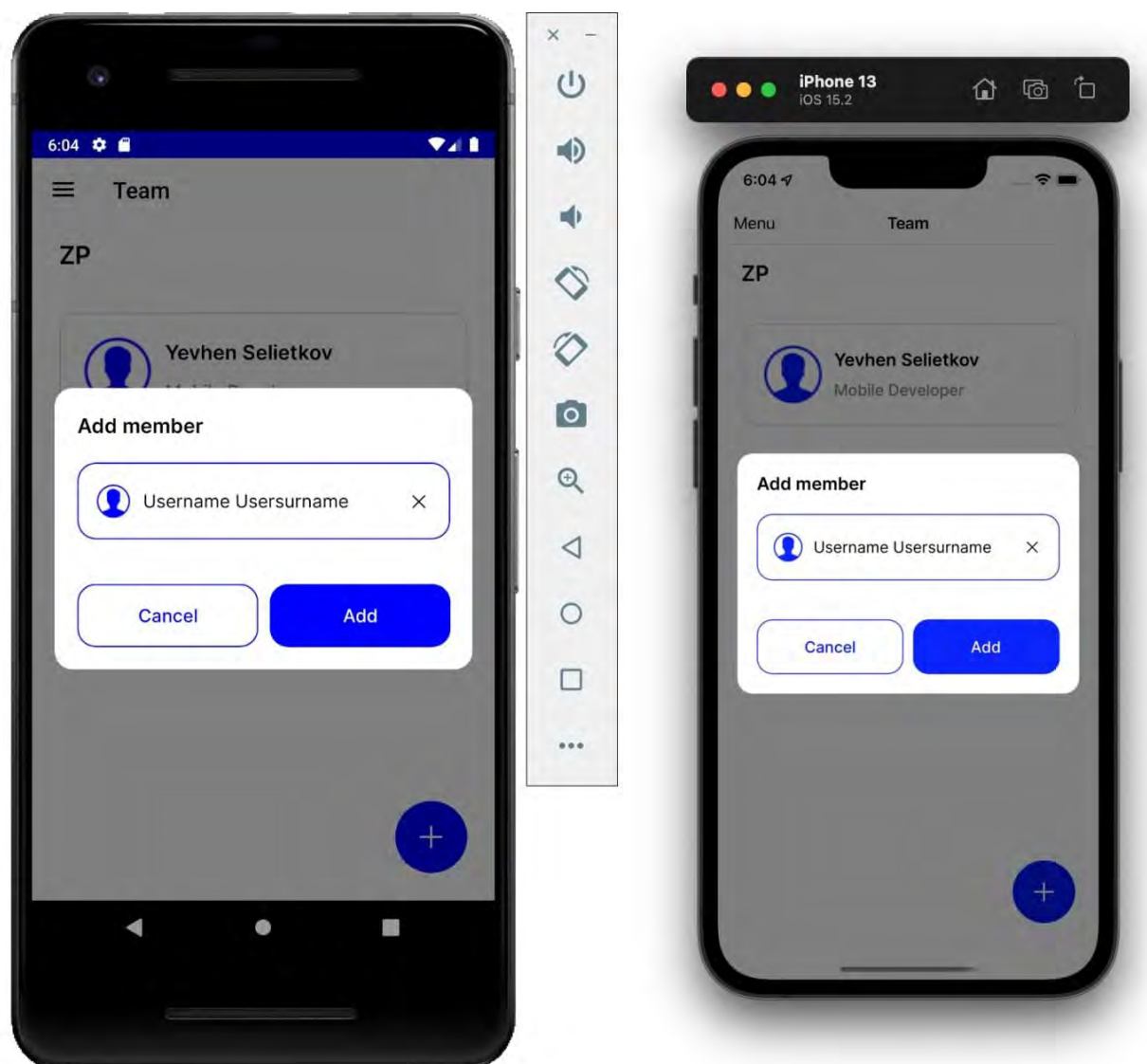


Рисунок 3.14 – Спливаюче вікно «Add member» з обраним користувачем

Натиснувши кнопку «Add», сторінка закриється та обраний користувач додається до команди.

Якщо натиснути на кнопку «Menu» (Меню) на iOS та кнопку з іконкою «бокового меню» на Android в лівому верхньому куті сторінки «Team» відкриється бокове меню. На рисунку 3.15 зображено бокове меню. Зверху цього меню знаходиться коротка інформація про авторизованого користувача: фото, ім'я, прізвище та email. Нижче знаходяться кнопки: «Team», «Leave» (Відпустки) та «Team time» (Командний час). У нижній частині меню знаходиться кнопка «Log Out» (Вийти). При натисненні на кнопку «Team», «Leave» або «Team time» відкриється однойменна сторінка.

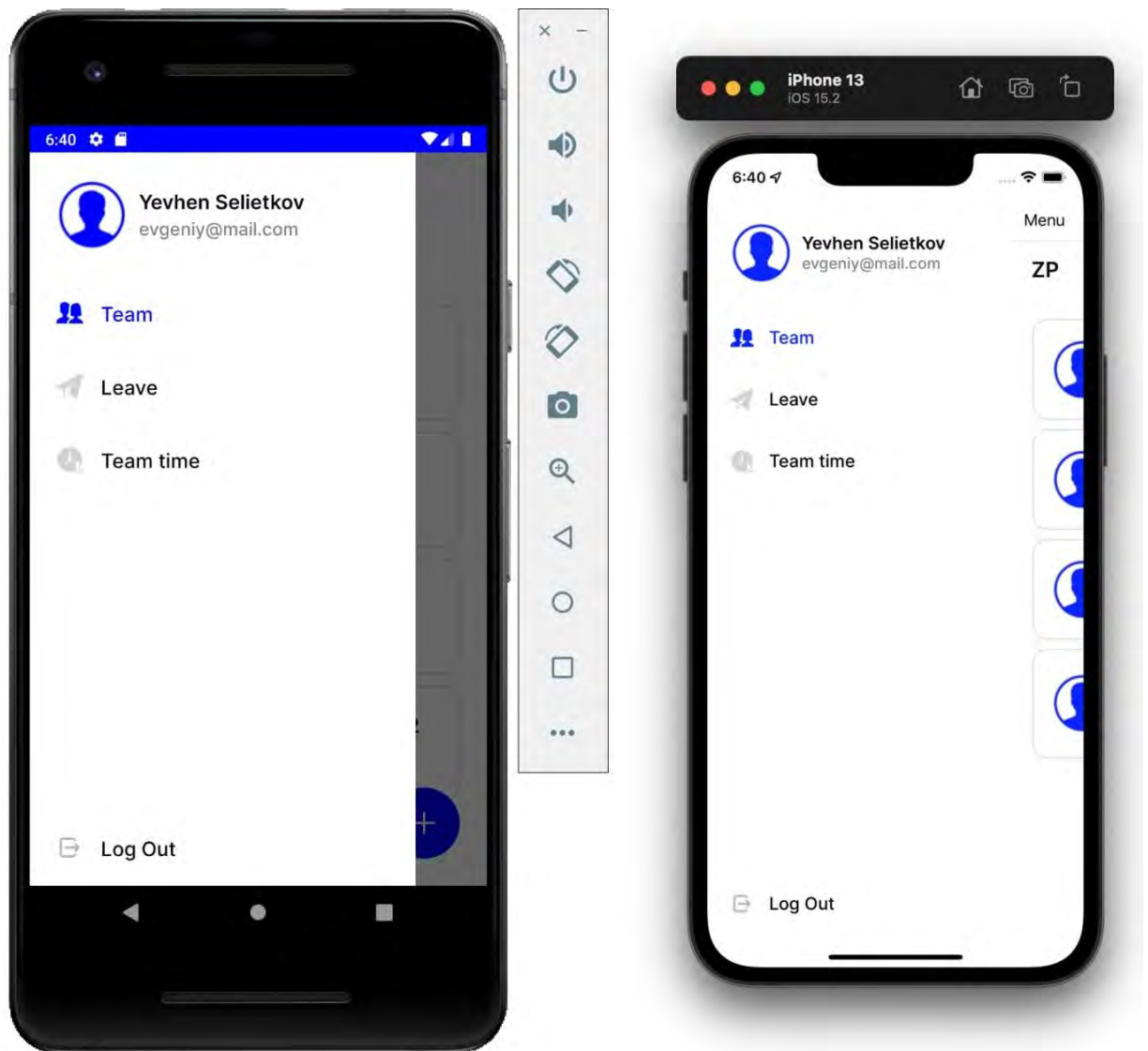


Рисунок 3.15 – Бокове меню

Якщо натиснути кнопку «Log Out», відкриється спливаюче вікно для підтвердження виходу. На рисунку 3.16 наведено це спливаюче вікно. Якщо натиснути «No» (Ні) це спливаюче вікно закриється, якщо натиснути «Yes» (Так) користувач вийде з аккаунта та відкриється сторінка «Login».

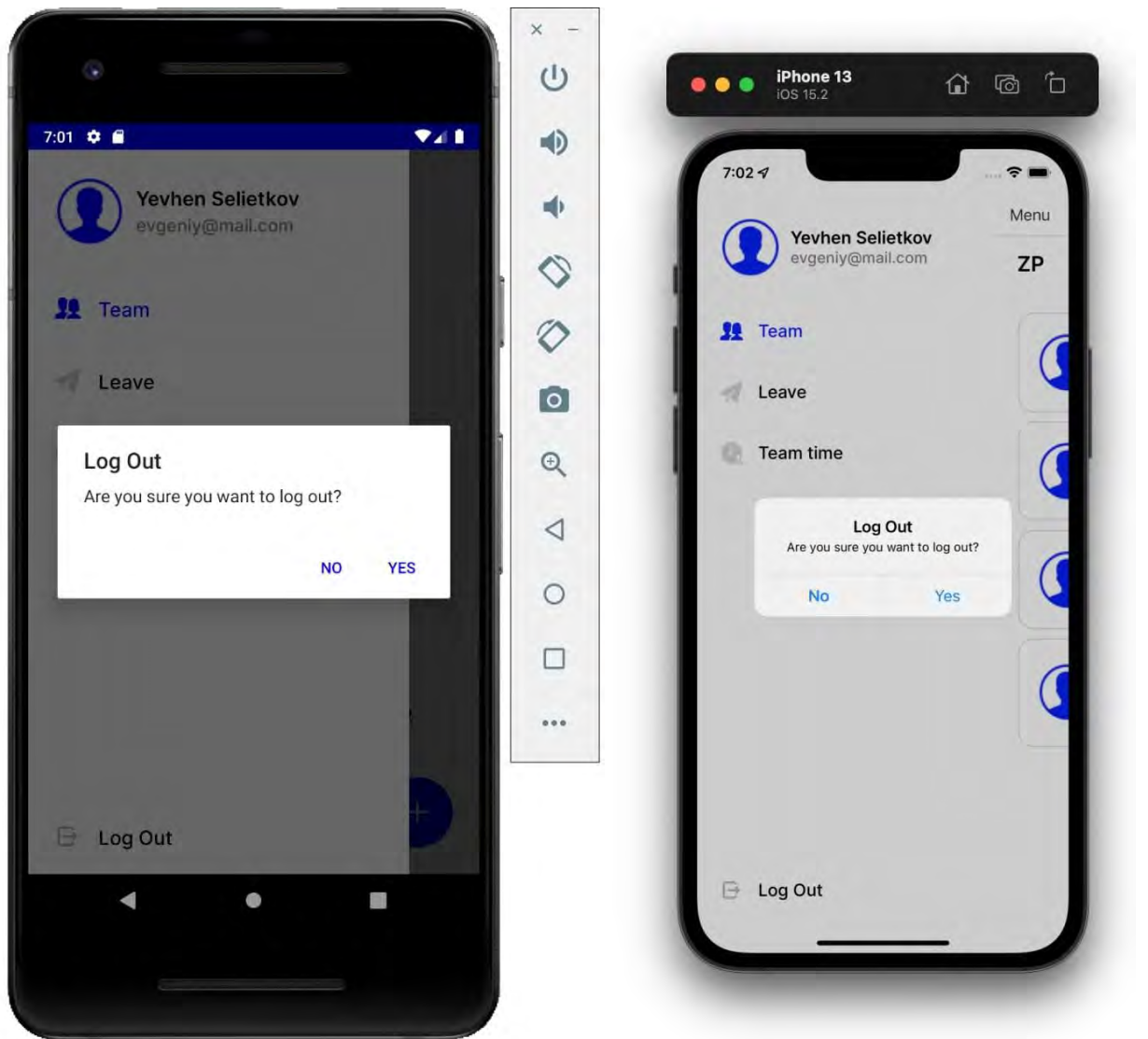


Рисунок 3.16 – Спливаюче вікно для підтвердження виходу

Якщо в меню натиснути на кнопку «Leave», відкриється сторінка зі списком усіх відпусток на тиждень або повідомленням «No leave» (Немає відпусток). Сторінка зі списком відпусток наведено на рисунку 3.17. Сторінка, крім списку відпусток, містить декілька кнопок. Дві кнопки знаходяться у правому верхньому куті: кнопка з іконкою «календаря» та кнопка з іконкою «стрілочки». Внизу знаходиться кругла кнопка з іконкою «плюса».

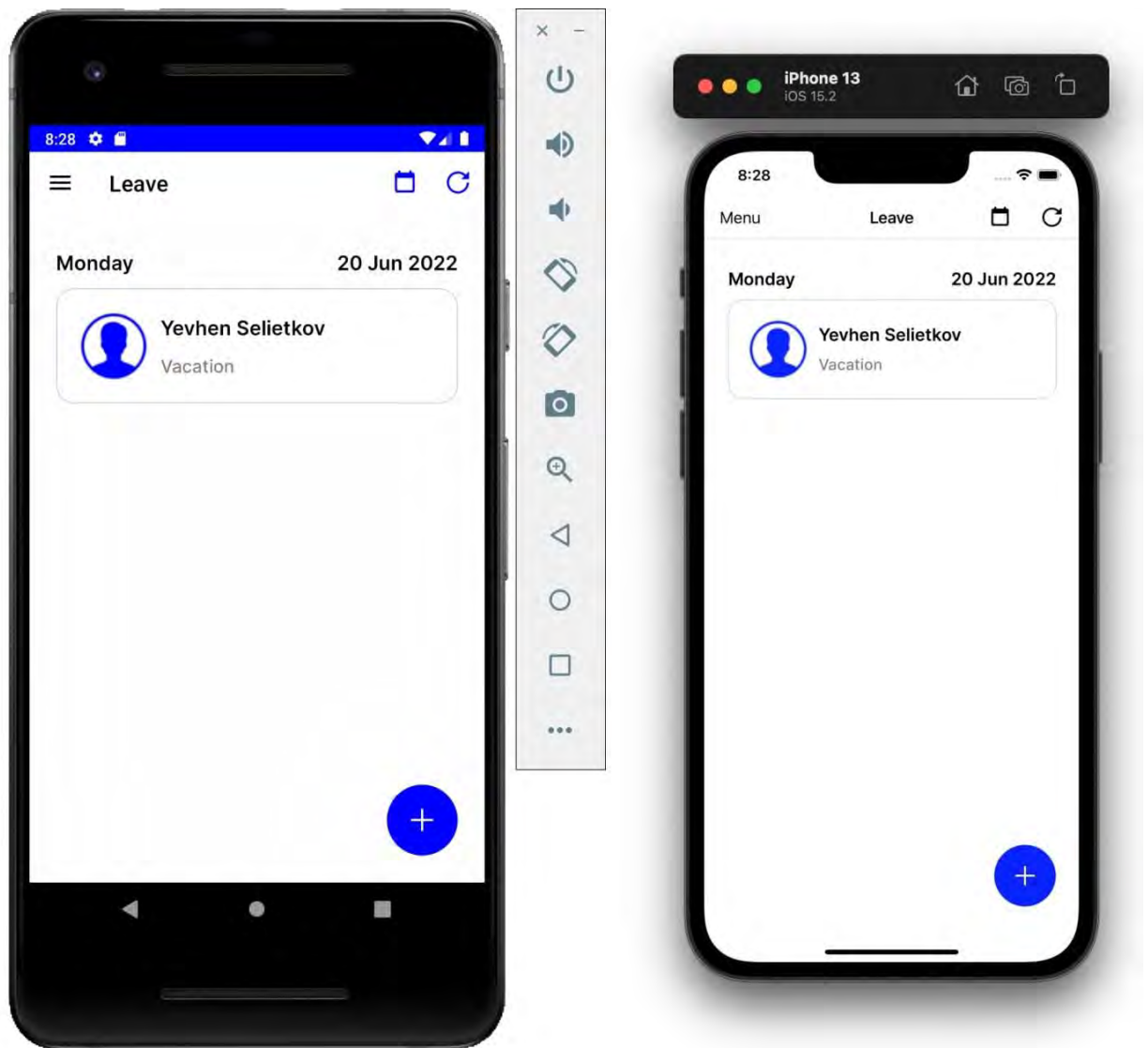


Рисунок 3.17 – Сторінка зі списком відпусток

Якщо натиснути на кнопку з іконкою «календаря», відкриється спливаюче вікно з календарем. Це спливаюче вікно наведено на рисунку 3.18. Після вибору дати, на сторінці «Leave» відобразиться список відпусток тільки за вибраний день.

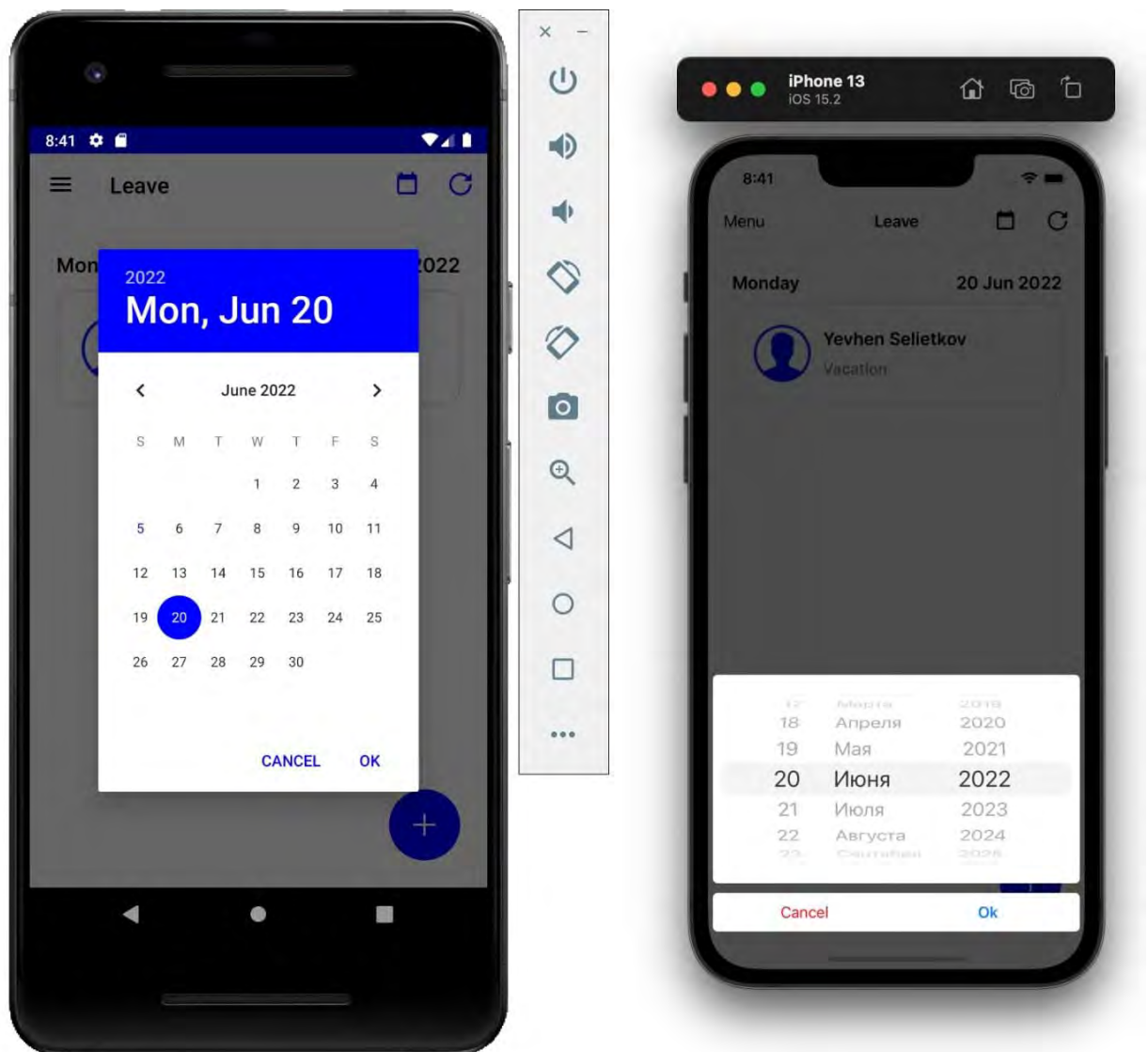


Рисунок 3.18 – Спливаюче вікно з календарем

Якщо натиснути кнопку з іконкою «стрілочки», список знову заповниться відпустками за неділю. Якщо натиснути кнопку з іконкою «плюса», відкриється сторінка «New request» (Новий запит). Ця сторінка зображена на рисунку 3.19. Сторінка містить поле «Select type» (Вибрати тип), а також поля «From» та «To». Внизу знаходиться кнопка «Create request» (Створити запит).

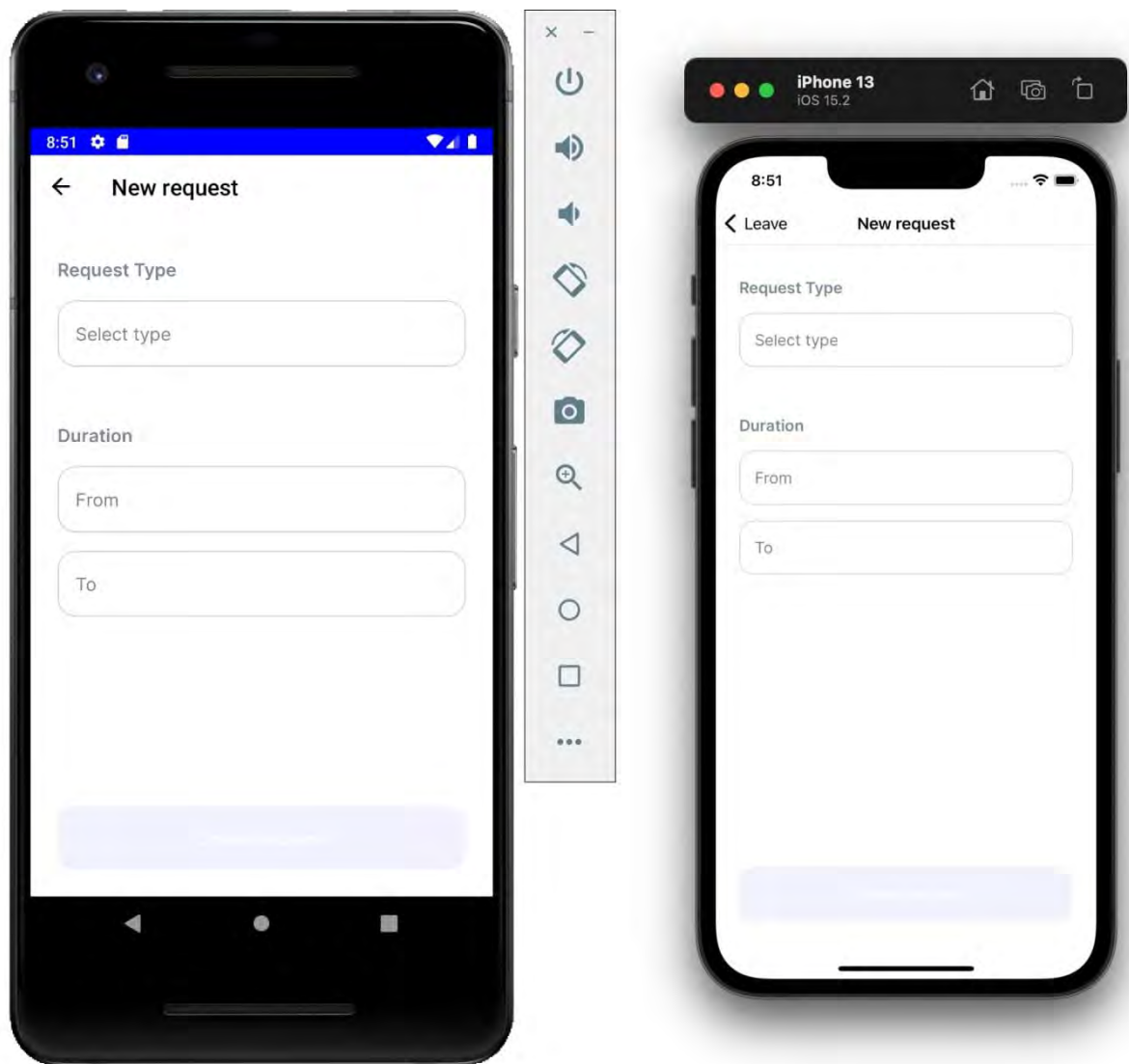


Рисунок 3.19 – Сторінка «New request»

Якщо натиснути на поле «Select type», відкриється спливаюче вікно вибору типу відпустки. Це спливаюче вікно наведено на рисунку 3.20.

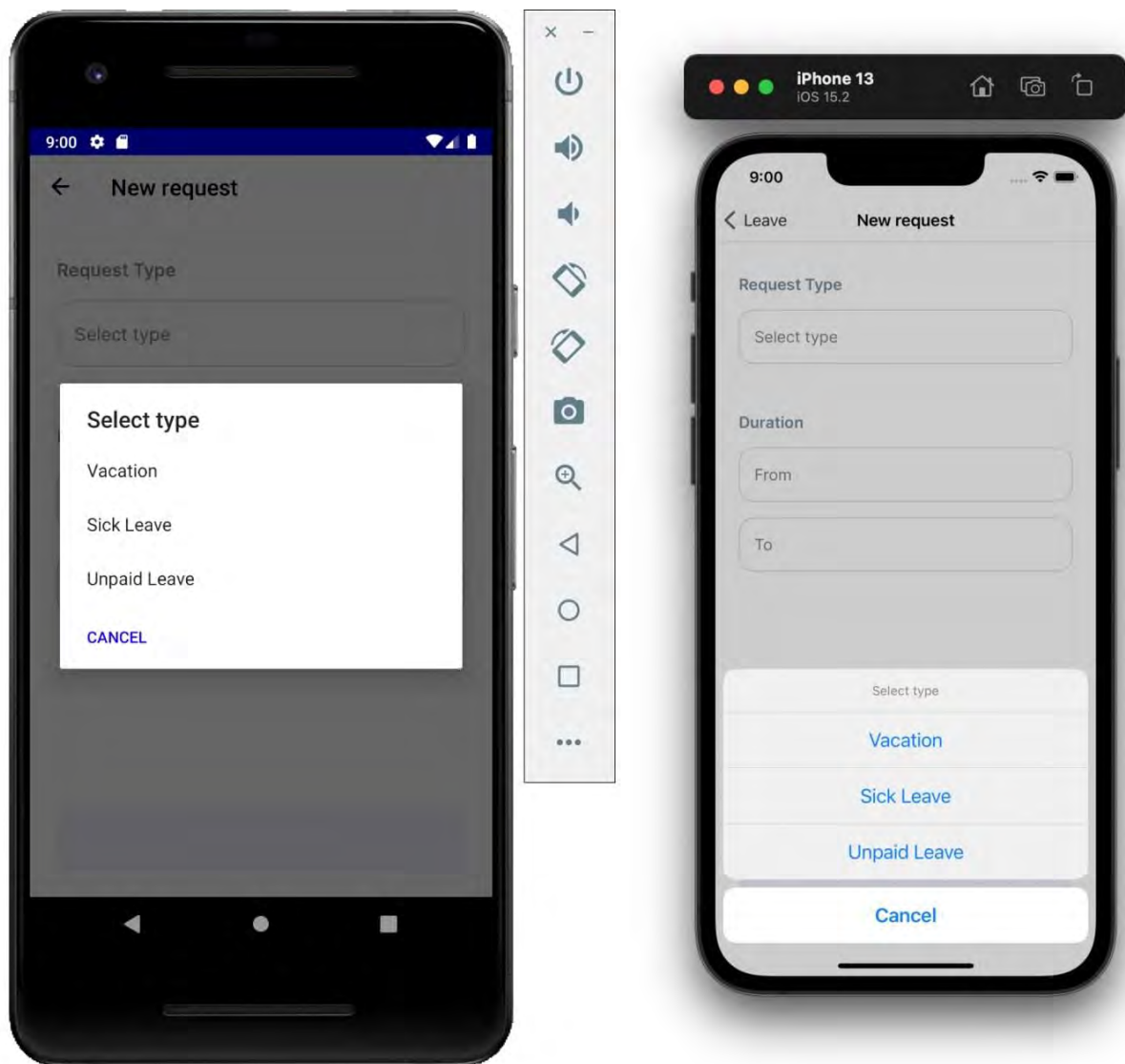


Рисунок 3.20 – Спливаюче вікно вибору типу відпустки

Натиснувши на поле «From» або «To», відкриється спливаюче вікно з календарем. Після заповнення всіх полів та натиснення на кнопку «Create request» створиться запит на відпустку.

Якщо у боковому меню натиснути на кнопку «Team time» (Командний час), відкриється сторінка «Team time». Ця сторінка зображена на рисунку 3.21. На цій сторінці знаходиться список з спільними робочими проміжками часу для різних подій в команді. Крім списку у верхньому правому куті знаходиться кнопка з іконкою «календаря». Для того щоб обрати спільний проміжок часу застосунок аналізує кожного члена команди на наявність відпустки, вихідного, святкового дня, а також робочі години та часовий пояс.

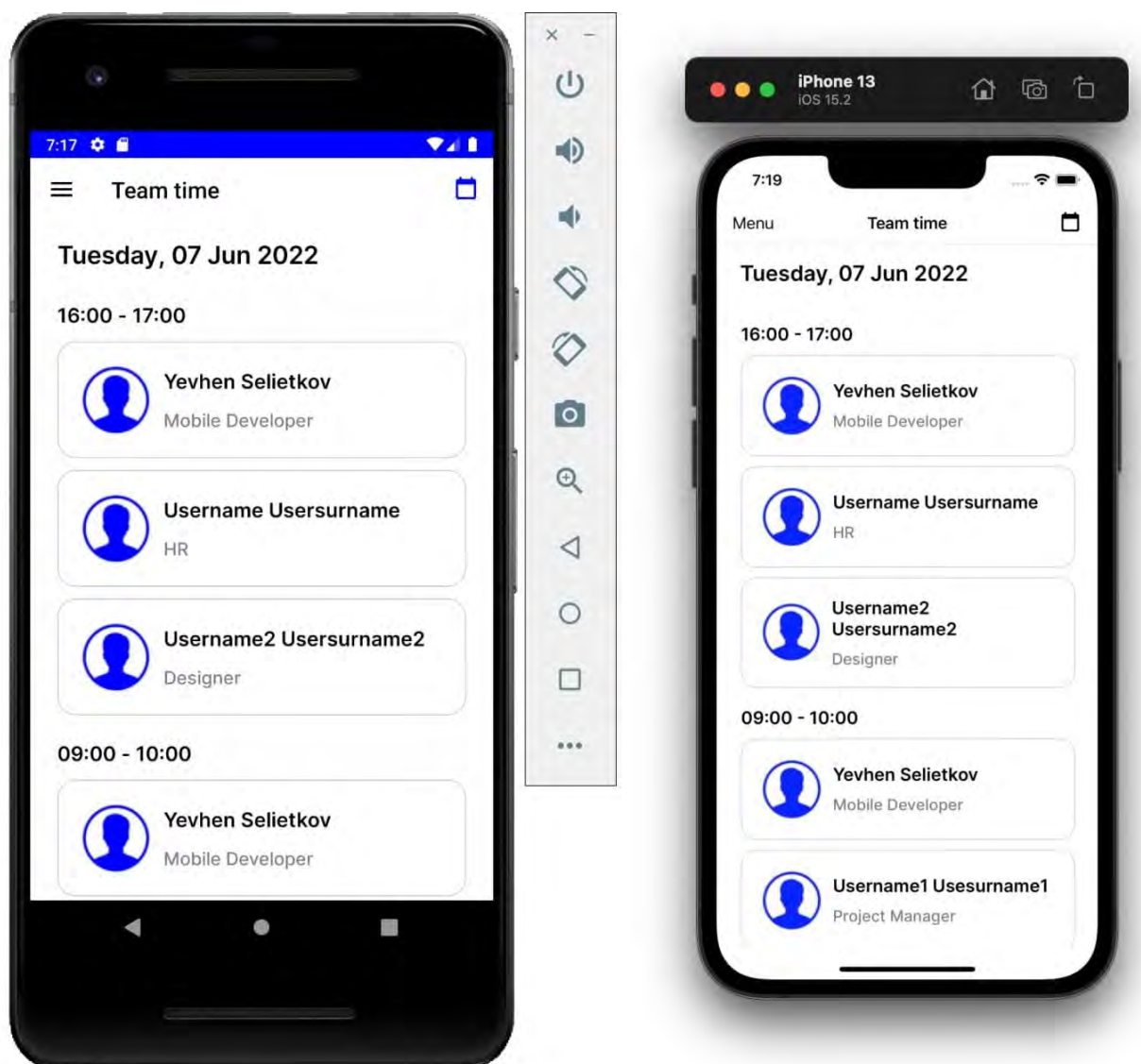


Рисунок 3.21 – Сторінка «Team time»

Натиснувши на кнопку з іконкою «календаря», відкриється спливаюче вікно з календарем, де можна обрати день, після чого список оновиться згідно з обраним днем.

3.3 ViewModel

У кожної сторінки є свій клас `ViewModel`. Кожен такий клас наслідує клас `BaseViewModel`, який наслідує клас `BindableBase`, а також реалізовує інтерфейси `Initialize` та `INavigationAware`. Цей клас та інтерфейси додає фреймворк `Prism.Unity.Forms`.

Самі сторінки також наслідують базовий клас сторінки `BaseContentPage`, у якому прописані необхідні налаштування спільні для всіх сторінок і який наслідує клас `ContentPage` – базовий клас для сторінок у `Xamarin.Forms`.

3.4 Висновки з розділу 3

В процесі роботи було створено мобільний застосунок для керування розподіленою командою. Застосунок був умовно поділений на три частини: `Model`, `View` та `ViewModel`. `Model` теж умовно був поділений на дві частини: класи-моделі та класи-сервіси. Були створені необхідні сторінки, а також класи для взаємодії між інтерфейсом та даними.

Всі сторінки та класи були описані у цьому розділі.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Мінімальні вимоги до апаратних та програмних засобів

Для користування застосунком потрібен смартфон з ОС Android або ОС iOS.

Мінімальна версія ОС Android – 5.0.

Мінімальна версія ОС iOS – 8.0.

4.2 Керівництво користувача

Для того щоб почати роботу у застосунку, необхідно знайти його у списку всіх застосунків у смартфоні, після чого запустити натиснувши на його іконку.

Після запуску застосунка відкривається сторінка входу, де зареєстрований користувач може, ввівши свої email та пароль у текстові поля та натиснувши кнопку «Log in», увійти у свій аккаунт.

Якщо користувач не зареєстрований, необхідно натиснути кнопку «Sign up» На сторінці, яка відкрилась необхідно ввести своє ім'я та прізвище та натиснути кнопку «Next». На наступній сторінці необхідно ввести email та пароль та натиснути кнопку «Create account».

Після того як аккаунт був створений, користувач може увійти у свій аккаунт, але перед входом на головну сторінку «Team» необхідно завершити реєстрацію, вказавши необхідні дані: локацію, у якій мешкає користувач, посаду, а також проміжок робочого часу. Для цього потрібно ввести email та пароль, які були введені при реєстрації та натиснути кнопку «Log in». Після цього відкриється сторінка з картою, де потрібно обрати локацію та натиснути кнопку «Next». На сторінці, яка відкриється необхідно обрати посаду, а також час початку та кінця роботи, після чого натиснути кнопку «Complete registration». На цьому процес реєстрації буде завершено.

Головною сторінкою застосунку є «Team». На цій сторінці зареєстрований користувач може переглянути список членів команди, якщо він є її частиною, або побачити кнопку для створення нової команди.

Якщо користувач не знаходиться у складі команди, він може створити нову. Для цього необхідно на сторінці «Team» натиснути кнопку «Create a new team», після чого відкриється спливаюча сторінка. На цій сторінці потрібно ввести назву команди та натиснути кнопку «Create». Тепер на сторінці з'явиться список, в якому буде тільки один користувач, а також з'явиться нова кнопка для того щоб додати нових членів команди.

Користувач, який є членом команди може додати до команди інших користувачів. Для того щоб додати користувачів, потрібно натиснути на кнопку з іконкою «плюса» на сторінці «Team». У спливаючій сторінці, яка відкриється потрібно ввести ім'я, прізвище або email користувача. Якщо він буде знайдений, з'явиться список з можливими користувачами. Щоб додати певного користувача, необхідно натиснути на нього у списку та натиснути кнопку «Add».

Зареєстрований користувач може переглянути список відпусток команди. Для цього потрібно відкрити бокове меню, натиснувши у верхньому лівому куті сторінки «Team» кнопку «Menu» на ОС iOS або кнопку з іконкою «бокового меню» на ОС Android. Далі потрібно натиснути на кнопку «Leave». На сторінці, яка відкриється можна переглянути список відпусток за неділю.

Для того щоб створити запит на відпустку, треба на сторінці «Leave» натиснути круглу кнопку з іконкою «плюса», після чого відкриється сторінка «New request». На цій сторінці потрібно обрати тип відпустки: відпустка, лікарняний або за свій рахунок, а також день початку та день кінця відпустки і натиснути кнопку «Create request».

Зареєстрований користувач також може переглянути список спільних для команди робочих проміжків часу. Для цього потрібно у боковому меню

натиснути кнопку «Team time». На сторінці, яка відкриється можна переглянути список робочих годин спільних для команди.

Зареєстрований користувач може вийти зі свого аккаунта. Для цього потрібно відкрити бокове меню та натиснути кнопку «Log Out», після чого відкриється спливаюче вікно, у якому необхідно підтвердити вихід, натиснувши кнопку «Yes».

4.3 Тестування

Було проведено тестування усього можливого функціоналу для всіх типів користувачів: не зареєстрований, зареєстрований, не в команді та в команді. На рисунках 4.1-4.2 наведено use-case діаграми функцій користувачів.

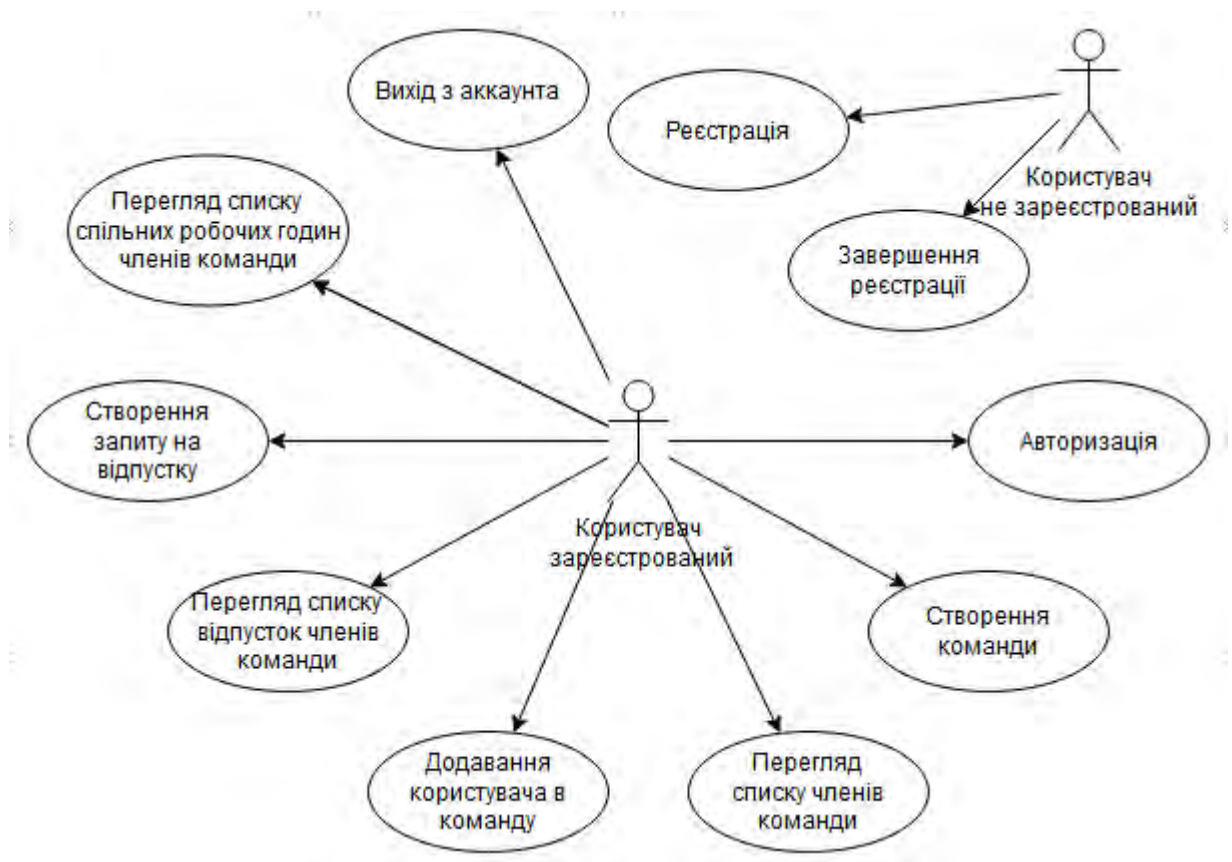


Рисунок .4.1 – Use-case діаграма функцій користувача

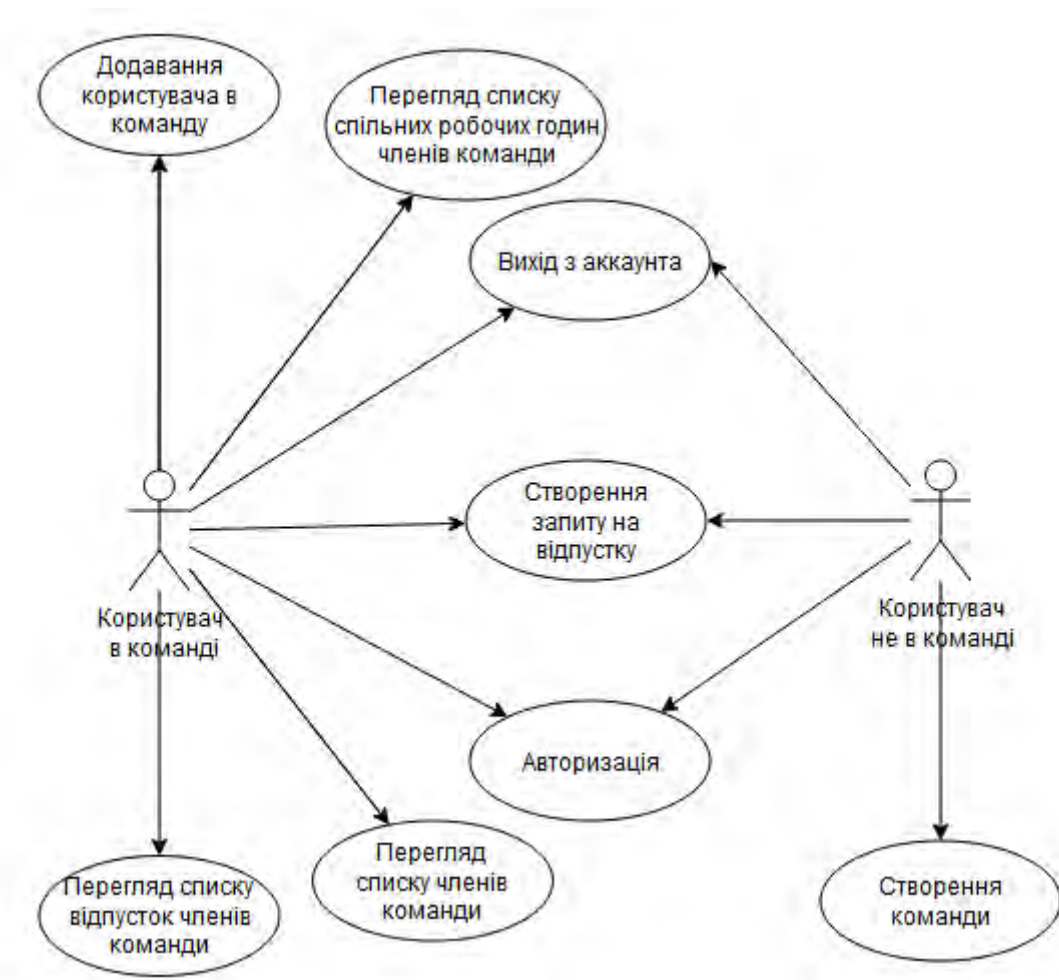


Рисунок .4.2 – Use-case діаграма функцій користувача, який в команді та користувача, який не в команді

Усі результати тестування наведені в таблиці 4.1.

Таблиця 4.1 – Результати тестування застосунку

Функції	Зареєстрований користувач	Не зареєстрований користувач	Користувач не в команді	Користувач в команді
1	2	3	4	5
Реєстрація	-	+	-	-
Завершення реєстрації	-	+	-	-
Авторизація	+	-	+	+
Створення команди	+	-	+	-
Перегляд списку членів команди	+	-	-	+

Продовження таблиці 4.1

1	2	3	4	5
Додавання користувача в команду	+	-	-	+
Перегляд списку відпусток членів команди	+	-	-	+
Створення запиту на відпустку	+	-	+	+
Перегляд списку спільних робочих годин членів команди	+	-	-	+
Вихід з аккаунта	+	-	+	+

Для не зареєстрованого користувача були протестовані функції реєстрації та завершення реєстрації. Успішним результатом реєстрації для користувача є створений аккаунт, дані якого збережені в БД. Успішним результатом завершення реєстрації вважається оновлений аккаунт з даними про локацію, часовий пояс, посаду та робочі години.

Для зареєстрованого користувача, який не є членом команди були протестовані функції авторизації, створення команди, створення запиту на відпустку, виходу з аккаунта. Успішним результатом авторизації є підтвердження введених email та пароля і відкриття головної сторінки «Team». Успішним результатом створення команди є поява списку членів команди з назвою, яка була введена при створенні. Успішним результатом створення запиту на відпустку є поява відпустки у списку відпусток. Успішним результатом виходу з аккаунта є відкриття сторінки входу, та подальше її відкриття після запуску застосунка.

Для зареєстрованого користувача, який є членом команди були протестовані функції авторизації, перегляд списку членів команди, додавання користувачів в команду, перегляд списку відпусток, створення запиту на відпустку, перегляд списку спільних робочих годин членів команди та виходу з аккаунта. Успішним результатом додавання користувача в команду

є наявність його у списку членів команди. Успішним результатом виконання функцій отримання списків: членів команди, відпусток та спільних робочих годин є відображення цих списків на відповідних сторінках «Team», «Leave» та «Team time».

Всі тести пройдені успішно. З цього можна зробити висновок, що застосунок працює справно та стабільно.

ВИСНОВКИ

Для організації роботи розподіленої команди було прийнято рішення розробити мобільний застосунок. Було обрано одразу дві мобільні платформи: ОС Android та ОС iOS. Під час розробки була проаналізована предметна область, а також програмні засоби, які схожі за функціоналом.

Під час аналізу технологій для розробки мобільних застосунків, була обрана мова програмування C# та фреймворк Xamarin, який дозволяє створювати кросплатформові застосунки як мобільні так і для персональних комп'ютерів. У майбутньому за бажанням можна без проблем, майже не доповнюючи та не переписуючи код програми, додати інші платформи.

Для розробки застосунку був обраний архітектурний шаблон проєктування MVVM, як найпопулярніший для розробки мобільних застосунків.

Під час написання коду було створено 4 моделі, 8 сервісів, 12 сторінок та стільки ж вьюмоделей, які відповідають сторінкам.

Були проведені всі тести функціоналу для кожного типу користувача. Всі тести були пройдені успішно.

Повний код проєкту наведено у додатку А.

Створений застосунок планується розвивати й далі. Список функціоналу, який планується реалізувати:

- серверна частина застосунку;
- можливість створення декількох команд;
- новий тип користувача адміністратор;
- можливість надати користувачу прав адміністратора;
- можливість підтверджувати та відмовляти в запиті на відпустку адміністратором;
- можливість підтверджувати вступ у команду;
- можливість редагувати аккаунт.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Розподілені команди: нова реальність та допоміжні інструменти [Електрон. ресурс] / – Режим доступу: <https://jobs.innovects.com/uk/blog/rozpodileni-komandy-instumenty/>.
2. Як зробити віддалену роботу в компанії ефективною [Електрон. ресурс] / – Режим доступу: <https://dou.ua/lenta/articles/distributed-teams-are-future/>.
3. Длугунович, Н. Засади створення системи комунікаційної взаємодії при розробці програмного забезпечення [Електрон. ресурс] / Н. Длугунович, Ю. Форкун, Л. Ряба – Режим доступу: <http://miljournals.knu.ua/index.php/zbirnuk/article/download/73/59/217>.
4. Для чего используется Jira Software? [Электрон. ресурс] / – Режим доступа: <https://www.atlassian.com/ru/software/jira/guides/use-cases/what-is-jira-used-for#jira-for-bug-tracking-features>.
5. Jira [Електрон. ресурс] / – Режим доступу: <https://uk.wikipedia.org/wiki/Jira>.
6. Trello [Електрон. ресурс] / – Режим доступу: <https://trello.com/uk/guide/trello-101>.
7. Что такое Trello и как им пользоваться? [Электрон. ресурс] / – Режим доступа: <https://hostiq.ua/blog/what-is-trello-2/>.
8. Slack [Електрон. ресурс] / – Режим доступу: <https://uk.wikipedia.org/wiki/Slack>.
9. Что такое GitHub и для чего именно он нужен? [Электрон. ресурс] / – Режим доступа: <https://www.hostinger.com.ua/rukovodstva/chto-takoe-github/>.
10. Hurma – автоматизация HR-процессов и рекрутинга [Электрон. ресурс] / – Режим доступа: <https://dou.ua/lenta/articles/dou-projector-hurma/>.
11. Языки для разработки мобильных приложений на iOS и Android – какой выбрать? [Электрон. ресурс] / – Режим доступа:

<https://appcraft.pro/blog/yazyki-dlya-razrabotki-mobilnykh-prilozhenij/>.

12. C Sharp [Электрон. ресурс] / – Режим доступа: https://ru.wikipedia.org/wiki/C_Sharp.

13. Flutter VS Xamarin VS React Native: What's the best in 2021? [Electron. resource] / – Access source: <https://www.ego-cms.com/post/flutter-vs-xamarin-vs-react-native-whats-best-in-2021>.

14. Xamarin Native vs Xamarin.Forms: How to choose? [Electron. resource] / – Access source: <https://dzone.com/articles/xamarin-native-vs-xamarinforms-how-to-choose>.

15. Rider: кросс-платформенная IDE для .NET [Электрон. ресурс] / – Режим доступа: <https://www.jetbrains.com/ru-ru/rider/>.

16. ReSharper [Электрон. ресурс] / – Режим доступа: <https://uk.wikipedia.org/wiki/ReSharper>.

17. Что такое MVVM и как это работает [Электрон. ресурс] / – Режим доступа: <https://highload.today/kratkoe-rukovodstvo-po-mvvm-klyuchevye-momenty-i-primery/>.

18. Prism [Electron. resource] / – Access source: <https://prismlibrary.com/docs/index.html>.

ДОДАТОК А
Текст програми

BorderlessEntry.cs

```
namespace TeamConnect.Controls
{
    public class BorderlessEntry : Entry
    {
    }
}
```

CustomEntry.xaml

```
<StackLayout xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:controls="clr-namespace:TeamConnect.Controls"
              x:Class="TeamConnect.Controls.CustomEntry"
              x:Name="customEntry">

    <Frame x:Name="frame"
          BindingContext="{x:Reference customEntry}"
          HasShadow="False"
          Padding="16, 18"
          CornerRadius="16"
          BackgroundColor="Transparent">

        <Frame.GestureRecognizers>

            <TapGestureRecognizer Tapped="EntryTapped"/>

        </Frame.GestureRecognizers>

        <Grid ColumnDefinitions="*, Auto"
              ColumnSpacing="12">

            <controls:BorderlessEntry x:Name="entry"
                                      Grid.Column="0"
                                      BindingContext="{x:Reference
customEntry}"

                                      VerticalOptions="Center"
                                      FontFamily="{Binding FontFamily}"
                                      FontSize="{Binding FontSize}"
                                      TextColor="{Binding TextColor}"
                                      PlaceholderColor="{Binding
PlaceholderColor}"

                                      Placeholder="{Binding Placeholder}"
                                      Keyboard="{Binding Keyboard}"
                                      TranslationY="{OnPlatform Android=-2,
iOS=0}"

                                      Text="{Binding Text}"
                                      IsPassword="{Binding IsPassword}"
                                      Focused="EntryFocused"
                                      Unfocused="EntryUnfocused"/>

            <Image x:Name="passwordButton"
                  Grid.Column="1"
                  IsVisible="{Binding IsPasswordButtonVisible,
Source={x:Reference customEntry}}">

                <Image.GestureRecognizers>

                    <TapGestureRecognizer Tapped="PasswordButtonTapped"/>

```

```

        </Image.GestureRecognizers>

    </Image>

</Grid>

</Frame>

<Label BindingContext="{x:Reference customEntry}"
    IsVisible="{Binding IsError}"
    FontFamily="{Binding ErrorFontFamily}"
    FontSize="{Binding ErrorFontSize}"
    Text="{Binding ErrorText}"
    TextColor="{Binding ErrorColor}"/>

</StackLayout>

```

CustomEntry.xaml.cs

```

namespace TeamConnect.Controls
{
    public partial class CustomEntry
    {
        public CustomEntry()
        {
            InitializeComponent();
        }

        #region -- Public properties --

        public static readonly BindableProperty BorderColorProperty =
BindableProperty.Create(
    propertyName: nameof(BorderColor),
    returnType: typeof(Color),
    declaringType: typeof(CustomEntry));

        public Color BorderColor
        {
            get => (Color)GetValue(BorderColorProperty);
            set => SetValue(BorderColorProperty, value);
        }

        public static readonly BindableProperty TextProperty =
BindableProperty.Create(
    propertyName: nameof(Text),
    returnType: typeof(string),
    declaringType: typeof(CustomEntry),
    defaultBindingMode: BindingMode.TwoWay);

        public string Text
        {
            get => (string)GetValue(TextProperty);
            set => SetValue(TextProperty, value);
        }

        public static readonly BindableProperty PlaceholderProperty =
BindableProperty.Create(
    propertyName: nameof(Placeholder),
    returnType: typeof(string),
    declaringType: typeof(CustomEntry));

```

```

        public string Placeholder
        {
            get => (string)GetValue(PlaceholderProperty);
            set => SetValue(PlaceholderProperty, value);
        }

        public static readonly BindableProperty FontFamilyProperty =
BindableProperty.Create(
            propertyName: nameof(FontFamily),
            returnType: typeof(string),
            declaringType: typeof(CustomEntry));

        public string FontFamily
        {
            get => (string)GetValue(FontFamilyProperty);
            set => SetValue(FontFamilyProperty, value);
        }

        public static readonly BindableProperty FontSizeProperty =
BindableProperty.Create(
            propertyName: nameof(FontSize),
            returnType: typeof(double),
            declaringType: typeof(CustomEntry));

        public double FontSize
        {
            get => (double)GetValue(FontSizeProperty);
            set => SetValue(FontSizeProperty, value);
        }

        public static readonly BindableProperty TextColorProperty =
BindableProperty.Create(
            propertyName: nameof(TextColor),
            returnType: typeof(Color),
            declaringType: typeof(CustomEntry));

        public Color TextColor
        {
            get => (Color)GetValue(TextColorProperty);
            set => SetValue(TextColorProperty, value);
        }

        public static readonly BindableProperty PlaceholderColorProperty =
BindableProperty.Create(
            propertyName: nameof(PlaceholderColor),
            returnType: typeof(Color),
            declaringType: typeof(CustomEntry));

        public Color PlaceholderColor
        {
            get => (Color)GetValue(PlaceholderColorProperty);
            set => SetValue(PlaceholderColorProperty, value);
        }

        public static readonly BindableProperty IsPasswordProperty =
BindableProperty.Create(
            propertyName: nameof(IsPassword),
            returnType: typeof(bool),
            declaringType: typeof(CustomEntry));

        public bool IsPassword
        {

```

```

        get => (bool)GetValue(IsPasswordProperty);
        set => SetValue(IsPasswordProperty, value);
    }

    public static readonly BindableProperty
IsPasswordButtonVisibleProperty = BindableProperty.Create(
    propertyName: nameof(IsPasswordButtonVisible),
    returnType: typeof(bool),
    declaringType: typeof(CustomEntry));

    public bool IsPasswordButtonVisible
    {
        get => (bool)GetValue(IsPasswordButtonVisibleProperty);
        set => SetValue(IsPasswordButtonVisibleProperty, value);
    }

    public static readonly BindableProperty NotPasswordIconProperty =
BindableProperty.Create(
    propertyName: nameof(NotPasswordIcon),
    returnType: typeof(ImageSource),
    declaringType: typeof(CustomEntry));

    public ImageSource NotPasswordIcon
    {
        get => (ImageSource)GetValue(NotPasswordIconProperty);
        set => SetValue(NotPasswordIconProperty, value);
    }

    public static readonly BindableProperty PasswordIconProperty =
BindableProperty.Create(
    propertyName: nameof(PasswordIcon),
    returnType: typeof(ImageSource),
    declaringType: typeof(CustomEntry));

    public ImageSource PasswordIcon
    {
        get => (ImageSource)GetValue(PasswordIconProperty);
        set => SetValue(PasswordIconProperty, value);
    }

    public static readonly BindableProperty FocusedColorProperty =
BindableProperty.Create(
    propertyName: nameof(FocusedColor),
    returnType: typeof(Color),
    declaringType: typeof(CustomEntry));

    public Color FocusedColor
    {
        get => (Color)GetValue(FocusedColorProperty);
        set => SetValue(FocusedColorProperty, value);
    }

    public static readonly BindableProperty IsErrorProperty =
BindableProperty.Create(
    propertyName: nameof(IsError),
    returnType: typeof(bool),
    declaringType: typeof(CustomEntry),
    defaultBindingMode: BindingMode.TwoWay);

    public bool IsError
    {
        get => (bool)GetValue(IsErrorProperty);

```

```

        set => SetValue(IsErrorProperty, value);
    }

    public static readonly BindableProperty ErrorTextProperty =
BindableProperty.Create(
    propertyName: nameof(ErrorText),
    returnType: typeof(string),
    declaringType: typeof(CustomEntry));

    public string ErrorText
    {
        get => (string)GetValue(ErrorTextProperty);
        set => SetValue(ErrorTextProperty, value);
    }

    public static readonly BindableProperty ErrorFontFamilyProperty =
BindableProperty.Create(
    propertyName: nameof(ErrorFontFamily),
    returnType: typeof(string),
    declaringType: typeof(CustomEntry));
    public string ErrorFontFamily
    {
        get => (string)GetValue(ErrorFontFamilyProperty);
        set => SetValue(ErrorFontFamilyProperty, value);
    }

    public static readonly BindableProperty ErrorFontSizeProperty =
BindableProperty.Create(
    propertyName: nameof(ErrorFontSize),
    returnType: typeof(double),
    declaringType: typeof(CustomEntry));

    public double ErrorFontSize
    {
        get => (double)GetValue(ErrorFontSizeProperty);
        set => SetValue(ErrorFontSizeProperty, value);
    }

    public static readonly BindableProperty ErrorColorProperty =
BindableProperty.Create(
    propertyName: nameof(ErrorColor),
    returnType: typeof(Color),
    declaringType: typeof(CustomEntry));

    public Color ErrorColor
    {
        get => (Color)GetValue(ErrorColorProperty);
        set => SetValue(ErrorColorProperty, value);
    }

    public static readonly BindableProperty IsEntryFocusedProperty =
BindableProperty.Create(
    propertyName: nameof(IsEntryFocused),
    returnType: typeof(bool),
    declaringType: typeof(CustomEntry));

    private bool IsEntryFocused
    {
        get => (bool)GetValue(IsEntryFocusedProperty);
        set => SetValue(IsEntryFocusedProperty, value);
    }

```

```

        public static readonly BindableProperty KeyboardProperty =
BindableProperty.Create(
    propertyName: nameof(Keyboard),
    returnType: typeof(Keyboard),
    declaringType: typeof(CustomEntry));

    public Keyboard Keyboard
    {
        get => (Keyboard)GetValue(KeyboardProperty);
        set => SetValue(KeyboardProperty, value);
    }

    #endregion

    #region -- Overrides --

    protected override void OnPropertyChanged([CallerMemberName] string
propertyName = null)
    {
        base.OnPropertyChanged(propertyName);

        if (propertyName == BorderColorProperty.PropertyName)
        {
            SetFrameColor();
        }
        else if (propertyName == IsEntryFocusedProperty.PropertyName)
        {
            IsError = false;
            SetFrameColor();
        }
        else if (propertyName == IsErrorProperty.PropertyName)
        {
            SetFrameColor();
        }
        else if (propertyName == IsPasswordProperty.PropertyName)
        {
            passwordButton.RemoveDynamicResource(Image.SourceProperty);
            passwordButton.Source = IsPassword ? PasswordIcon :
NotPasswordIcon;
        }
    }

    #endregion

    #region -- Private helpers --

    private void SetFrameColor()
    {
        frame.RemoveDynamicResource(Frame.BorderColorProperty);
        frame.BorderColor = IsEntryFocused ? FocusedColor : IsError ?
ErrorColor : BorderColor;
    }

    private void PasswordButtonTapped(object sender, EventArgs e)
    {
        IsPassword = !IsPassword;
    }

    private void EntryFocused(object sender, FocusEventArgs e)
    {
        IsEntryFocused = true;
    }

```

```

        private void EntryUnfocused(object sender, FocusEventArgs e)
        {
            IsEntryFocused = false;
        }

        private void EntryTapped(object sender, EventArgs e)
        {
            entry.Focus();
        }

        #endregion
    }
}

```

CustomMap.cs

```

namespace TeamConnect.Controls
{
    public class CustomMap : Map
    {
        #region -- Public properties --

        public static readonly BindableProperty PinProperty =
BindableProperty.Create(
            propertyName: nameof(Pin),
            returnType: typeof(Pin),
            declaringType: typeof(CustomMap));

        public Pin Pin
        {
            get => (Pin)GetValue(PinProperty);
            set => SetValue(PinProperty, value);
        }

        #endregion

        #region -- Overrides --

        protected override void OnPropertyChanged([CallerMemberName] string
propertyName = null)
        {
            base.OnPropertyChanged(propertyName);

            if (propertyName == nameof(Pin))
            {
                ClearPins();
                SetPin();
            }
        }

        #endregion

        #region -- Private helpers --

        private void ClearPins()
        {
            Pins.Clear();
        }

        private void SetPin()
        {
            if (Pin is not null)

```

```

        {
            Pins.Add(Pin);
        }
    }

    #endregion
}

```

EDetailPage.cs

```

namespace TeamConnect.Enums
{
    public enum EDetailPage
    {
        Team,
        Leave,
        TeamTime,
    }
}

```

EPageState.cs

```

namespace TeamConnect.Enums
{
    public enum EPageState
    {
        Loading,
        Complete,
        NoResult,
        TeamNotCreated,
    }
}

```

LeaveExtension.cs

```

namespace TeamConnect.Extensions
{
    public static class LeaveExtension
    {
        public static LeaveViewModel ToViewModel(this LeaveModel leave,
UserModel user)
        {
            return new LeaveViewModel
            {
                Id = leave.Id,
                UserPhoto = user.Photo,
                UserName = user.Name,
                UserSurname = user.Surname,
                Type = leave.Type,
                StartDate = leave.StartDate,
                EndDate = leave.EndDate,
                UserId = leave.UserId,
            };
        }

        public static LeaveModel ToModel(this LeaveViewModel viewModel)
        {

```

```

        return new LeaveModel
        {
            Id = viewModel.Id,
            Type = viewModel.Type,
            StartDate = viewModel.StartDate,
            EndDate = viewModel.EndDate,
            UserId = viewModel.UserId,
        };
    }
}

```

TeamExtension.cs

```

namespace TeamConnect.Extensions
{
    public static class TeamExtension
    {
        public static TeamViewModel ToViewModel(this TeamModel model)
        {
            return new TeamViewModel
            {
                Id = model.Id,
                Name = model.Name,
            };
        }

        public static TeamModel ToModel(this TeamViewModel viewModel)
        {
            return new TeamModel
            {
                Id = viewModel.Id,
                Name = viewModel.Name,
            };
        }
    }
}

```

TimeZoneExtension.cs

```

namespace TeamConnect.Extensions
{
    public static class TimeZoneExtension
    {
        public static TimeZoneModel ToModel(this TimeZoneModelDTO modelDTO)
        {
            return new TimeZoneModel
            {
                DstOffset = modelDTO.DstOffset,
                RawOffset = modelDTO.RawOffset,
                Status = modelDTO.Status,
                TimeZoneID = modelDTO.TimeZoneID,
                TimeZoneName = modelDTO.TimeZoneName,
            };
        }
    }
}

```

UserExtension.cs

```

namespace TeamConnect.Extensions
{
    public static class UserExtension
    {
        #region -- Public helpers --

        public static UserViewModel ToViewModel(this UserModel model)
        {
            return new UserViewModel
            {
                Id = model.Id,
                Photo = model.Photo,
                Name = model.Name,
                Surname = model.Surname,
                Email = model.Email,
                Password = model.Password,
                Latitude = model.Latitude,
                Longitude = model.Longitude,
                Address = model.Address,
                TimeZoneId = model.TimeZoneId,
                CountryCode = model.CountryCode,
                TeamId = model.TeamId,
                Position = model.Position,
                StartWorkTime =
!string.IsNullOrEmpty(model.StartWorkTime)
?
ConvertTimeByTimeZoneId(TimeSpan.Parse(model.StartWorkTime), model.TimeZoneId,
TimeZoneInfo.Local.Id)
: DateTime.Now.Date,
                EndWorkTime = !string.IsNullOrEmpty(model.EndWorkTime)
? ConvertTimeByTimeZoneId(TimeSpan.Parse(model.EndWorkTime),
model.TimeZoneId, TimeZoneInfo.Local.Id)
: DateTime.Now.Date,
                IsAccountCreated = model.IsAccountCreated,
            };
        }

        public static UserModel ToModel(this UserViewModel viewModel)
        {
            return new UserModel
            {
                Id = viewModel.Id,
                Photo = viewModel.Photo,
                Name = viewModel.Name,
                Surname = viewModel.Surname,
                Email = viewModel.Email,
                Password = viewModel.Password,
                Latitude = viewModel.Latitude,
                Longitude = viewModel.Longitude,
                Address = viewModel.Address,
                TimeZoneId = viewModel.TimeZoneId,
                CountryCode = viewModel.CountryCode,
                TeamId = viewModel.TeamId,
                Position = viewModel.Position,
                StartWorkTime =
ConvertTimeByTimeZoneId(viewModel.StartWorkTime.TimeOfDay,
viewModel.TimeZoneId).TimeOfDay.ToString(),

```

```

        EndWorkTime
ConvertTimeByTimeZoneId(viewModel.EndWorkTime.TimeOfDay,
viewModel.TimeZoneId).TimeOfDay.ToString(),
        IsAccountCreated = viewModel.IsAccountCreated,
    };
}

public static UserModel ToModelWithoutTimeConverting(this
UserViewModel viewModel)
{
    return new UserModel
    {
        Id = viewModel.Id,
        Photo = viewModel.Photo,
        Name = viewModel.Name,
        Surname = viewModel.Surname,
        Email = viewModel.Email,
        Password = viewModel.Password,
        Latitude = viewModel.Latitude,
        Longitude = viewModel.Longitude,
        Address = viewModel.Address,
        TimeZoneId = viewModel.TimeZoneId,
        CountryCode = viewModel.CountryCode,
        TeamId = viewModel.TeamId,
        Position = viewModel.Position,
        StartWorkTime = viewModel.StartWorkTime.TimeOfDay.ToString(),
        EndWorkTime = viewModel.EndWorkTime.TimeOfDay.ToString(),
        IsAccountCreated = viewModel.IsAccountCreated,
    };
}

#endregion

#region -- Private helpers --

private static DateTime ConvertTimeByTimeZoneId(TimeSpan time, string
sourceTimeZoneId, string destinationTimeZoneId)
{
    var result = DateTime.Now.Date + time;

    if (!string.IsNullOrEmpty(sourceTimeZoneId)
        && !string.IsNullOrEmpty(destinationTimeZoneId))
    {
        var dateTime = DateTime.SpecifyKind(result,
DateTimeKind.Unspecified);
        var convertedDateTime =
TimeZoneInfo.ConvertTimeBySystemTimeZoneId(dateTime,
sourceTimeZoneId,
destinationTimeZoneId);
        result = DateTime.SpecifyKind(convertedDateTime,
DateTimeKind.Local);
    }

    return result;
}

private static DateTime ConvertTimeByTimeZoneId(TimeSpan time, string
timeZoneId)
{
    return !string.IsNullOrEmpty(timeZoneId)
        ?
TimeZoneInfo.ConvertTimeBySystemTimeZoneId(DateTime.Now.Date + time, timeZoneId)
        : DateTime.Now.Date + time;
}

```

```

    }

    #endregion
}

```

OperationResult.cs

```

namespace TeamConnect.Helpers
{
    public class OperationResult
    {
        #region -- Public properties --

        public bool IsSuccess { get; private set; }

        public string Message { get; private set; }

        public string ErrorName { get; private set; }

        public Exception Exception { get; private set; }

        #endregion
        #region -- Public helpers --

        public void SetSuccess()
        {
            SetResult(true, null, null, null);
        }

        public void SetFailure()
        {
            SetResult(false, null, null, null);
        }

        public void SetFailure(string message)
        {
            SetResult(false, null, message, null);
        }

        public void SetError(string errorName, string message, Exception
exception = null)
        {
            SetResult(false, errorName, message, exception);
        }

        #endregion

        #region -- Private helpers --

        private void SetResult(bool isSuccess, string errorName, string
message, Exception exception)
        {
            IsSuccess = isSuccess;
            ErrorName = errorName;
            Message = message;
            Exception = exception;
        }

        #endregion
    }
}

```

```

public class OperationResult<T> : OperationResult
{
    public T Result { get; private set; }

    #region -- Public properties --

    public void SetSuccess(T result)
    {
        Result = result;
        SetSuccess();
    }

    public void SetFailure(T result)
    {
        Result = result;
        SetFailure();
    }

    #endregion
}

```

ServerResponse.cs

```

namespace TeamConnect.Helpers
{
    public class ServerResponse<TSuccess, TError>
    {
        public TSuccess SuccessResult { get; set; }

        public TError ErrorResult { get; set; }

        public bool IsSuccess { get; set; }

        public int Code { get; set; }
    }
}

```

Validator.cs

```

namespace TeamConnect.Helpers
{
    public static class Validator
    {
        public static bool CheckIsEmailValid(string email)
        {
            var regex = new Regex(@"^([\w\.\-]+)@([\w\-\-
]+)((\.\w){2,3})+$");

            return regex.IsMatch(email);
        }

        public static bool CheckIsPasswordValid(string password)
        {
            var regex = new Regex(@"^[A-Z](?=.*[a-z])(?=.*\d)[a-zA-
Z\d]{5,15}$"); ;

            return regex.IsMatch(password);
        }
    }
}

```

```
    }
}
```

IEntityBase.cs

```
namespace TeamConnect.Models
{
    public interface IEntityBase
    {
        public int Id { get; set; }
    }
}
```

LeaveGroupViewModel.cs

```
namespace TeamConnect.Models.Leave
{
    public class LeaveGroupViewModel : ObservableCollection<LeaveViewModel>
    {
        public LeaveGroupViewModel(
            DateTime date)
            : base()
        {
            Date = date;
        }

        #region -- Public properties --

        private DateTime _date;
        public DateTime Date
        {
            get => _date;
            set => SetProperty(ref _date, value);
        }

        #endregion

        #region -- Private helpers --

        private void SetProperty<T>(ref T field, T newValue,
[CallerMemberName] string propertyName = null)
        {
            if (!EqualityComparer<T>.Default.Equals(field, newValue))
            {
                field = newValue;
                OnPropertyChanged(new
PropertyChangedEventArgs(propertyName));
            }
        }

        #endregion
    }
}
```

LeaveModel.cs

```
namespace TeamConnect.Models.Leave
{
```

```

public class LeaveModel : IEntityBase
{
    [PrimaryKey]
    [AutoIncrement]
    public int Id { get; set; }

    public string Type { get; set; }

    public DateTime StartDate { get; set; }

    public DateTime EndDate { get; set; }

    public int UserId { get; set; }
}

```

LeaveViewModel.cs

```

namespace TeamConnect.Models.Leave
{
    public class LeaveViewModel : BindableBase
    {
        #region -- Public properties --

        private int _id;
        public int Id
        {
            get => _id;
            set => SetProperty(ref _id, value);
        }

        private string _userPhoto;
        public string UserPhoto
        {
            get => _userPhoto;
            set => SetProperty(ref _userPhoto, value);
        }

        private string _userName;
        public string UserName
        {
            get => _userName;
            set => SetProperty(ref _userName, value);
        }

        private string _userSurname;
        public string UserSurname
        {
            get => _userSurname;
            set => SetProperty(ref _userSurname, value);
        }

        private string _type;
        public string Type
        {
            get => _type;
            set => SetProperty(ref _type, value);
        }

        private DateTime _startDate;
        public DateTime StartDate

```

```

        {
            get => _startDate;
            set => SetProperty(ref _startDate, value);
        }

        private DateTime _endDate;
        public DateTime EndDate
        {
            get => _endDate;
            set => SetProperty(ref _endDate, value);
        }

        private int _userId;
        public int UserId
        {
            get => _userId;
            set => SetProperty(ref _userId, value);
        }

        #endregion
    }
}

```

TeamModel.cs

```

namespace TeamConnect.Models.Team
{
    public class TeamModel : IEntityBase
    {
        [PrimaryKey]
        [AutoIncrement]
        public int Id { get; set; }

        public string Name { get; set; }
    }
}

```

TeamViewModel.cs

```

namespace TeamConnect.Models.Team
{
    public class TeamViewModel : BindableBase
    {
        #region -- Public properties --

        private int _id;
        public int Id
        {
            get => _id;
            set => SetProperty(ref _id, value);
        }

        private string _name;
        public string Name
        {
            get => _name;
            set => SetProperty(ref _name, value);
        }
    }
}

```

```

        #endregion
    }
}

```

TimeZoneModel.cs

```

namespace TeamConnect.Models.TimeZone
{
    public class TimeZoneModel
    {
        public int DstOffset { get; set; }

        public int RawOffset { get; set; }

        public string Status { get; set; }

        public string TimeZoneID { get; set; }

        public string TimeZoneName { get; set; }
    }
}

```

TimeZoneModelDTO.cs

```

namespace TeamConnect.Models.TimeZone
{
    [Serializable()]
    public class TimeZoneModelDTO
    {
        [JsonProperty("dstOffset")]
        public int DstOffset { get; set; }

        [JsonProperty("rawOffset")]
        public int RawOffset { get; set; }

        [JsonProperty("status")]
        public string Status { get; set; }

        [JsonProperty("timeZoneId")]
        public string TimeZoneID { get; set; }

        [JsonProperty("timeZoneName")]
        public string TimeZoneName { get; set; }
    }
}

```

UserGroupViewModel.cs

```

namespace TeamConnect.Models.User
{
    public class UserGroupViewModel : ObservableCollection<UserViewModel>
    {
        public UserGroupViewModel(
            DateTime startWorkTime,
            DateTime endWorkTime)
        {
            StartWorkTime = startWorkTime;
            EndWorkTime = endWorkTime;
        }
    }
}

```

```

    }

    #region -- Public properties --
    private DateTime _endWorkTime;
    public DateTime EndWorkTime
    {
        get => _endWorkTime;
        set => SetProperty(ref _endWorkTime, value);
    }

    private DateTime _startWorkTime;
    public DateTime StartWorkTime
    {
        get => _startWorkTime;
        set => SetProperty(ref _startWorkTime, value);
    }

    #endregion

    #region -- Private helpers --

    private void SetProperty<T>(ref T field, T newValue,
[CallerMemberName] string propertyName = null)
    {
        if (!EqualityComparer<T>.Default.Equals(field, newValue))
        {
            field = newValue;
            OnPropertyChanged(new
PropertyChangedEventArgs(propertyName));
        }
    }

    #endregion
}
}

```

UserModel.cs

```

namespace TeamConnect.Models.User
{
    public class UserModel : IEntityBase
    {
        [PrimaryKey]
        [AutoIncrement]
        public int Id { get; set; }

        public string Photo { get; set; }

        public string Name { get; set; }

        public string Surname { get; set; }

        public string Email { get; set; }

        public string Password { get; set; }

        public double Latitude { get; set; }

        public double Longitude { get; set; }

        public string Address { get; set; }
    }
}

```

```

        public string TimeZoneId { get; set; }

        public string CountryCode { get; set; }

        public int TeamId { get; set; }

        public string Position { get; set; }

        public string StartWorkTime { get; set; }

        public string EndWorkTime { get; set; }

        public bool IsAccountCreated { get; set; }
    }
}

```

UserViewModel.cs

```

namespace TeamConnect.Models.User
{
    public class UserViewModel : BindableBase
    {
        #region -- Public properties --

        private int _id;
        public int Id
        {
            get => _id;
            set => SetProperty(ref _id, value);
        }

        private string _photo;
        public string Photo
        {
            get => _photo;
            set => SetProperty(ref _photo, value);
        }

        private string _name;
        public string Name
        {
            get => _name;
            set => SetProperty(ref _name, value);
        }

        private string _surname;
        public string Surname
        {
            get => _surname;
            set => SetProperty(ref _surname, value);
        }

        private string _email;
        public string Email
        {
            get => _email;
            set => SetProperty(ref _email, value);
        }

        private string _password;
        public string Password

```

```

{
    get => _password;
    set => SetProperty(ref _password, value);
}
private double _latitude;
public double Latitude
{
    get => _latitude;
    set => SetProperty(ref _latitude, value);
}

private double _longitude;
public double Longitude
{
    get => _longitude;
    set => SetProperty(ref _longitude, value);
}

private string _address;
public string Address
{
    get => _address;
    set => SetProperty(ref _address, value);
}

private string _timeZoneId;
public string TimeZoneId
{
    get => _timeZoneId;
    set => SetProperty(ref _timeZoneId, value);
}

private string _countryCode;
public string CountryCode
{
    get => _countryCode;
    set => SetProperty(ref _countryCode, value);
}

private int _teamId;
public int TeamId
{
    get => _teamId;
    set => SetProperty(ref _teamId, value);
}

private string _position;
public string Position
{
    get => _position;
    set => SetProperty(ref _position, value);
}

private DateTime _startWorkTime;
public DateTime StartWorkTime
{
    get => _startWorkTime;
    set => SetProperty(ref _startWorkTime, value);
}

private DateTime _endWorkTime;
public DateTime EndWorkTime

```

```

    {
        get => _endWorkTime;
        set => SetProperty(ref _endWorkTime, value);
    }

    private bool _isAccountCreated;
    public bool IsAccountCreated
    {
        get => _isAccountCreated;
        set => SetProperty(ref _isAccountCreated, value);
    }

    private ICommand _tapCommand;
    public ICommand TapCommand
    {
        get => _tapCommand;
        set => SetProperty(ref _tapCommand, value);
    }

    #endregion
}

```

DefaultResources.xaml.cs

```

<?xml version="1.0" encoding="UTF-8"?>
<ResourceDictionary xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:controls="clr-namespace:TeamConnect.Controls"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    x:Class="TeamConnect.Resources.Styles.DefaultResources">

    <!-- TEXT SIZE -->
    <x:Double x:Key="tsize_i20">22</x:Double> <!-- 20px -->
    <x:Double x:Key="tsize_i16">18</x:Double> <!-- 16px -->
    <x:Double x:Key="tsize_i14">16</x:Double> <!-- 14px -->
    <x:Double x:Key="tsize_i10">12</x:Double> <!-- 10px -->

    <!-- NAVIGATION PAGE STYLE -->
    <Style TargetType="NavigationPage">
        <Setter Property="BarBackgroundColor" Value="{DynamicResource
appcolor_i4}"/>
        <Setter Property="BarTextColor" Value="{DynamicResource tcolor_i1}"/>
    </Style>

    <!-- TITLE STYLES -->
    <Style x:Key="BaseTitle" TargetType="Label">
        <Setter Property="FontFamily" Value="SemiBoldFont"/>
    </Style>

    <Style x:Key="Title_20px" TargetType="Label" BasedOn="{StaticResource
BaseTitle}"> <!-- Title 20px -->
        <Setter Property="FontSize" Value="{DynamicResource tsize_i20}"/>
    </Style>

    <Style x:Key="Title_16px" TargetType="Label" BasedOn="{StaticResource
BaseTitle}"> <!-- Title 16px -->
        <Setter Property="FontSize" Value="{DynamicResource tsize_i16}"/>
    </Style>

```

```

        <Style x:Key="Title_14px" TargetType="Label" BasedOn="{StaticResource
BaseTitle}"> <!-- Title 14px -->
            <Setter Property="FontSize" Value="{DynamicResource tsize_i14}"/>
        </Style>
        <Style x:Key="tstyle_i1_1" TargetType="Label" BasedOn="{StaticResource
Title_14px}"> <!-- Title 14px LightSlateGray -->
            <Setter Property="TextColor" Value="{DynamicResource tcolor_i5}"/>
        </Style>

        <Style x:Key="tstyle_i1_2" TargetType="Label" BasedOn="{StaticResource
Title_16px}"> <!-- Title 16px Black -->
            <Setter Property="TextColor" Value="{DynamicResource tcolor_i1}"/>
        </Style>

        <Style x:Key="tstyle_i1_3" TargetType="Label" BasedOn="{StaticResource
Title_20px}"> <!-- Title 20px Black -->
            <Setter Property="TextColor" Value="{DynamicResource tcolor_i1}"/>
        </Style>

        <!-- CONTENT STYLES -->
        <Style x:Key="BaseContentRegular" TargetType="Label">
            <Setter Property="FontFamily" Value="RegularFont"/>
        </Style>

        <Style
            x:Key="ContentRegular_14px"
            TargetType="Label"
BasedOn="{StaticResource BaseContentRegular}"> <!-- Content 14px-->
            <Setter Property="FontSize" Value="{DynamicResource tsize_i14}"/>
        </Style>

        <Style x:Key="tstyle_i2_1" TargetType="Label" BasedOn="{StaticResource
ContentRegular_14px}"> <!-- Content 14px Gray -->
            <Setter Property="TextColor" Value="{DynamicResource tcolor_i2}"/>
        </Style>

        <Style x:Key="tstyle_i2_2" TargetType="Label" BasedOn="{StaticResource
ContentRegular_14px}"> <!-- Content 14px Black -->
            <Setter Property="TextColor" Value="{DynamicResource tcolor_i1}"/>
        </Style>

        <Style x:Key="BaseContentMedium" TargetType="Label">
            <Setter Property="FontFamily" Value="MediumFont"/>
        </Style>

        <Style
            x:Key="ContentMedium_14px"
            TargetType="Label"
BasedOn="{StaticResource BaseContentMedium}"> <!-- Content 14px-->
            <Setter Property="FontSize" Value="{DynamicResource tsize_i14}"/>
        </Style>

        <Style x:Key="tstyle_i3_1" TargetType="Label" BasedOn="{StaticResource
ContentMedium_14px}"> <!-- Content 14px Gray -->
            <Setter Property="TextColor" Value="{DynamicResource tcolor_i2}"/>
        </Style>

        <!-- ACTION STYLES -->
        <Style x:Key="BaseAction" TargetType="Label">
            <Setter Property="FontFamily" Value="MediumFont"/>
        </Style>

        <Style x:Key="Action_16px" TargetType="Label" BasedOn="{StaticResource
BaseAction}"> <!-- Action 16px-->
            <Setter Property="FontSize" Value="{DynamicResource tsize_i16}"/>
        </Style>

```

```

        <Style x:Key="tstyle_i4_1" TargetType="Label" BasedOn="{StaticResource
Action_16px}"> <!-- Action 14px Black -->
        <Setter Property="TextColor" Value="{DynamicResource tcolor_i1}"/>
    </Style>
    <!-- ENTRY STYLES -->
    <Style x:Key="estyle_i1" TargetType="controls:CustomEntry">
        <Setter Property="BorderColor" Value="{DynamicResource
appcolor_i2}"/>
        <Setter Property="FocusedColor" Value="{DynamicResource
appcolor_i1}"/>
        <Setter Property="ErrorColor" Value="{DynamicResource appcolor_i3}"/>
        <Setter Property="FontFamily" Value="RegularFont"/>
        <Setter Property="FontSize" Value="{DynamicResource tsize_i14}"/>
        <Setter Property="TextColor" Value="{DynamicResource tcolor_i1}"/>
        <Setter Property="PlaceholderColor" Value="{DynamicResource
tcolor_i2}"/>
        <Setter Property="ErrorFontFamily" Value="RegularFont"/>
        <Setter Property="ErrorFontSize" Value="{DynamicResource
tsize_i10}"/>
    </Style>

    <!-- BUTTON STYLES -->
    <Style x:Key="bstyle_i1" TargetType="Button">
        <Setter Property="BackgroundColor" Value="{DynamicResource
appcolor_i1}"/>
        <Setter Property="TextColor" Value="{DynamicResource tcolor_i3}"/>
        <Setter Property="FontFamily" Value="MediumFont"/>
        <Setter Property="FontSize" Value="{DynamicResource tsize_i14}"/>
        <Setter Property="TextTransform" Value="None"/>
        <Setter Property="HeightRequest" Value="56"/>
        <Setter Property="CornerRadius" Value="16"/>
    </Style>
    <Style x:Key="bstyle_i2" TargetType="Button">
        <Setter Property="VisualStateManager.VisualStateGroups">
            <VisualStateGroupList>
                <VisualStateGroup x:Name="CommonStates">
                    <VisualState x:Name="Normal">
                        <VisualState.Setters>
                            <Setter Property="BackgroundColor"
Value="{DynamicResource appcolor_i4}"/>
                            <Setter Property="TextColor"
Value="{DynamicResource tcolor_i4}"/>
                            <Setter Property="FontFamily"
Value="MediumFont"/>
                            <Setter Property="FontSize"
Value="{DynamicResource tsize_i14}"/>
                            <Setter Property="TextTransform" Value="None"/>
                            <Setter Property="HeightRequest" Value="56"/>
                            <Setter Property="CornerRadius" Value="16"/>
                            <Setter Property="BorderWidth" Value="1"/>
                            <Setter Property="BorderColor"
Value="{DynamicResource appcolor_i1}"/>
                        </VisualState.Setters>
                    </VisualState>
                    <VisualState x:Name="Pressed">
                        <VisualState.Setters>
                            <Setter Property="BackgroundColor"
Value="{DynamicResource appcolor_i5}"/>
                            <Setter Property="TextColor"
Value="{DynamicResource tcolor_i4}"/>
                            <Setter Property="FontFamily"
Value="MediumFont"/>

```

```

        <Setter Property="FontSize"
Value="{DynamicResource tsize_i14}"/>
        <Setter Property="TextTransform" Value="None"/>
        <Setter Property="HeightRequest" Value="56"/>
        <Setter Property="CornerRadius" Value="16"/>
        <Setter Property="BorderWidth" Value="1"/>
        <Setter Property="BorderColor"
Value="{DynamicResource appcolor_i1}"/>
    </VisualState.Setters>
</VisualState>
</VisualStateGroup>
</VisualStateGroupList>
</Setter>
</Style>

<!-- DISABLED BUTTON STYLES -->
<Style x:Key="bstyle_i3" TargetType="Button">
    <Setter Property="BackgroundColor" Value="{DynamicResource
appcolor_i6}"/>
    <Setter Property="TextColor" Value="{DynamicResource tcolor_i3}"/>
    <Setter Property="FontFamily" Value="MediumFont"/>
    <Setter Property="FontSize" Value="{DynamicResource tsize_i14}"/>
    <Setter Property="TextTransform" Value="None"/>
    <Setter Property="HeightRequest" Value="56"/>
    <Setter Property="CornerRadius" Value="16"/>
</Style>

<Style x:Key="activityIndicator" TargetType="ActivityIndicator">
    <Setter Property="Color" Value="{DynamicResource appcolor_i1}"/>
    <Setter Property="IsRunning" Value="True"/>
    <Setter Property="HorizontalOptions" Value="Center"/>
    <Setter Property="VerticalOptions" Value="Center"/>
</Style>

<xct:IsNotNullOrEmptyConverter x:Key="isNotNullOrEmptyConverter"/>
<xct:IsNullOrEmptyConverter x:Key="isNullOrEmptyConverter"/>
<xct:ListIsNotNullOrEmptyConverter
x:Key="listIsNotNullOrEmptyConverter"/>

</ResourceDictionary>

```

LightModeResources.xaml.cs

```

<?xml version="1.0" encoding="UTF-8"?>
<ResourceDictionary xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="TeamConnect.Resources.Styles.
LightModeResources">

    <!-- APP COLOR -->
    <Color x:Key="appcolor_i1">Blue</Color>
    <Color x:Key="appcolor_i2">LightGray</Color>
    <Color x:Key="appcolor_i3">Red</Color>
    <Color x:Key="appcolor_i4">White</Color>
    <Color x:Key="appcolor_i5">#E5E5ff</Color>
    <Color x:Key="appcolor_i6">#F2F2FF</Color>

    <!-- TEXT COLORS -->
    <Color x:Key="tcolor_i1">Black</Color>
    <Color x:Key="tcolor_i2">Gray</Color>
    <Color x:Key="tcolor_i3">White</Color>

```

```

<Color x:Key="tcolor_i4">Blue</Color>
<Color x:Key="tcolor_i5">LightSlateGray</Color>

<!-- IMAGES -->
<ImageSource x:Key="ic_eye">ic_eye</ImageSource>
<ImageSource x:Key="ic_eye_off">ic_eye_off</ImageSource>
<ImageSource x:Key="ic_clock_blue">ic_clock_blue</ImageSource>
<ImageSource x:Key="ic_clock_gray">ic_clock_gray</ImageSource>
<ImageSource x:Key="ic_plane_blue">ic_plane_blue</ImageSource>
<ImageSource x:Key="ic_plane_gray">ic_plane_gray</ImageSource>
<ImageSource x:Key="ic_team_blue">ic_team_blue</ImageSource>
<ImageSource x:Key="ic_team_gray">ic_team_gray</ImageSource>
<ImageSource x:Key="pic_profile">pic_profile</ImageSource>
<ImageSource x:Key="ic_logout">ic_logout</ImageSource>
<ImageSource x:Key="ic_calendar">ic_calendar</ImageSource>
<ImageSource x:Key="ic_reset">ic_reset</ImageSource>
<ImageSource x:Key="ic_plus">ic_plus</ImageSource>
<ImageSource x:Key="ic_close">ic_close</ImageSource>

</ResourceDictionary>

```

IAuthorizationService.cs

```

namespace TeamConnect.Services.AuthorizationService
{
    public interface IAuthorizationService
    {
        public bool IsAuthorized { get; }

        Task<OperationResult> CheckIsEmailExistAsync(string email);

        Task<OperationResult<UserModel>> LogInAsync(string email, string
password);

        Task<OperationResult> CompleteRegistration(UserModel user);

        Task<OperationResult> SignUpAsync(UserModel user);

        void LogOut();
    }
}

```

AuthorizationService.cs

```

namespace TeamConnect.Services.AuthorizationService
{
    public class AuthorizationService : IAuthorizationService
    {
        private readonly IRepository _repository;
        private readonly ISettingsManager _settingsManager;

        public AuthorizationService(
            IRepository repository,
            ISettingsManager settingsManager)
        {
            _repository = repository;
            _settingsManager = settingsManager;
        }
    }
}

```

```

#region -- IUserService implementation --

public bool IsAuthorized => _settingsManager.UserId > 0;

public async Task<OperationResult> CheckIsEmailExistAsync(string
email)
{
    var result = new OperationResult();

    try
    {
        var user = await _repository.FindAsync<UserModel>(u =>
u.Email == email);

        if (user != null)
        {
            result.SetSuccess();
        }
        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(CheckIsEmailExistAsync)}
exception", "Something went wrong", ex);
    }
    return result;
}

public async Task<OperationResult<UserModel>> LogInAsync(string
email, string password)
{
    var result = new OperationResult<UserModel>();

    try
    {
        var user = await _repository.FindAsync<UserModel>(u =>
u.Email == email.ToLower() && u.Password == password);

        if (user is not null)
        {
            if (user.IsAccountCreated)
            {
                _settingsManager.UserId = user.Id;
                _settingsManager.TeamId = user.TeamId;
            }
            result.SetSuccess(user);
        }
        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(LogInAsync)}
"Something went wrong", ex);
    }

    return result;
}

```

```

    }

    public async Task<OperationResult> CompleteRegistration(UserModel
user)
    {
        var result = new OperationResult();

        try
        {
            if (user.Id != Constants.DEFAULT_ID)
            {
                await _repository.UpdateAsync(user);

                _settingsManager.UserId = user.Id;

                result.SetSuccess();
            }
            else
            {
                result.SetFailure();
            }
        }
        catch (Exception ex)
        {
            result.SetError($"{nameof(CompleteRegistration)}
exception", "Something went wrong", ex);

            return result;
        }
    }
    public async Task<OperationResult> SignUpAsync(UserModel user)
    {
        var result = new OperationResult();

        try
        {
            if (user is not null)
            {
                await _repository.InsertAsync(user);

                result.SetSuccess();
            }
            else
            {
                result.SetFailure();
            }
        }
        catch (Exception ex)
        {
            result.SetError($"{nameof(SignUpAsync)}
"Something went wrong", ex);

            return result;
        }
    }
    public void Logout()
    {
        _settingsManager.ClearSettings();
    }
    #endregion
}
}

```

ILeaveService.cs

```
namespace TeamConnect.Services.LeaveService
{
    public interface ILeaveService
    {
        Task<OperationResult<IEnumerable<LeaveModel>>>
        GetLeavesByDatesAsync(DateTime startDate, DateTime endDate);

        Task<OperationResult> AddLeaveAsync(LeaveModel newLeave);
    }
}
```

LeaveService.cs

```
namespace TeamConnect.Services.LeaveService
{
    public class LeaveService : ILeaveService
    {
        private readonly IRepository _repository;
        private readonly ISettingsManager _settingsManager;

        public LeaveService(
            IRepository repository,
            ISettingsManager settingsManager)
        {
            _repository = repository;
            _settingsManager = settingsManager;
        }

        #region -- ILeaveService implementation --

        public async Task<OperationResult<IEnumerable<LeaveModel>>>
        GetLeavesByDatesAsync(DateTime startDate, DateTime endDate)
        {
            var result = new OperationResult<IEnumerable<LeaveModel>>();
            try
            {
                var leaves = await _repository.GetAllAsync<LeaveModel>(
                    l => l.StartDate >= startDate
                        && l.StartDate <= endDate
                        || l.EndDate >= startDate
                        && l.EndDate <= endDate);

                if (leaves is not null && leaves.Count > 0)
                {
                    result.SetSuccess(leaves);
                }
                else
                {
                    result.SetFailure();
                }
            }
            catch (Exception ex)
            {
                result.SetError($"{nameof(GetLeavesByDatesAsync)}
                exception", "Something went wrong", ex);
            }

            return result;
        }
    }
}
```

```

    }

    public async Task<OperationResult> AddLeaveAsync(LeaveModel newLeave)
    {
        var result = new OperationResult();

        try
        {
            var leave = await _repository.FindAsync<LeaveModel>(
                l => l.UserId == _settingsManager.UserId
                && l.StartDate <= newLeave.StartDate
                && l.EndDate >= newLeave.StartDate
                || l.UserId == _settingsManager.UserId
                && l.StartDate <= newLeave.EndDate
                && l.EndDate >= newLeave.EndDate);

            if (leave is null)
            {
                newLeave.UserId = _settingsManager.UserId;

                await _repository.InsertAsync(newLeave);

                result.SetSuccess();
            }
            else
            {
                result.SetFailure(Strings.OverlappingLeavesError);
            }
        }
        catch (Exception ex)
        {
            result.SetError($"{nameof(GetLeavesByDatesAsync)}
exception", "Something went wrong", ex);
        }

        return result;
    }

    #endregion
}

```

IMapService.cs

```

namespace TeamConnect.Services.MapService
{
    public interface IMapService
    {
        Task<OperationResult<Placemark>> GetPlacemarkAsync(double latitude,
double longitude);

        Task<OperationResult<TimeZoneModel>> GetTimeZoneAsync(double
latitude, double longitude);
    }
}

```

MapService.cs

```

namespace TeamConnect.Services.MapService

```

```

{
    public class MapService : IMapService
    {
        private readonly IRestService _restService;

        public MapService(IRestService restService)
        {
            _restService = restService;
        }

        #region -- IMapService implementation --

        public async Task<OperationResult<Placemark>>
        GetPlacemarkAsync(double latitude, double longitude)
        {
            var result = new OperationResult<Placemark>();

            try
            {
                var placemarks = await Geocoding.GetPlacemarksAsync(latitude,
longitude);

                if (placemarks.ToList().Count > 0)
                {
                    result.SetSuccess(placemarks?.FirstOrDefault());
                }
                else
                {
                    result.SetFailure();
                }
            }
            catch (Exception ex)
            {
                result.SetError($"{nameof(GetPlacemarkAsync)}:    exception",
"Something went wrong", ex);
            }

            return result;
        }

        public async Task<OperationResult<TimeZoneModel>>
        GetTimeZoneAsync(double latitude, double longitude)
        {
            var result = new OperationResult<TimeZoneModel>();

            try
            {
                var offset = new DateTimeOffset(DateTime.Now);

                var response = await _restService.GetAsync<TimeZoneModelDTO,
object>(
                    $"{Constants.GoogleAPI.BASE_TIMEZONE_API_URI}" +
                    $"{Constants.GoogleAPI.LOCATION}{latitude.ToString(CultureInfo.InvariantCulture)},{longitude.ToString(CultureInfo.InvariantCulture)}" +
                    $"{Constants.GoogleAPI.TIMESTAMP}{offset.ToUnixTimeSeconds()}" +
                    $"{Constants.GoogleAPI.KEY}{Constants.GoogleAPI.API_KEY}");

                if (response.IsSuccess)
                {
                    result.SetSuccess(response.SuccessResult.ToModel());
                }
            }
        }
    }
}

```

```

        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(GetTimeZoneAsync)}: exception",
"Something went wrong", ex);
    }

    return result;
}

#endregion
}

```

IRepository.cs

```

namespace TeamConnect.Services.Repository
{
    public interface IRepository
    {
        Task<int> DeleteAsync<T>(T entity) where T : IEntityBase, new();

        Task<List<T>> GetAllAsync<T>(Expression<Func<T, bool>> predicate =
null) where T : IEntityBase, new();

        Task<int> InsertAsync<T>(T entity) where T : IEntityBase, new();

        Task<int> UpdateAsync<T>(T entity) where T : IEntityBase, new();

        Task<T> FindAsync<T>(Expression<Func<T, bool>> predicate = null)
where T : IEntityBase, new();
    }
}

```

Repository.cs

```

namespace TeamConnect.Services.Repository
{
    public class Repository : IRepository
    {
        private Lazy<SQLiteAsyncConnection> _database;

        public Repository()
        {
            _database = new Lazy<SQLiteAsyncConnection>(() =>
            {
                var path =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData
), Constants.DATABASE_NAME);
                var database = new SQLiteAsyncConnection(path);

                database.CreateTableAsync<UserModel>().Wait();
                database.CreateTableAsync<TeamModel>().Wait();
                database.CreateTableAsync<LeaveModel>().Wait();
            });
        }
    }
}

```

```

        return database;
    });
}

public Task<int> DeleteAsync<T>(T entity) where T : IEntityTypeBase,
new()
{
    return _database.Value.DeleteAsync(entity);
}

public Task<List<T>> GetAllAsync<T>(Expression<Func<T, bool>>
predicate = null) where T : IEntityTypeBase, new()
{
    return predicate is null
        ? _database.Value.Table<T>().ToListAsync()
        : _database.Value.Table<T>().Where(predicate).ToListAsync();
}

public Task<int> InsertAsync<T>(T entity) where T : IEntityTypeBase,
new()
{
    return _database.Value.InsertAsync(entity);
}

public Task<int> UpdateAsync<T>(T entity) where T : IEntityTypeBase,
new()
{
    return _database.Value.UpdateAsync(entity);
}

public Task<T> FindAsync<T>(Expression<Func<T, bool>> predicate =
null) where T : IEntityTypeBase, new()
{
    return _database.Value.FindAsync(predicate);
}
}
}

```

IRestService.cs

```

namespace TeamConnect.Services.RestService
{
    public interface IRestService
    {
        /// <summary>
        /// Get request.
        /// </summary>
        /// <typeparam name="TSuccess">The object that is returned if the
request succeeds.</typeparam>
        /// <typeparam name="TError">The object that is returned if an
exception is thrown.</typeparam>
        /// <param name="resource">Request URL</param>
        /// <param name="additioalHeaders">Additioal headers
dictionary.</param>
        /// <returns>Returns the object that contains the data and results of
the request.</returns>
        Task<ServerResponse<TSuccess, TError>> GetAsync<TSuccess,
TError>(string resource, Dictionary<string, string> additioalHeaders = null);
    }
}

```

RestService.cs

```

namespace TeamConnect.Services.RestService
{
    public class RestService : IRestService
    {
        #region -- IRestService implementation --
        public Task<ServerResponse<TSuccess, TError>> GetAsync<TSuccess,
TError>(string resource, Dictionary<string, string> additionalHeaders = null)
        {
            var request = new HttpRequestMessage
            {
                Method = HttpMethod.Get,
                RequestUri = new Uri(resource),
            };

            return SendRequest<TSuccess, TError>(request, additionalHeaders);
        }

        #endregion

        #region -- Private helpers --

        private async Task<ServerResponse<TSuccess, TError>>
SendRequest<TSuccess, TError>(HttpRequestMessage request, Dictionary<string, string>
additionalHeaders = null)
        {
            var result = new ServerResponse<TSuccess, TError>();

            using (var client = CreateHttpClient(DefaultHeaders(additionalHeaders)))
            using (var response = await client.SendAsync(request).ConfigureAwait(false))
            {
                var responseString = await response.Content.ReadAsStringAsync();

                result.IsSuccess = response.IsSuccessStatusCode;
                result.Code = (int)response.StatusCode;

                if (response.IsSuccessStatusCode)
                {
                    result.SuccessResult =
JsonConvert.DeserializeObject<TSuccess>(responseString, converters: new
JsonConverter[] { new IsoDateTimeConverter() });
                }
                else
                {
                    result.ErrorResult =
JsonConvert.DeserializeObject<TError>(responseString, converters: new JsonConverter[]
{ new IsoDateTimeConverter() });
                }
            }

            return result;
        }

        private HttpClient CreateHttpClient(Dictionary<string, string>
headerParams = null)
        {
            var httpClient = new HttpClient();

```

```

        if (headerParams != null)
        {
            foreach (var headerParam in headerParams)
            {
                httpClient.DefaultRequestHeaders.Add(headerParam.Key,
headerParam.Value);
            }
        }

        return httpClient;
    }

    private Dictionary<string, string> DefaultHeaders(Dictionary<string,
string> additionalHeaders = null)
    {
        if (additionalHeaders == null)
        {
            additionalHeaders = new Dictionary<string, string>();
        }

        var defheaders = new Dictionary<string, string>();

        defheaders["User-Agent"] = "Mobile";
        defheaders["Accept"] = "*/*";

        foreach (var kv in additionalHeaders)
        {
            defheaders[kv.Key] = kv.Value;
        }

        return defheaders;
    }

    #endregion
}
}

```

ISettingsManager.cs

```

namespace TeamConnect.Services.SettingsManager
{
    public interface ISettingsManager
    {
        public int UserId { get; set; }

        public int TeamId { get; set; }

        void ClearSettings();
    }
}

```

SettingsManager.cs

```

namespace TeamConnect.Services.SettingsManager
{
    public class SettingsManager : ISettingsManager
    {
        #region -- ISettingsManager implementation --
    }
}

```

```

        public int UserId
        {
            get => Preferences.Get(nameof(UserId), default(int));
            set => Preferences.Set(nameof(UserId), value);
        }
        public int TeamId
        {
            get => Preferences.Get(nameof(TeamId), default(int));
            set => Preferences.Set(nameof(TeamId), value);
        }

        public void ClearSettings()
        {
            UserId = default(int);
            TeamId = default(int);
        }

        #endregion
    }
}

```

ITeamService.cs

```

namespace TeamConnect.Services.TeamService
{
    public interface ITeamService
    {
        public bool IsUserInTeam { get; }

        Task<OperationResult<TeamModel>> GetTeamAsync();

        Task<OperationResult> CreateTeamAsync(TeamModel team);

        Task<OperationResult> AddMemberAsync(UserModel user);
    }
}

```

TeamService.cs

```

namespace TeamConnect.Services.TeamService
{
    public class TeamService : ITeamService
    {
        private readonly IRepository _repository;
        private readonly ISettingsManager _settingsManager;
        private readonly IUserService _userService;
        public TeamService(
            IRepository repository,
            ISettingsManager settingsManager,
            IUserService userService)
        {
            _repository = repository;
            _settingsManager = settingsManager;
            _userService = userService;
        }
        #region -- ITeamService implementation --

        public bool IsUserInTeam => _settingsManager.TeamId > 0;
        public async Task<OperationResult<TeamModel>> GetTeamAsync()

```

```

    {
        var result = new OperationResult<TeamModel>();

        try
        {
            var team = await _repository.FindAsync<TeamModel>(t => t.Id
== _settingsManager.TeamId);

            if (team is not null)
            {
                result.SetSuccess(team);
            }
            else
            {
                result.SetFailure();
            }
        }
        catch (Exception ex)
        {
            result.SetError($"{nameof(GetTeamAsync)} : exception",
"Something went wrong", ex);
        }

        return result;
    }

    public async Task<OperationResult> CreateTeamAsync(TeamModel team)
    {
        var result = new OperationResult();
        try
        {
            await _repository.InsertAsync(team);

            var          getCurrentUserResult          =          await
_userService.GetCurrentUserAsync();

            if (getCurrentUserResult.IsSuccess)
            {
                var user = getCurrentUserResult.Result;

                user.TeamId = team.Id;

                var          updateUserResult          =          await
_repository.UpdateAsync(user);
                _settingsManager.TeamId = team.Id;

                result.SetSuccess();
            }
            else
            {
                result.SetFailure();
            }
        }
        catch (Exception ex)
        {
            result.SetError($"{nameof(CreateTeamAsync)} : exception",
"Something went wrong", ex);
        }

        return result;
    }

```

```

        public async Task<OperationResult>
AddMemberAsync(models.User.UserModel user)
    {
        var result = new OperationResult<TeamModel>();

        try
        {
            if (user.TeamId == 0)
            {
                user.TeamId = _settingsManager.TeamId;

                await _repository.UpdateAsync(user);
                result.SetSuccess();
            }
            else
            {
                result.SetFailure(Strings.ThisUserIsInAnotherTeam);
            }
        }
        catch (Exception ex)
        {
            result.SetError($"{nameof(AddMemberAsync)} : exception",
"Something went wrong", ex);
        }

        return result;
    }
}
#endregion
}

```

IUserService.cs

```

namespace TeamConnect.Services.UserService
{
    public interface IUserService
    {
        Task<OperationResult<IEnumerable<UserModel>>> GetTeamMembersAsync();

        Task<OperationResult<IEnumerable<UserModel>>>
GetNotMissingUsersAsync(DateTime date);

        Task<OperationResult<UserModel>> GetUserByIdAsync(int userId);

        Task<OperationResult<UserModel>> GetCurrentUserAsync();

        Task<OperationResult<IEnumerable<UserModel>>> SearchUsers(string
searchRequest);
    }
}

```

UserService.cs

```

namespace TeamConnect.Services.UserService
{
    public class UserService : IUserService
    {
        private readonly IRepository _repository;
        private readonly ISettingsManager _settingsManager;
    }
}

```

```

public UserService(
    IRepository repository,
    ISettingsManager settingsManager)
{
    _repository = repository;
    _settingsManager = settingsManager;
}

#region -- IUserService implementation --

public async Task<OperationResult<IEnumerable<UserModel>>>
GetTeamMembersAsync()
{
    var result = new OperationResult<IEnumerable<UserModel>>();

    try
    {
        var users = await _repository.GetAllAsync<UserModel>(u =>
u.TeamId == _settingsManager.TeamId);

        if (users is not null && users.Count > 0)
        {
            result.SetSuccess(users);
        }
        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(GetTeamMembersAsync)} : exception",
"Something went wrong", ex);
    }

    return result;
}

public async Task<OperationResult<IEnumerable<UserModel>>>
GetNotMissingUsersAsync(DateTime date)
{
    var result = new OperationResult<IEnumerable<UserModel>>();

    try
    {
        var getTeamMembersResult = await GetTeamMembersAsync();

        if (getTeamMembersResult.IsSuccess)
        {
            var users = getTeamMembersResult.Result;

            var notMissingUsers = new List<UserModel>();

            foreach (var item in users)
            {
                var leave = await _repository.FindAsync<LeaveModel>(
                    r => r.StartDate <= date
                        && r.EndDate >= date
                        && r.UserId == item.Id);

                if (leave is null)
                {

```

```

        var isHoliday = DateSystem.IsPublicHoliday(date,
item.CountryCode);
        var isWeekend = DateSystem.IsWeekend(date,
item.CountryCode);

        if (!isHoliday && !isWeekend)
        {
            notMissingUsers.Add(item);
        }
    }

    if (notMissingUsers.Count > 0)
    {
        result.SetSuccess(notMissingUsers);
    }
    else
    {
        result.SetFailure();
    }
}
else
{
    result.SetFailure();
}
}
catch (Exception ex)
{
    result.SetError($"{nameof(GetNotMissingUsersAsync)} :
exception", "Something went wrong", ex);
}

return result;
}

public async Task<OperationResult<UserModel>> GetUserByIdAsync(int
userId)
{
    var result = new OperationResult<UserModel>();

    try
    {
        var user = await _repository.FindAsync<UserModel>(u => u.Id
== userId);

        if (user is not null)
        {
            result.SetSuccess(user);
        }
        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(GetUserByIdAsync)} : exception",
"Something went wrong", ex);
    }

    return result;
}

```

```

public async Task<OperationResult<UserModel>> GetCurrentUserAsync()
{
    var result = new OperationResult<UserModel>();

    try
    {
        var getUserByIdResult = await
        GetUserByIdAsync(_settingsManager.UserId);

        if (getUserByIdResult.IsSuccess)
        {
            result.SetSuccess(getUserByIdResult.Result);
        }
        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(GetCurrentUserAsync)} : exception",
        "Something went wrong", ex);
    }

    return result;
}

public async Task<OperationResult<IEnumerable<UserModel>>>
SearchUsers(string searchRequest)
{
    var result = new OperationResult<IEnumerable<UserModel>>();

    try
    {
        var users = await _repository.GetAllAsync<UserModel>(
            u => u.TeamId != _settingsManager.TeamId
            && (u.Name.StartsWith(searchRequest,
            StringComparison.OrdinalIgnoreCase)
            || u.Surname.StartsWith(searchRequest,
            StringComparison.OrdinalIgnoreCase)
            || u.Email.StartsWith(searchRequest,
            StringComparison.OrdinalIgnoreCase)));

        if (users is not null && users.Count > 0)
        {
            result.SetSuccess(users);
        }
        else
        {
            result.SetFailure();
        }
    }
    catch (Exception ex)
    {
        result.SetError($"{nameof(GetTeamMembersAsync)} : exception",
        "Something went wrong", ex);
    }

    return result;
}

#endregion

```

```
    }
}
```

AddMembersPopupPageViewModel.cs

```
namespace TeamConnect.ViewModels.Popups
{
    public class AddMembersPopupPageViewModel : BaseViewModel
    {
        private readonly IUserService _userService;
        private readonly ITeamService _teamService;

        public AddMembersPopupPageViewModel(
            INavigationService navigationService,
            IUserService userService,
            ITeamService teamService)
            : base(navigationService)
        {
            _userService = userService;
            _teamService = teamService;
        }

        #region -- Public properties --

        private string _searchRequest;
        public string SearchRequest
        {
            get => _searchRequest;
            set => SetProperty(ref _searchRequest, value);
        }

        private bool _isUserListVisible;
        public bool IsUserListVisible
        {
            get => _isUserListVisible;
            set => SetProperty(ref _isUserListVisible, value);
        }

        private List<UserViewModel> _users;
        public List<UserViewModel> Users
        {
            get => _users;
            set => SetProperty(ref _users, value);
        }

        private int _userListHeight;
        public int UserListHeight
        {
            get => _userListHeight;
            set => SetProperty(ref _userListHeight, value);
        }

        private UserViewModel _selectedUser;
        public UserViewModel SelectedUser
        {
            get => _selectedUser;
            set => SetProperty(ref _selectedUser, value);
        }

        private ICommand _deleteUserTapCommand;
```

```

        public ICommand DeleteUserTapCommand => _deleteUserTapCommand ??= new
AsyncCommand(OnDeleteUserTapCommandAsync);

        private ICommand _cancelTapCommand;
        public ICommand CancelTapCommand => _cancelTapCommand ??= new
AsyncCommand(OnCancelTapCommandAsync);

        private ICommand _addTapCommand;
        public ICommand AddTapCommand => _addTapCommand ??= new
AsyncCommand(OnAddTapCommandAsync);

        #endregion

        #region -- Overrides --

        protected override async void
OnPropertyChanged(PropertyChangedEventArgs args)
        {
            base.OnPropertyChanged(args);

            if (args.PropertyName == nameof(SearchRequest))
            {
                if (!string.IsNullOrEmpty(SearchRequest))
                {
                    var searchResult = await
_userService.SearchUsers(SearchRequest);

                    if (searchResult.IsSuccess)
                    {
                        Users = searchResult.Result.Select(u
=> u.ToViewModel()).ToList();

                        ICommand userTapCommand = new
AsyncCommand<UserViewModel>(OnUserTapCommandAsync);
                        Users.ForEach(u => u.TapCommand = userTapCommand);
                    }
                    else
                    {
                        Users?.Clear();
                    }
                }
                else
                {
                    Users?.Clear();
                }

                ChangeListHeight();
            }
        }

        #endregion

        #region -- Private helpers --

        private Task OnUserTapCommandAsync(UserViewModel user)
        {
            SelectedUser = user;

            SearchRequest = string.Empty;

            return Task.CompletedTask;
        }

```

```

private void ChangeListHeight()
{
    IsUserListVisible = Users?.Count > 0;

    UserListHeight = Users is null
        ? UserListHeight = 0
        : Users.Count < 3
        ? Users.Count * 53
        : UserListHeight = 159;
}

private Task OnDeleteUserTapCommandAsync()
{
    SelectedUser = null;
    return Task.CompletedTask;
}

private Task OnCancelTapCommandAsync()
{
    return NavigationService.GoBackAsync();
}

private async Task OnAddTapCommandAsync()
{
    var addMemberResult = await
_teamService.AddMemberAsync(SelectedUser.ToModel());

    var parameters = new NavigationParameters();

    if (!addMemberResult.IsSuccess)
    {
        parameters.Add(Constants.Navigation.MEMBER_NOT_ADDED,
addMemberResult.Message);
    }

    await NavigationService.GoBackAsync(parameters);
}

#endregion
}
}

```

CreateTeamPopupPageViewModel.cs

```

namespace TeamConnect.ViewModels.Popups
{
    public class CreateTeamPopupPageViewModel : BaseViewModel
    {
        private readonly ITeamService _teamService;

        public CreateTeamPopupPageViewModel(
            INavigationService navigationService,
            ITeamService teamService)
            : base(navigationService)
        {
            _teamService = teamService;
        }

        #region -- Public properties --

        private string _teamName;

```

```

        public string TeamName
        {
            get => _teamName;
            set => SetProperty(ref _teamName, value);
        }

        private ICommand _cancelTapCommand;
        public ICommand CancelTapCommand => _cancelTapCommand ??= new
AsyncCommand(OnCancelTapCommandAsync);

        private ICommand _createTapCommand;
        public ICommand CreateTapCommand => _createTapCommand ??= new
AsyncCommand(OnCreateTapCommandAsync);

        #endregion

        #region -- Private helpers --

        private Task OnCancelTapCommandAsync()
        {
            return NavigationService.GoBackAsync();
        }

        private async Task OnCreateTapCommandAsync()
        {
            var team = new TeamViewModel
            {
                Name = TeamName,
            };

            await _teamService.CreateTeamAsync(team.ToModel());

            await NavigationService.GoBackAsync();
        }

        #endregion
    }
}

```

BaseViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class BaseViewModel : BindableBase, IInitialize, INavigationAware
    {
        public BaseViewModel(INavigationService navigationService)
        {
            NavigationService = navigationService;
        }

        #region -- Public properties --

        public INavigationService NavigationService { get; private set; }

        #endregion

        #region -- IInitialize implementation --

        public virtual void Initialize(INavigationParameters parameters)
        {
        }
    }
}

```

```

#endregion

#region -- INavigationAware implementation --

public virtual void OnNavigatedFrom(INavigationParameters parameters)
{
}

public virtual void OnNavigatedTo(INavigationParameters parameters)
{
    if (Device.RuntimePlatform == Device.Android &&
App.Current.MainPage is MasterDetailPage page)
    {
        page.IsGestureEnabled =
page.Detail.Navigation.NavigationStack.LastOrDefault() is BaseDetailPage;
    }
}

#endregion
}
}

```

CompleteRegistrationFirstPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class CompleteRegistrationFirstPageViewModel : BaseViewModel
    {
        private readonly IUserDialogs _userDialogs;
        private readonly IMapService _mapService;

        private UserViewModel _user;

        public CompleteRegistrationFirstPageViewModel(
            INavigationService navigationService,
            IUserDialogs userDialogs,
            IMapService mapService)
            : base(navigationService)
        {
            _userDialogs = userDialogs;
            _mapService = mapService;
        }

        #region -- Public properties --

        private string _address;
        public string Address
        {
            get => _address;
            set => SetProperty(ref _address, value);
        }

        private Pin _pin;
        public Pin Pin
        {
            get => _pin;
            set => SetProperty(ref _pin, value);
        }

        private ICommand _selectLocationCommand;
    }
}

```

```

        public ICommand SelectLocationCommand => _selectLocationCommand ??=
new AsyncCommand<MapClickedEventArgs>(OnSelectLocationCommandAsync);

        private ICommand _nextTapCommand;
        public ICommand NextTapCommand => _nextTapCommand ??= new
AsyncCommand(OnNextTapCommandAsync);

        #endregion

        #region -- Overrides --

        public override void Initialize(INavigationParameters parameters)
        {
            base.Initialize(parameters);

            if (parameters.TryGetValue(Constants.Navigation.USER, out
UserModel user))
            {
                _user = user.ToViewModel();
            }
        }

        protected override void OnPropertyChanged(PropertyChangedEventArgs
args)
        {
            base.OnPropertyChanged(args);
            if (args.PropertyName == nameof(Address) &&
string.IsNullOrEmpty(Address))
            {
                Pin = null;
            }
        }

        #endregion

        #region -- Private helpers --

        private async Task OnSelectLocationCommandAsync(MapClickedEventArgs
args)
        {
            var position = args.Position;

            _userDialogs.ShowLoading("Location processing", maskType:
MaskType.Clear);

            await SetLocationAsync(position);

            _userDialogs.HideLoading();
        }

        private async Task SetLocationAsync(Position position)
        {
            var getPlacemarkResult = await
_mapService.GetPlacemarkAsync(position.Latitude, position.Longitude);

            if (getPlacemarkResult.IsSuccess)
            {
                var placemark = getPlacemarkResult.Result;

                var getTimeZoneResult = await
_mapService.GetTimeZoneAsync(position.Latitude, position.Longitude);

```

```

        if (placemark.CountryName is not null &&
getTimeZoneResult.IsSuccess)
        {
            var locality = placemark.Locality is not null
                ? $"{placemark.Locality}, "
                : string.Empty;
            var country = placemark.CountryName;
            var adminArea = placemark.AdminArea is not null
                ? $"{placemark.AdminArea}"
                : string.Empty;

            Address = locality + country + adminArea;

            _user.Latitude = position.Latitude;
            _user.Longitude = position.Longitude;
            _user.Address = Address;
            _user.CountryCode = placemark.CountryCode;
            _user.TimeZoneId = getTimeZoneResult.Result.TimeZoneID;

            Pin = new Pin
            {
                Label = string.Empty,
                Position = position,
            };
        }
        else
        {
            await _userDialogs.AlertAsync(Strings.WrongLocation);
        }
    }
    else
    {
        await _userDialogs.AlertAsync(Strings.WrongLocation);
    }
}

private Task OnNextTapCommandAsync()
{
    var parameters = new NavigationParameters
    {
        { Constants.Navigation.USER, _user.ToModel() },
    };

    return
NavigationService.NavigateAsync(nameof(CompleteRegistrationSecondPage), parameters,
false, true);
}

#endregion
}
}

```

CompleteRegistrationSecondPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class CompleteRegistrationSecondPageViewModel : BaseViewModel
    {
        private readonly IUserDialogs _userDialogs;
        private readonly IAuthorizationService _authorizationService;
    }
}

```

```

private UserViewModel _user;

public CompleteRegistrationSecondPageViewModel(
    INavigationService navigationService,
    IUserDialogs userDialogs,
    IAuthorizationService authorizationService)
    : base(navigationService)
{
    _userDialogs = userDialogs;
    _authorizationService = authorizationService;
}

#region -- Public properties --

private List<string> _positions;
public List<string> Positions
{
    get => _positions;
    set => SetProperty(ref _positions, value);
}

private string _selectedPosition;
public string SelectedPosition
{
    get => _selectedPosition;
    set => SetProperty(ref _selectedPosition, value);
}

private bool _isStartWorkingTimeChanged;
public bool IsStartWorkingTimeChanged
{
    get => _isStartWorkingTimeChanged;
    set => SetProperty(ref _isStartWorkingTimeChanged, value);
}

private TimeSpan _startWorkingTime;
public TimeSpan StartWorkingTime
{
    get => _startWorkingTime;
    set => SetProperty(ref _startWorkingTime, value);
}

private bool _isEndWorkingTimeChanged;
public bool IsEndWorkingTimeChanged
{
    get => _isEndWorkingTimeChanged;
    set => SetProperty(ref _isEndWorkingTimeChanged, value);
}

private TimeSpan _endWorkingTime;
public TimeSpan EndWorkingTime
{
    get => _endWorkingTime;
    set => SetProperty(ref _endWorkingTime, value);
}

private ICommand _selectPositionCommand;
public ICommand SelectPositionCommand => _selectPositionCommand ??=
new AsyncCommand(OnSelectPositionCommandAsync);

private ICommand _selectStartWorkingTimeCommand;

```

```

        public ICommand SelectStartWorkingTimeCommand => new
        _selectStartWorkingTimeCommand ??=
        AsyncCommand(OnSelectStartWorkingTimeCommandAsync);

        private ICommand _selectEndWorkingTimeCommand;
        public ICommand SelectEndWorkingTimeCommand => new
        _selectEndWorkingTimeCommand ??=
        AsyncCommand(OnSelectEndWorkingTimeCommandAsync);

        private ICommand _completeRegistrationTapCommand;
        public ICommand CompleteRegistrationTapCommand => new
        _completeRegistrationTapCommand ??=
        AsyncCommand(OnCompleteRegistrationTapCommandAsync);

        #endregion

        #region -- Overrides --

        public override void Initialize(INavigationParameters parameters)
        {
            base.Initialize(parameters);

            if (parameters.TryGetValue(Constants.Navigation.USER, out
            UserModel user))
            {
                _user = user.ToViewModel();
            }

            InitializePositions();
        }

        #endregion

        #region -- Private helpers --

        private void InitializePositions()
        {
            Positions = new List<string>
            {
                Strings.HR,
                Strings.ProjectManager,
                Strings.FrontendDeveloper,
                Strings.BackendDeveloper,
                Strings.MobileDeveloper,
                Strings.QA,
                Strings.Designer,
            };
        }

        private async Task OnSelectPositionCommandAsync()
        {
            var selectedPosition = await
            _userDialogs.ActionSheetAsync(Strings.SelectPosition, Strings.Cancel, null, null,
            Positions.ToArray());

            if (selectedPosition != Strings.Cancel)
            {
                SelectedPosition = selectedPosition;
            }
        }

        private async Task OnSelectStartWorkingTimeCommandAsync()

```

```

    {
        var config = new TimePromptConfig();
        config.iOSPickerStyle = iOSPickerStyle.Wheels;

        var timePromptResult = await
_userDialogs.TimePromptAsync(config);

        if (timePromptResult.Ok)
        {
            IsStartWorkingTimeChanged = true;
            var time = timePromptResult.SelectedTime;
            StartWorkingTime = new TimeSpan(time.Days, time.Hours,
time.Minutes, 0);

            if (!IsEndWorkingTimeChanged)
            {
                IsEndWorkingTimeChanged = true;
                EndWorkingTime = StartWorkingTime +
TimeSpan.FromHours(Constants.DEFAULT_WORKING_TIME);
            }
        }
    }

    private async Task OnSelectEndWorkingTimeCommandAsync()
    {
        var config = new TimePromptConfig();
        config.iOSPickerStyle = iOSPickerStyle.Wheels;

        var timePromptResult = await
_userDialogs.TimePromptAsync(config);

        if (timePromptResult.Ok)
        {
            IsEndWorkingTimeChanged = true;
            var time = timePromptResult.SelectedTime;
            EndWorkingTime = new TimeSpan(time.Days, time.Hours,
time.Minutes, 0);

            if (!IsStartWorkingTimeChanged)
            {
                IsStartWorkingTimeChanged = true;
                StartWorkingTime = EndWorkingTime -
TimeSpan.FromHours(Constants.DEFAULT_WORKING_TIME);
            }
        }
    }

    private async Task OnCompleteRegistrationTapCommandAsync()
    {
        _user.Position = SelectedPosition;
        _user.StartWorkTime = DateTime.Now.Date + StartWorkingTime;
        _user.EndWorkTime = DateTime.Now.Date + EndWorkingTime;
        _user.IsAccountCreated = true;

        var completeRegistrationResult = await
_authorizationService.CompleteRegistration(_user.ToModelWithoutTimeConverting());

        if (completeRegistrationResult.IsSuccess)
        {
            await
NavigationService.NavigateAsync($"{nameof(MainMasterPage)}/{nameof(NavigationPage)}/{
nameof(TeamPage)}", null, false, true);

```

```

    }
}

#endregion
}
}

```

CreateAccountFirstPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class CreateAccountFirstPageViewModel : BaseViewModel
    {
        public CreateAccountFirstPageViewModel(
            INavigationService navigationService)
            : base(navigationService)
        {
        }

        #region -- Public properties --

        private string _name;
        public string Name
        {
            get => _name;
            set => SetProperty(ref _name, value);
        }

        private string _surname;
        public string Surname
        {
            get => _surname;
            set => SetProperty(ref _surname, value);
        }

        private ICommand _nextTapCommand;
        public ICommand NextTapCommand => _nextTapCommand ??= new
AsyncCommand(OnNextTapCommandAsync);

        #endregion

        #region -- Private helpers --

        private Task OnNextTapCommandAsync()
        {
            var user = new UserViewModel
            {
                Name = Name,
                Surname = Surname,
            };
            var parameters = new NavigationParameters
            {
                { Constants.Navigation.USER, user.ToModel() },
            };

            return
NavigationService.NavigateAsync(nameof(CreateAccountSecondPage), parameters, false,
true);
        }

        #endregion
    }
}

```

```
    }
}
```

CreateAccountSecondPageViewModel.cs

```
namespace TeamConnect.ViewModels
{
    public class CreateAccountSecondPageViewModel : BaseViewModel
    {
        private readonly IAuthorizationService _authorizationService;

        private UserViewModel _user;

        public CreateAccountSecondPageViewModel(
            INavigationService navigationService,
            IAuthorizationService authorizationService)
            : base(navigationService)
        {
            _authorizationService = authorizationService;
        }

        #region -- Public properties --

        private string _email;
        public string Email
        {
            get => _email;
            set => SetProperty(ref _email, value);
        }

        private bool _isEmailError;
        public bool IsEmailError
        {
            get => _isEmailError;
            set => SetProperty(ref _isEmailError, value);
        }

        private string _emailError;
        public string EmailError
        {
            get => _emailError;
            set => SetProperty(ref _emailError, value);
        }

        private string _password;
        public string Password
        {
            get => _password;
            set => SetProperty(ref _password, value);
        }

        private bool _isPasswordError;
        public bool IsPasswordError
        {
            get => _isPasswordError;
            set => SetProperty(ref _isPasswordError, value);
        }

        private string _passwordError;
        public string PasswordError
        {

```

```

        get => _passwordError;
        set => SetProperty(ref _passwordError, value);
    }

    private string _confirmPassword;
    public string ConfirmPassword
    {
        get => _confirmPassword;
        set => SetProperty(ref _confirmPassword, value);
    }

    private bool _isConfirmPasswordError;
    public bool IsConfirmPasswordError
    {
        get => _isConfirmPasswordError;
        set => SetProperty(ref _isConfirmPasswordError, value);
    }

    private string _confirmPasswordError;
    public string ConfirmPasswordError
    {
        get => _confirmPasswordError;
        set => SetProperty(ref _confirmPasswordError, value);
    }

    private ICommand _createAccountTapCommand;
    public ICommand CreateAccountTapCommand => _createAccountTapCommand
    ??= new AsyncCommand(OnCreateAccountTapCommandAsync);

    #endregion

    #region -- Overrides --

    public override void Initialize(INavigationParameters parameters)
    {
        base.Initialize(parameters);

        if (parameters.TryGetValue(Constants.Navigation.USER, out
        UserModel user))
        {
            _user = user.ToViewModel();
        }
    }

    #endregion

    #region -- Private helpers --

    private async Task OnCreateAccountTapCommandAsync()
    {
        if (CheckIsEmailValid())
        {
            if (CheckIsPasswordValid())
            {
                var checkIsEmailExistResult = await
                _authorizationService.CheckIsEmailExistAsync(Email);

                if (!checkIsEmailExistResult.IsSuccess)
                {
                    _user.Email = Email.ToLower();
                    _user.Password = Password;
                }
            }
        }
    }

```

```

        var signUpResult = await
            _authorizationService.SignUpAsync(_user.ToModel());

        if (signUpResult.IsSuccess)
        {
            var parameters = new NavigationParameters
            {
                { Constants.Navigation.EMAIL, Email },
            };

            await
                NavigationService.NavigateAsync($"/{nameof(NavigationPage)}/{nameof(LoginPage)}",
                    parameters, false, true);
        }
        else
        {
            IsEmailError = true;
            EmailError = Strings.PleaseTypeYourEmail;
        }
    }
}

private bool CheckIsEmailValid()
{
    bool result = false;

    if (Validator.CheckIsEmailValid(Email))
    {
        IsEmailError = false;
        result = true;
    }
    else
    {
        IsEmailError = true;
        EmailError = Strings.WrongEmail;
    }

    return result;
}

private bool CheckIsPasswordValid()
{
    bool result = false;
    if (Validator.CheckIsPasswordValid>Password))
    {
        IsPasswordError = false;

        if (ConfirmPassword == Password)
        {
            IsConfirmPasswordError = false;
            result = true;
        }
        else
        {
            IsConfirmPasswordError = true;
            ConfirmPasswordError = Strings.PleaseConfirmPassword;
        }
    }
    else
    {

```

```

        IsPasswordError = true;
        PasswordError = Strings.WrongPassword;
    }

    return result;
}

#endregion
}
}

```

LeavePageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class LeavePageViewModel : BaseViewModel
    {
        private readonly IUserDialogs _userDialogs;
        private readonly ILeaveService _leaveService;
        private readonly IUserService _userService;

        public LeavePageViewModel(
            INavigationService navigationService,
            IUserDialogs userDialogs,
            ILeaveService leaveService,
            IUserService userService)
            : base(navigationService)
        {
            _userDialogs = userDialogs;
            _leaveService = leaveService;
            _userService = userService;
        }

        #region -- Public properties --

        private EPageState _pageState;
        public EPageState PageState
        {
            get => _pageState;
            set => SetProperty(ref _pageState, value);
        }

        private List<LeaveGroupViewModel> _leaves;
        public List<LeaveGroupViewModel> Leaves
        {
            get => _leaves;
            set => SetProperty(ref _leaves, value);
        }

        private ICommand _selectDateTapCommand;
        public ICommand SelectDateTapCommand => _selectDateTapCommand ??= new
AsyncCommand(OnSelectDateTapCommandAsync);

        private ICommand _resetDateTapCommand;
        public ICommand ResetDateTapCommand => _resetDateTapCommand ??= new
AsyncCommand(OnResetDateTapCommandAsync);

        private ICommand _addLeaveTapCommand;
        public ICommand AddLeaveTapCommand => _addLeaveTapCommand ??= new
AsyncCommand(OnAddLeaveTapCommandAsync);

```

```

#endregion

#region -- Overrides --

parameters) public override async void OnNavigatedTo(INavigationParameters
{
    base.OnNavigatedTo(parameters);

    await LoadLeavesAsync();
}

#endregion

#region -- Private helpers --

private async Task OnResetDateTapCommandAsync()
{
    await LoadLeavesAsync();
}

private async Task LoadLeavesAsync()
{
    PageState = EPageState.Loading;

    Leaves = null;

    var getLeavesResult = await
_leaveService.GetLeavesByDatesAsync(DateTime.Now.Date, DateTime.Now.AddDays(6).Date);

    if (getLeavesResult.IsSuccess)
    {
        var leaves = getLeavesResult.Result;

        var leavesGroups = new List<LeaveGroupViewModel>();

        for (int i = 0; i < 7; i++)
        {
            leavesGroups.Add(new
LeaveGroupViewModel(DateTime.Now.AddDays(i).Date));
        }

        foreach (var leaveGroup in leavesGroups)
        {
            foreach (var leave in leaves)
            {
                if (leaveGroup.Date >= leave.StartDate &&
leaveGroup.Date <= leave.EndDate)
                {
                    var getUserResult = await
_userService.GetUserByIdAsync(leave.UserId);

                    if (getUserResult.IsSuccess)
                    {
leaveGroup.Add(leave.ToViewModel(getUserResult.Result));
                    }
                }
            }
        }
        Leaves = leavesGroups;
    }
}

```

```

        PageState = EPageState.Complete;
    }
    else
    {
        PageState = EPageState.NoResult;
    }
}

private async Task OnSelectDateTapCommandAsync()
{
    var config = new DatePromptConfig();
    config.iOSPickerStyle = iOSPickerStyle.Wheels;

    var datePromptResult = await
_userDialogs.DatePromptAsync(config);

    if (datePromptResult.Ok)
    {
        await
LoadLeavesByDateAsync(datePromptResult.SelectedDate.Date);
    }
}

private async Task LoadLeavesByDateAsync(DateTime date)
{
    PageState = EPageState.Loading;

    var leavesGroups = new List<LeaveGroupViewModel>();

    leavesGroups.Add(new LeaveGroupViewModel(date));

    var getLeavesResult = await
_leaveService.GetLeavesByDatesAsync(date, date);

    if (getLeavesResult.IsSuccess)
    {
        var leaves = getLeavesResult.Result;

        foreach (var leaveGroup in leavesGroups)
        {
            foreach (var leave in leaves)
            {
                if (leaveGroup.Date >= leave.StartDate &&
leaveGroup.Date <= leave.EndDate)
                {
                    var getUserResult = await
_userService.GetUserByIdAsync(leave.UserId);

                    if (getUserResult.IsSuccess)
                    {
                        leaveGroup.Add(leave.ToViewModel(getUserResult.Result));
                    }
                }
            }
        }

        Leaves = leavesGroups;
        PageState = EPageState.Complete;
    }
    else
    {

```

```

        Leaves = leavesGroups;
        PageState = EPageState.NoResult;
    }
}

private Task OnAddLeaveTapCommandAsync()
{
    return NavigationService.NavigateAsync(nameof(NewRequestPage),
null, false, true);
}

#endregion
}
}

```

LoginPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class LoginPageViewModel : BaseViewModel
    {
        private readonly IAuthorizationService _authorizationService;

        public LoginPageViewModel(
            INavigationService navigationService,
            IAuthorizationService authorizationService)
            : base(navigationService)
        {
            _authorizationService = authorizationService;
        }

        #region -- Public properties --

        private string _email;
        public string Email
        {
            get => _email;
            set => SetProperty(ref _email, value);
        }

        private bool _isEmailError;
        public bool IsEmailError
        {
            get => _isEmailError;
            set => SetProperty(ref _isEmailError, value);
        }

        private string _password;
        public string Password
        {
            get => _password;
            set => SetProperty(ref _password, value);
        }

        private bool _isPasswordError;
        public bool IsPasswordError
        {
            get => _isPasswordError;
            set => SetProperty(ref _isPasswordError, value);
        }
    }
}

```

```

        private ICommand _loginTapCommand;
        public ICommand LoginTapCommand => _loginTapCommand ??= new
AsyncCommand(OnLoginTapCommandAsync);

        private ICommand _signUpTapCommand;
        public ICommand SignUpTapCommand => _signUpTapCommand ??= new
AsyncCommand(OnSignUpTapCommandAsync);
        #endregion

        #region -- Overrides --
        public override void Initialize(INavigationParameters parameters)
        {
            base.Initialize(parameters);

            if (parameters.TryGetValue(Constants.Navigation.EMAIL, out string
email))
            {
                Email = email;
            }
        }
        #endregion

        #region -- Private helpers --

        private async Task OnLoginTapCommandAsync()
        {
            var loginResult = await _authorizationService.LoginAsync(Email,
Password);

            if (loginResult.IsSuccess)
            {
                IsEmailError = false;
                IsPasswordError = false;

                var user = loginResult.Result.ToViewModel();

                if (user.IsAccountCreated)
                {
                    await
NavigationService.NavigateAsync($"{nameof(MainMasterPage)}/{nameof(NavigationPage)}/{
nameof(TeamPage)}", null, false, true);
                }
                else
                {
                    var parameters = new NavigationParameters
                    {
                        { Constants.Navigation.USER, user.ToModel() },
                    };

                    await
NavigationService.NavigateAsync(nameof(CompleteRegistrationFirstPage), parameters,
false, true);
                }
            }
            else
            {
                IsEmailError = true;
                IsPasswordError = true;
            }
        }
    }

```

```

        private Task OnSignUpTapCommandAsync()
        {
            return
NavigationService.NavigateAsync(nameof(CreateAccountFirstPage), null, false, true);
        }

        #endregion
    }
}

```

MainMasterPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class MainMasterPageViewModel : BaseViewModel
    {
        private readonly IUserService _userService;
        private readonly IAuthorizationService _authorizationService;
        private readonly IUserDialogs _userDialogs;

        public MainMasterPageViewModel(
            INavigationService navigationService,
            IUserDialogs userDialogs,
            IUserService userService,
            IAuthorizationService authorizationService)
            : base(navigationService)
        {
            _userDialogs = userDialogs;
            _userService = userService;
            _authorizationService = authorizationService;
        }

        #region -- Public properties --

        private bool _isPresented;
        public bool IsPresented
        {
            get => _isPresented;
            set => SetProperty(ref _isPresented, value);
        }

        private EDetailPage _detailPage;
        public EDetailPage DetailPage
        {
            get => _detailPage;
            set => SetProperty(ref _detailPage, value);
        }

        private UserViewModel _user;
        public UserViewModel User
        {
            get => _user;
            set => SetProperty(ref _user, value);
        }

        private ICommand _teamTapCommand;
        public ICommand TeamTapCommand => _teamTapCommand ??= new
AsyncCommand(OnTeamTapCommandAsync);

        private ICommand _leaveTapCommand;
    }
}

```

```

        public ICommand LeaveTapCommand => _leaveTapCommand ??= new
AsyncCommand(OnLeaveTapCommandAsync);

        private ICommand _teamTimeTapCommand;
        public ICommand TeamTimeTapCommand => _teamTimeTapCommand ??= new
AsyncCommand(OnTeamTimeTapCommandAsync);

        private ICommand _logOutTapCommand;
        public ICommand LogOutTapCommand => _logOutTapCommand ??= new
AsyncCommand(OnLogOutTapCommandAsync);

        #endregion
        #region -- Overrides --

        public override async void Initialize(INavigationParameters
parameters)
        {
            base.Initialize(parameters);

            if (parameters.TryGetValue(Constants.Navigation.DETAIL_PAGE, out
EDetailPage detailPage))
            {
                DetailPage = detailPage;
            }

            var currentUserResult = await
_userService.GetCurrentUserAsync();

            if (currentUserResult.IsSuccess)
            {
                User = currentUserResult.Result.ToViewModel();
            }
        }

        #endregion

        #region -- Private helpers --

        private Task OnTeamTapCommandAsync()
        {
            IsPresented = false;

            var parameters = new NavigationParameters
            {
                { Constants.Navigation.DETAIL_PAGE, EDetailPage.Team },
            };

            return
NavigationService.NavigateAsync($"{nameof(MainMasterPage)}/{nameof(NavigationPage)}/{
nameof(TeamPage)}", parameters, false, true);
        }

        private Task OnLeaveTapCommandAsync()
        {
            IsPresented = false;

            var parameters = new NavigationParameters
            {
                { Constants.Navigation.DETAIL_PAGE, EDetailPage.Leave },
            };

```

```

        return
        NavigationService.NavigateAsync($"{nameof(MainMasterPage)}/{nameof(NavigationPage)}/{
        nameof(LeavePage)})", parameters, false, true);
    }

    private Task OnTeamTimeTapCommandAsync()
    {
        IsPresented = false;

        var parameters = new NavigationParameters
        {
            { Constants.Navigation.DETAIL_PAGE, EDetailPage.TeamTime },
        };

        return
        NavigationService.NavigateAsync($"{nameof(MainMasterPage)}/{nameof(NavigationPage)}/{
        nameof(TeamTimePage)})", parameters, false, true);
    }
    private async Task OnLogOutTapCommandAsync()
    {
        var islogOutConfirmed = await _userDialogs.ConfirmAsync(
            Strings.AreYouSureYouWantLogOut,
            Strings.LogOut,
            Strings.Yes,
            Strings.No);

        if (islogOutConfirmed)
        {
            IsPresented = false;

            _authorizationService.LogOut();

            await
            NavigationService.NavigateAsync($"{nameof(NavigationPage)}/{nameof(LoginPage)})",
            null, false, true);
        }
    }

    #endregion
}

```

NewRequestPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class NewRequestPageViewModel : BaseViewModel
    {
        private readonly IUserDialogs _userDialogs;
        private readonly ILeaveService _leaveService;

        public NewRequestPageViewModel(
            INavigationService navigationService,
            IUserDialogs userDialogs,
            ILeaveService leaveService)
            : base(navigationService)
        {
            _userDialogs = userDialogs;
            _leaveService = leaveService;
        }
    }
}

```

```

#region -- Public properties --

private List<string> _requestTypes;
public List<string> RequestTypes
{
    get => _requestTypes;
    set => SetProperty(ref _requestTypes, value);
}

private string _selectedRequestType;
public string SelectedRequestType
{
    get => _selectedRequestType;
    set => SetProperty(ref _selectedRequestType, value);
}

private bool _isStartDateChanged;
public bool IsStartDateChanged
{
    get => _isStartDateChanged;
    set => SetProperty(ref _isStartDateChanged, value);
}

private DateTime _startDate;
public DateTime StartDate
{
    get => _startDate;
    set => SetProperty(ref _startDate, value);
}

private bool _isEndDateChanged;
public bool IsEndDateChanged
{
    get => _isEndDateChanged;
    set => SetProperty(ref _isEndDateChanged, value);
}

private DateTime _endDate;
public DateTime EndDate
{
    get => _endDate;
    set => SetProperty(ref _endDate, value);
}

private ICommand _selectRequestTypeCommand;
public ICommand SelectRequestTypeCommand => _selectRequestTypeCommand
??= new AsyncCommand(OnSelectRequestTypeCommandAsync);

private ICommand _selectStartDateCommand;
public ICommand SelectStartDateCommand => _selectStartDateCommand ??=
new AsyncCommand(OnSelectStartDateCommandAsync);

private ICommand _selectEndDateCommand;
public ICommand SelectEndDateCommand => _selectEndDateCommand ??= new
AsyncCommand(OnSelectEndDateCommandAsync);

private ICommand _createRequestTapCommand;
public ICommand CreateRequestTapCommand => _createRequestTapCommand
??= new AsyncCommand(OnCreateRequestTapCommandAsync);

#endregion

#region -- Overrides --

```

```

public override void Initialize(INavigationParameters parameters)
{
    base.Initialize(parameters);

    InitializePositions();
}

#endregion

#region -- Private helpers --

private void InitializePositions()
{
    RequestTypes = new List<string>
    {
        Strings.Vacation,
        Strings.SickLeave,
        Strings.UnpaidLeave,
    };
}

private async Task OnSelectRequestTypeCommandAsync()
{
    var selectedType = await
_userDialogs.ActionSheetAsync(Strings.SelectType, Strings.Cancel, null, null,
RequestTypes.ToArray());
    if (selectedType != Strings.Cancel)
    {
        SelectedRequestType = selectedType;
    }
}

private async Task OnSelectStartDateCommandAsync()
{
    var config = new DatePromptConfig();
    config.iOSPickerStyle = iOSPickerStyle.Wheels;

    var datePromptResult = await
_userDialogs.DatePromptAsync(config);

    if (datePromptResult.Ok)
    {
        if (!IsEndDateChanged)
        {
            IsStartDateChanged = true;
            StartDate = datePromptResult.SelectedDate.Date;
            IsEndDateChanged = true;
            EndDate = StartDate;
        }
        else
        {
            if (datePromptResult.SelectedDate <= EndDate)
            {
                IsStartDateChanged = true;
                StartDate = datePromptResult.SelectedDate.Date;
            }
            else
            {
                await
_userDialogs.AlertAsync(Strings.StartRequestDateError);
            }
        }
    }
}

```

```

    }

    private async Task OnSelectEndDateCommandAsync()
    {
        var config = new DatePromptConfig();
        config.iOSPickerStyle = iOSPickerStyle.Wheels;

        var datePromptResult = await
        _userDialogs.DatePromptAsync(config);

        if (datePromptResult.Ok)
        {
            if (!IsStartDateChanged)
            {
                IsEndDateChanged = true;
                EndDate = datePromptResult.SelectedDate.Date;
                IsStartDateChanged = true;
                StartDate = EndDate;
            }
            else
            {
                if (datePromptResult.SelectedDate >= StartDate)
                {
                    IsEndDateChanged = true;
                    EndDate = datePromptResult.SelectedDate.Date;
                }
                else
                {
                    await
                    _userDialogs.AlertAsync(Strings.EndRequestDateError);
                }
            }
        }
    }

    private async Task OnCreateRequestTapCommandAsync()
    {
        var leave = new LeaveViewModel
        {
            Type = SelectedRequestType,
            StartDate = StartDate,
            EndDate = EndDate,
        };

        var addLeaveResult = await
        _leaveService.AddLeaveAsync(leave.ToModel());

        if (addLeaveResult.IsSuccess)
        {
            await NavigationService.GoBackAsync(null, false, true);
        }
        else
        {
            if (!string.IsNullOrEmpty(addLeaveResult.Message))
            {
                await _userDialogs.AlertAsync(addLeaveResult.Message);
            }
            else
            {
                await
                _userDialogs.AlertAsync(Strings.OoopsSomethingWentWrong);
            }
        }
    }

```

```

    }
}

#endregion
}
}

```

TeamPageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class TeamPageViewModel : BaseViewModel
    {
        private readonly IUserDialogs _userDialogs;
        private readonly ITeamService _teamService;
        private readonly IUserService _userService;

        public TeamPageViewModel(
            INavigationService navigationService,
            IUserDialogs userDialogs,
            ITeamService teamService,
            IUserService userService)
            : base(navigationService)
        {
            _userDialogs = userDialogs;
            _teamService = teamService;
            _userService = userService;
        }
        #region -- Public properties --

        private EPageState _pageState;
        public EPageState PageState
        {
            get => _pageState;
            set => SetProperty(ref _pageState, value);
        }

        private TeamViewModel _team;
        public TeamViewModel Team
        {
            get => _team;
            set => SetProperty(ref _team, value);
        }

        private List<UserViewModel> _users;
        public List<UserViewModel> Users
        {
            get => _users;
            set => SetProperty(ref _users, value);
        }

        private ICommand _createTeamTapCommand;
        public ICommand CreateTeamTapCommand => _createTeamTapCommand ??= new
AsyncCommand(OnCreateTeamTapCommandAsync);

        private ICommand _addMemberTapCommand;
        public ICommand AddMemberTapCommand => _addMemberTapCommand ??= new
AsyncCommand(OnAddMemberTapCommandAsync);

        #endregion
    }
}

```

```

#region -- Overrides --
public override async void OnNavigatedTo(INavigationParameters
parameters)
{
    base.OnNavigatedTo(parameters);

    if (parameters.TryGetValue(Constants.Navigation.MEMBER_NOT_ADDED,
out string addMemberError))
    {
        await _userDialogs.AlertAsync(addMemberError);
    }
    else
    {
        await LoadTeamAsync();
    }
}

#endregion

#region -- Private helpers --

private async Task LoadTeamAsync()
{
    var isUserInTeam = _teamService.IsUserInTeam;

    if (isUserInTeam)
    {
        var getTeamResult = await _teamService.GetTeamAsync();

        if (getTeamResult.IsSuccess)
        {
            var getAllUsersResult = await
_userService.GetTeamMembersAsync();
            if (getAllUsersResult.IsSuccess)
            {
                Team = getTeamResult.Result.ToViewModel();
                Users = getAllUsersResult.Result.Select(u =>
u.ToViewModel()).ToList();
                PageState = EPageState.Complete;
            }
            else
            {
                PageState = EPageState.NoResult;
            }
        }
        else
        {
            PageState = EPageState.NoResult;
        }
    }
    else
    {
        PageState = EPageState.NoResult;
    }
}

private Task OnCreateTeamTapCommandAsync()
{
    return
NavigationService.NavigateAsync(nameof(CreateTeamPopupPage));
}

```

```

        private Task OnAddMemberTapCommandAsync()
        {
            return
NavigationService.NavigateAsync(nameof(AddMembersPopupPage));
        }

        #endregion
    }
}

```

TeamTimePageViewModel.cs

```

namespace TeamConnect.ViewModels
{
    public class TeamTimePageViewModel : BaseViewModel
    {
        private readonly IUserDialogs _userDialogs;
        private readonly IUserService _userService;

        public TeamTimePageViewModel(
            INavigationService navigationService,
            IUserDialogs userDialogs,
            IUserService userService)
            : base(navigationService)
        {
            _userDialogs = userDialogs;
            _userService = userService;
        }

        #region -- Public properties --

        private EPageState _pageState;
        public EPageState PageState
        {
            get => _pageState;
            set => SetProperty(ref _pageState, value);
        }

        private DateTime _date;
        public DateTime Date
        {
            get => _date;
            set => SetProperty(ref _date, value);
        }

        private List<UserGroupViewModel> _teamTimes;
        public List<UserGroupViewModel> TeamTimes
        {
            get => _teamTimes;
            set => SetProperty(ref _teamTimes, value);
        }

        private ICommand _selectDateTapCommand;
        public ICommand SelectDateTapCommand => _selectDateTapCommand ??= new
AsyncCommand(OnSelectDateTapCommandAsync);

        #endregion

        #region -- Overrides --

```

```

parameters) public override async void Initialize(INavigationParameters
{
    base.Initialize(parameters);

    await LoadTeamTimesAsync(DateTime.Now.Date);
}

#endregion

#region -- Private helpers --

private async Task LoadTeamTimesAsync(DateTime date)
{
    Date = date;

    PageState = EPageState.Loading;

    var getUsersResult = await
_userService.GetNotMissingUsersAsync(Date);

    if (getUsersResult.IsSuccess)
    {
        var users = getUsersResult.Result.Select(u =>
u.ToViewModel()).ToList();

        var teamTimeGroups = new List<UserGroupViewModel>();

        for (int i = 0; i < users.Count; i++)
        {
            for (int j = i + 1; j < users.Count; j++)
            {
                var group = GetTeamTimeGroup(users[i], users[j]);

                if (group is not null
                    && group.StartWorkTime != group.EndWorkTime
                    && teamTimeGroups.FirstOrDefault(
                        t => t.StartWorkTime == group.StartWorkTime
                        && t.EndWorkTime == group.EndWorkTime) is
null)
                {
                    teamTimeGroups.Add(group);
                }
            }
        }

        foreach (var user in users)
        {
            foreach (var group in teamTimeGroups)
            {
                if (user.StartWorkTime <= group.StartWorkTime
                    && user.EndWorkTime >= group.EndWorkTime)
                {
                    group.Add(user);
                }
            }
        }

        if (teamTimeGroups.Count > 0)
        {
            TeamTimes = teamTimeGroups;
            PageState = EPageState.Complete;
        }
    }
}

```

```

        }
        else
        {
            PageState = EPageState.NoResult;
        }
    }
    else
    {
        PageState = EPageState.NoResult;
    }
}

private UserGroupViewModel GetTeamTimeGroup(UserViewModel user1,
UserViewModel user2)
{
    UserGroupViewModel result = null;

    if (user1.StartWorkTime >= user2.StartWorkTime
        && user2.EndWorkTime <= user1.EndWorkTime)
    {
        if (user1.StartWorkTime < user2.EndWorkTime)
        {
            result = new UserGroupViewModel(user1.StartWorkTime,
user2.EndWorkTime);
        }
    }
    else if (user2.StartWorkTime >= user1.StartWorkTime
        && user1.EndWorkTime <= user2.EndWorkTime)
    {
        if (user2.StartWorkTime < user1.EndWorkTime)
        {
            result = new UserGroupViewModel(user2.StartWorkTime,
user1.EndWorkTime);
        }
    }
    else if (user1.StartWorkTime >= user2.StartWorkTime
        && user1.EndWorkTime <= user2.EndWorkTime)
    {
        if (user1.StartWorkTime < user1.EndWorkTime)
        {
            result = new UserGroupViewModel(user1.StartWorkTime,
user1.EndWorkTime);
        }
    }
    else if (user2.StartWorkTime >= user1.StartWorkTime
        && user2.EndWorkTime <= user1.EndWorkTime)
    {
        if (user2.StartWorkTime < user2.EndWorkTime)
        {
            result = new UserGroupViewModel(user2.StartWorkTime,
user2.EndWorkTime);
        }
    }

    return result;
}

private async Task OnSelectDateTapCommandAsync()
{
    var config = new DatePromptConfig();
    config.iOSPickerStyle = iOSPickerStyle.Wheels;
}

```

```

        var datePromptResult = await
_userDialogs.DatePromptAsync(config);

        if (datePromptResult.Ok)
        {
            await LoadTeamTimesAsync(datePromptResult.SelectedDate);
        }
    }
#endregion
}
}

```

LeaveDataTemplate.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<DataTemplate xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="TeamConnect.Views.DataTemplates.LeaveDataTemplate"
  xmlns:viewmodels="clr-namespace:TeamConnect.Models.Leave"
  xmlns:ff="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
  x:DataType="viewmodels:LeaveViewModel">

  <StackLayout Padding="24, 0">

    <Frame HasShadow="False"
      CornerRadius="16"
      BorderColor="{DynamicResource appcolor_i2}">

      <StackLayout Orientation="Horizontal"
        Spacing="10">

        <Frame HasShadow="False"
          IsClippedToBounds="True"
          Padding="0"
          HeightRequest="64"
          WidthRequest="64"
          CornerRadius="32"
          HorizontalOptions="Start"
          VerticalOptions="CenterAndExpand">

          <ff:CachedImage Source="{Binding UserPhoto}"
            LoadingPlaceholder="{DynamicResource
pic_profile}"
            ErrorPlaceholder="{DynamicResource
pic_profile}">

          <ff:CachedImage.Triggers>

            <DataTrigger TargetType="ff:CachedImage"
              Binding="{Binding UserPhoto,
Converter={StaticResource isNullOrEmptyConverter}}"
              Value="True">

              <Setter Property="Source"
Value="{DynamicResource pic_profile}"/>

            </DataTrigger>

          </ff:CachedImage.Triggers>

        </Frame>

      </StackLayout>

    </Frame>

  </StackLayout>

</DataTemplate>
```

```

        </ff:CachedImage>

    </Frame>

    <StackLayout Spacing="10"
        VerticalOptions="CenterAndExpand">

        <Label Style="{DynamicResource tstyle_i1_2}">

            <Label.FormattedText>

                <FormattedString>

                    <Span Text="{Binding UserName}"/>
                    <Span Text=" "/>
                    <Span Text="{Binding UserSurname}"/>

                </FormattedString>

            </Label.FormattedText>

        </Label>

        <Label Style="{DynamicResource tstyle_i3_1}"
            Text="{Binding Type}"/>

    </StackLayout>

</StackLayout>

</Frame>

</StackLayout>

</DataTemplate>

```

LeaveGroupDataTemplate.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<DataTemplate xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodels="clr-namespace:TeamConnect.Models.Leave"
    x:Class="TeamConnect.Views.DataTemplates.LeaveGroupDataTemplate"
    x:DataType="viewmodels:LeaveGroupViewModel">

    <StackLayout Orientation="Horizontal"
        Padding="{OnPlatform Android='24, 10, 24, 0', iOS='24, 20,
24, 0'}">

        <Label Style="{DynamicResource tstyle_i1_2}"
            Text="{Binding Date, StringFormat='{0:dddd}'}"/>

        <Label Style="{DynamicResource tstyle_i1_2}"
            Text="{Binding Date, StringFormat='{0:dd MMM yyyy}'}"
            HorizontalOptions="EndAndExpand"/>

    </StackLayout>

</DataTemplate>

```

SearchedUsersDataTemplate.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<DataTemplate xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
  xmlns:ff="clr-
namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"  xmlns:viewmodels="clr-
namespace:TeamConnect.Models.User"

x:Class="TeamConnect.Views.DataTemplates.SearchedUsersDataTemplate"
  x:DataType="viewmodels:UserViewModel">

  <StackLayout Orientation="Horizontal"
    Spacing="10"
    Padding="0, 12"
    xct:TouchEffect.PressedBackgroundColor="{DynamicResource
appcolor_i5}"

    xct:TouchEffect.Command="{Binding TapCommand}"
    xct:TouchEffect.CommandParameter="{Binding .}">

    <Frame HasShadow="False"
      IsClippedToBounds="True"
      Padding="0"
      HeightRequest="32"
      WidthRequest="32"
      CornerRadius="16"
      HorizontalOptions="Start"
      VerticalOptions="CenterAndExpand">

      <ff:CachedImage Source="{Binding Photo}"
        LoadingPlaceholder="{DynamicResource
pic_profile}"

        ErrorPlaceholder="{DynamicResource pic_profile}">

        <ff:CachedImage.Triggers>

          <DataTrigger TargetType="ff:CachedImage"
            Binding="{Binding Photo,
Converter={StaticResource isNullOrEmptyConverter}}"
            Value="True">

            <Setter Property="Source" Value="{DynamicResource
pic_profile}"/>

          </DataTrigger>

        </ff:CachedImage.Triggers>

      </ff:CachedImage>

    </Frame>

    <Label Style="{DynamicResource tstyle_i1_2}"
      VerticalOptions="CenterAndExpand">

      <Label.FormattedText>

        <FormattedString>

          <Span Text="{Binding Name}"/>

```

```

        <Span Text=" "/>
        <Span Text="{Binding Surname}"/>

    </FormattedString>

</Label.FormattedText>

</Label>

</StackLayout>
</DataTemplate>

```

TeamDataTemplate.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<DataTemplate xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:viewmodels="clr-namespace:TeamConnect.Models.User"
  xmlns:ff="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
  x:Class="TeamConnect.Views.DataTemplates.TeamDataTemplate"
  x:DataType="viewmodels:UserViewModel">

  <StackLayout Padding="24, 0">

    <Frame HasShadow="False"
      CornerRadius="16"
      BorderColor="{DynamicResource appcolor_i2}">

      <StackLayout Orientation="Horizontal"
        Spacing="10">

        <Frame HasShadow="False"
          IsClippedToBounds="True"
          Padding="0"
          HeightRequest="64"
          WidthRequest="64"
          CornerRadius="32"
          HorizontalOptions="Start"
          VerticalOptions="CenterAndExpand">

          <ff:CachedImage Source="{Binding Photo}"
            LoadingPlaceholder="{DynamicResource
pic_profile}"
            ErrorPlaceholder="{DynamicResource
pic_profile}">

            <ff:CachedImage.Triggers>

              <DataTrigger TargetType="ff:CachedImage"
                Binding="{Binding Photo,
Converter={StaticResource isNullOrEmptyConverter}}"
                Value="True">

                <Setter Property="Source"
Value="{DynamicResource pic_profile}"/>

              </DataTrigger>

            </ff:CachedImage.Triggers>

```

```

        </ff:CachedImage>

    </Frame>

    <StackLayout Spacing="10"
                VerticalOptions="CenterAndExpand">
        <Label Style="{DynamicResource tstyle_i1_2}">

            <Label.FormattedText>

                <FormattedString>

                    <Span Text="{Binding Name}"/>
                    <Span Text=" "/>
                    <Span Text="{Binding Surname}"/>

                </FormattedString>

            </Label.FormattedText>

        </Label>

        <Label Style="{DynamicResource tstyle_i3_1}"
                Text="{Binding Position}"/>

    </StackLayout>

</StackLayout>

</Frame>

</StackLayout>

</DataTemplate>

```

TeamTimeDataTemplate.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<DataTemplate xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:viewmodels="clr-namespace:TeamConnect.Models.User"
              xmlns:ff="clr-
namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
              x:Class="TeamConnect.Views.DataTemplates.TeamTimeDataTemplate"
              x:DataType="viewmodels:UserViewModel">

    <StackLayout Padding="24, 0">

        <Frame HasShadow="False"
                CornerRadius="16"
                BorderColor="{DynamicResource appcolor_i2}">

            <StackLayout Orientation="Horizontal"
                        Spacing="10">

                <Frame HasShadow="False"
                        IsClippedToBounds="True"
                        Padding="0"
                        HeightRequest="64"
                        WidthRequest="64"
                        CornerRadius="32"

```

```

        HorizontalOptions="Start"
        VerticalOptions="CenterAndExpand">
        <ff:CachedImage Source="{Binding Photo}"
                    LoadingPlaceholder="{DynamicResource
pic_profile}"
                    ErrorPlaceholder="{DynamicResource
pic_profile}">
            <ff:CachedImage.Triggers>
                <DataTrigger TargetType="ff:CachedImage"
                            Binding="{Binding Photo,
Converter={StaticResource isNullOrEmptyConverter}}"
                            Value="True">
                    <Setter Property="Source"
Value="{DynamicResource pic_profile}"/>
                </DataTrigger>
            </ff:CachedImage.Triggers>
        </ff:CachedImage>
    </Frame>
<StackLayout Spacing="10"
            VerticalOptions="CenterAndExpand">
    <Label Style="{DynamicResource tstyle_i1_2}">
        <Label.FormattedText>
            <FormattedString>
                <Span Text="{Binding Name}"/>
                <Span Text=" "/>
                <Span Text="{Binding Surname}"/>
            </FormattedString>
        </Label.FormattedText>
    </Label>
    <Label Style="{DynamicResource tstyle_i3_1}"
        Text="{Binding Position}"/>
</StackLayout>
</StackLayout>
</Frame>
</StackLayout>
</DataTemplate>

```

TeamTimeGroupDataTemplate.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<DataTemplate xmlns="http://xamarin.com/schemas/2014/forms"

```

```

        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:viewmodels="clr-namespace:TeamConnect.Models.User"
        x:Class="TeamConnect.Views.DataTemplates.
TeamTimeGroupDataTemplate"
        x:DataType="viewmodels:UserGroupViewModel">

    <StackLayout Padding="{OnPlatform Android='24, 10, 24, 0', iOS='24, 20,
24, 0'}">

        <Label Style="{DynamicResource tstyle_i1_2}">

            <Label.FormattedText>

                <FormattedString>

                    <Span Text="{Binding StartWorkTime,
StringFormat='{0:HH\\:mm}'}"/>

                    <Span Text=" - "/>

                    <Span Text="{Binding EndWorkTime,
StringFormat='{0:HH\\:mm}'}"/>

                </FormattedString>

            </Label.FormattedText>

        </Label>

    </StackLayout>

</DataTemplate>

```

AddMembersPopupPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<popup:PopupPage xmlns:popup="clr-
namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
        xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
        xmlns:controls="clr-namespace:TeamConnect.Controls"
        xmlns:viewmodels="clr-
namespace:TeamConnect.ViewModels.Popups"
        xmlns:datatemplates="clr-
namespace:TeamConnect.Views.DataTemplates"
        xmlns:ff="clr-
namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
        x:DataType="viewmodels:AddMembersPopupPageViewModel"
        x:Class="TeamConnect.Views.Popups.AddMembersPopupPage"
        Padding="20">

    <Frame HasShadow="False"
        BackgroundColor="{DynamicResource appcolor_i4}"
        CornerRadius="16"
        HorizontalOptions="Fill"
        VerticalOptions="Center">

        <Grid RowDefinitions="Auto, Auto, Auto, Auto"
            RowSpacing="20">

```

```

<Label Grid.Row="0"
    Style="{DynamicResource tstyle_i1_2}"
    Text="{xct:Translate AddMember}"/>

<controls:CustomEntry Grid.Row="1"
    Style="{DynamicResource estyle_i1}"
    Placeholder="{xct:Translate
SearchByNameSurnameOrEmail}"
    Text="{Binding SearchRequest}">

    <controls:CustomEntry.Triggers>

        <DataTrigger TargetType="controls:CustomEntry"
            Binding="{Binding SelectedUser,
Converter={StaticResource isNotNullOrEmptyConverter}}"
            Value="True">

            <Setter Property="IsVisible" Value="False"/>

        </DataTrigger>

    </controls:CustomEntry.Triggers>
</controls:CustomEntry>

<Frame Grid.Row="1"
    HasShadow="False"
    Padding="16, 18"
    CornerRadius="16"
    BorderColor="{DynamicResource appcolor_i1}"
    IsVisible="False">

    <Frame.Triggers>

        <DataTrigger TargetType="Frame"
            Binding="{Binding SelectedUser,
Converter={StaticResource isNotNullOrEmptyConverter}}"
            Value="True">

            <Setter Property="IsVisible" Value="True"/>

        </DataTrigger>

    </Frame.Triggers>

    <Grid ColumnDefinitions="Auto, *, Auto"
        ColumnSpacing="10">

        <Frame Grid.Column="0"
            HasShadow="False"
            IsClippedToBounds="True"
            Padding="0"
            HeightRequest="32"
            WidthRequest="32"
            CornerRadius="16"
            HorizontalOptions="Start"
            VerticalOptions="CenterAndExpand">

            <ff:CachedImage Source="{Binding SelectedUser.Photo}"
                LoadingPlaceholder="{DynamicResource
pic_profile}"

```

```

ErrorPlaceholder="{DynamicResource
pic_profile}">

    <ff:CachedImage.Triggers>

        <DataTrigger TargetType="ff:CachedImage"
            Binding="{Binding
SelectedUser.Photo, Converter={StaticResource isNullOrEmptyConverter}}"
            Value="True">
            <Setter Property="Source"
Value="{DynamicResource pic_profile}"/>
        </DataTrigger>
    </ff:CachedImage.Triggers>

</ff:CachedImage>

</Frame>

<Label Grid.Column="1"
    Style="{DynamicResource tstyle_i2_2}"
    VerticalOptions="CenterAndExpand">

    <Label.FormattedText>

        <FormattedString>
            <Span Text="{Binding SelectedUser.Name}"/>
            <Span Text=" "/>
            <Span Text="{Binding SelectedUser.Surname}"/>
        </FormattedString>

    </Label.FormattedText>

</Label>

<Frame Grid.Column="2"
    HasShadow="False"
    Padding="0"
    HeightRequest="24"
    WidthRequest="24"
    CornerRadius="12"
    BackgroundColor="{DynamicResource appcolor_i4}"

xct:TouchEffect.PressedBackgroundColor="{DynamicResource appcolor_i5}"
xct:TouchEffect.Command="{Binding
DeleteUserTapCommand}"

    HorizontalOptions="EndAndExpand"
    VerticalOptions="CenterAndExpand">

    <Image Source="{DynamicResource ic_close}"
        HorizontalOptions="Center"
        VerticalOptions="Center"
        Margin="3"/>

</Frame>

</Grid>

</Frame>

<CollectionView Grid.Row="2"

```

```

ItemsSource="{Binding Users}"
HeightRequest="{Binding UserListHeight}"
IsVisible="{Binding IsUserListVisible}">

<CollectionView.ItemTemplate>

    <datatemplates:SearchedUsersDataTemplate/>

</CollectionView.ItemTemplate>

</CollectionView>

<Grid Grid.Row="3"
      ColumnSpacing="10">

    <Button Grid.Column="0"
            Style="{DynamicResource bstyle_i2}"
            Text="{xct:Translate Cancel}"
            Command="{Binding CancelTapCommand}"/>

    <Button Grid.Column="1"
            Style="{DynamicResource bstyle_i1}"
            Text="{xct:Translate Add}"
            Command="{Binding AddTapCommand}"
            InputTransparent="True">

        <Button.Triggers>

            <Trigger TargetType="Button"
                    Property="InputTransparent"
                    Value="True">

                <Setter Property="Style" Value="{DynamicResource
bstyle_i3}"/>

            </Trigger>

            <DataTrigger TargetType="Button"
                        Binding="{Binding SelectedUser,
Converter={StaticResource isNotNullOrEmptyConverter}}">
                Value="True">

                <Setter
                    Property="InputTransparent"
Value="False"/>

            </DataTrigger>

        </Button.Triggers>

    </Button>

</Grid>

</Grid>

</Frame>

</popup:PopupPage>

```

CreateTeamPopupPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<popup:PopupPage
    namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
    xmlns:popup="clr-
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:controls="clr-namespace:TeamConnect.Controls"
    xmlns:viewmodels="clr-
    namespace:TeamConnect.ViewModels.Popups"
    x:DataType="viewmodels:CreateTeamPopupPageViewModel"
    x:Class="TeamConnect.Views.Popups.CreateTeamPopupPage"
    Padding="20">

    <Frame HasShadow="False"
        BackgroundColor="{DynamicResource appcolor_i4}"
        CornerRadius="16"
        HorizontalOptions="Center"
        VerticalOptions="Center">

        <StackLayout Spacing="20">
            <Label Style="{DynamicResource tstyle_i1_2}"
                Text="{xct:Translate CreateNewTeam}"/>

            <controls:CustomEntry Style="{DynamicResource estyle_i1}"
                Placeholder="{xct:Translate TeamName}"
                Text="{Binding TeamName}"/>

            <Grid ColumnSpacing="10">

                <Button Grid.Column="0"
                    Style="{DynamicResource bstyle_i2}"
                    Text="{xct:Translate Cancel}"
                    Command="{Binding CancelTapCommand}"/>

                <Button Grid.Column="1"
                    Style="{DynamicResource bstyle_i1}"
                    Text="{xct:Translate Create}"
                    Command="{Binding CreateTapCommand}"
                    InputTransparent="True">

                    <Button.Triggers>

                        <Trigger TargetType="Button"
                            Property="InputTransparent"
                            Value="True">

                            <Setter Property="Style" Value="{DynamicResource
bstyle_i3}"/>

                        </Trigger>

                        <DataTrigger TargetType="Button"
                            Binding="{Binding TeamName,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                            Value="True">

                            <Setter
                                Property="InputTransparent"
                                Value="False"/>

```

```

        </DataTrigger>

        </Button.Triggers>

    </Button>

    </Grid>
</StackLayout>

</Frame>

</popup:PopupPage>

```

BaseContentPage.cs

```

namespace TeamConnect.Views
{
    public class BaseContentPage : ContentPage
    {
        public BaseContentPage()
        {
            ViewModelLocator.SetAutowireViewModel(this, true);
            On<iOS>().SetUseSafeArea(true);
        }
    }
}

```

BaseDetailPage.cs

```

namespace TeamConnect.Views
{
    public class BaseDetailPage : BaseContentPage
    {
    }
}

```

CompleteRegistrationFirstPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<pages:BaseContentPage xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodels="clr-
namespace:TeamConnect.ViewModels"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:controls="clr-namespace:TeamConnect.Controls"
    x:Class="TeamConnect.Views.
CompleteRegistrationFirstPage"
    x:DataType="viewmodels:CompleteRegistrationFirstPageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"
    Title="{xct:Translate CompleteRegistration}">

    <StackLayout Spacing="0">

        <controls:CustomEntry Style="{DynamicResource estyle_i1}"
            Placeholder="{xct:Translate Address}"
            Text="{Binding Address}"

```

```

        Margin="24"/>

<Grid HorizontalOptions="FillAndExpand"
      VerticalOptions="FillAndExpand">

    <controls:CustomMap IsShowingUser="True"
        Pin="{Binding Pin}">

        <controls:CustomMap.Behaviors>

            <xct:EventToCommandBehavior EventName="MapClicked"
                Command="{Binding
SelectLocationCommand}"/>

        </controls:CustomMap.Behaviors>

    </controls:CustomMap>

    <Button Style="{DynamicResource bstyle_i1}"
        Text="{xct:Translate Next}"
        Command="{Binding NextTapCommand}"
        InputTransparent="True"
        VerticalOptions="End"
        Margin="24, 0, 24, 24">

        <Button.Triggers>

            <Trigger TargetType="Button"
                Property="InputTransparent"
                Value="True">

                <Setter Property="Style" Value="{DynamicResource
bstyle_i3}"/>

            </Trigger>

            <DataTrigger TargetType="Button"
                Binding="{Binding Pin,
Converter={StaticResource isNotNullOrEmptyConverter}}">
                Value="True">

                <Setter Property="InputTransparent" Value="False"/>

            </DataTrigger>

        </Button.Triggers>

    </Button>

</Grid>

</StackLayout>

</pages:BaseContentPage>

```

CompleteRegistrationSecondPage.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<pages:BaseContentPage xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"

```

```

                                xmlns:viewmodels="clr-
namespace:TeamConnect.ViewModels"

x:Class="TeamConnect.Views.CompleteRegistrationSecondPage"

x:DataType="viewmodels:CompleteRegistrationSecondPageViewModel"
                                BackgroundColor="{DynamicResource tcolor_i3}"
                                Title="{xct:Translate CompleteRegistration}">

    <StackLayout Padding="24, 34, 24, 24"
                Spacing="16">

        <Label Style="{DynamicResource tstyle_i1_1}"
                Text="{xct:Translate Position}"/>

        <Frame HasShadow="False"
                Padding="16, 18"
                CornerRadius="16"
                BorderColor="{DynamicResource appcolor_i2}"
                BackgroundColor="Transparent">

            <Frame.GestureRecognizers>

                <TapGestureRecognizer                                Command="{Binding
SelectPositionCommand}"/>

            </Frame.GestureRecognizers>
            <Label Text="{xct:Translate SelectPosition}"
                Style="{DynamicResource tstyle_i2_1}">

                <Label.Triggers>

                    <DataTrigger TargetType="Label"
                                Binding="{Binding                                SelectedPosition,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                                Value="True">

                        <Setter                                Property="Text"                                Value="{Binding
SelectedPosition}"/>

                        <Setter Property="TextColor" Value="{DynamicResource
tcolor_i1}"/>

                    </DataTrigger>

                </Label.Triggers>

            </Label>

        </Frame>

        <Label Style="{DynamicResource tstyle_i1_1}"
                Text="{xct:Translate WorkingTime}"
                Margin="0, 34, 0, 0"/>

        <Frame HasShadow="False"
                Padding="16, 18"
                CornerRadius="16"
                BorderColor="{DynamicResource appcolor_i2}"
                BackgroundColor="Transparent">
            <Frame.GestureRecognizers>

```

```

        <TapGestureRecognizer                                Command="{Binding
SelectStartWorkingTimeCommand}"/>

    </Frame.GestureRecognizers>

    <Label Text="{xct:Translate From}"
        Style="{DynamicResource tstyle_i2_1}">

        <Label.Triggers>

            <DataTrigger TargetType="Label"
                Binding="{Binding
IsStartWorkingTimeChanged}"
                Value="True">

                <Setter          Property="Text"          Value="{Binding
StartWorkingTime}"/>

                <Setter Property="TextColor" Value="{DynamicResource
tcolor_i1}"/>

            </DataTrigger>

        </Label.Triggers>

    </Label>

</Frame>

<Frame HasShadow="False"
    Padding="16, 18"
    CornerRadius="16"
    BorderColor="{DynamicResource appcolor_i2}"
    BackgroundColor="Transparent">

    <Frame.GestureRecognizers>

        <TapGestureRecognizer                                Command="{Binding
SelectEndWorkingTimeCommand}"/>

    </Frame.GestureRecognizers>

    <Label Text="{xct:Translate To}"
        Style="{DynamicResource tstyle_i2_1}">
        <Label.Triggers>

            <DataTrigger TargetType="Label"
                Binding="{Binding IsEndWorkingTimeChanged}"
                Value="True">

                <Setter          Property="Text"          Value="{Binding
EndWorkingTime}"/>

                <Setter Property="TextColor" Value="{DynamicResource
tcolor_i1}"/>

            </DataTrigger>

        </Label.Triggers>

    </Label>

</Frame>

```

```

        <Button Style="{DynamicResource bstyle_i3}"
            Text="{xct:Translate CompleteRegistration}"
            Command="{Binding CompleteRegistrationTapCommand}"
            InputTransparent="True"
            VerticalOptions="EndAndExpand">

            <Button.Triggers>

                <Trigger TargetType="Button"
                    Property="InputTransparent"
                    Value="False">

                    <Setter Property="Style" Value="{DynamicResource
bstyle_i1}"/>

                </Trigger>

                <MultiTrigger TargetType="Button">

                    <MultiTrigger.Conditions>

                        <BindingCondition Binding="{Binding SelectedPosition,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                            Value="True"/>

                        <BindingCondition Binding="{Binding
IsStartWorkingTimeChanged}"
                            Value="True"/>

                        <BindingCondition Binding="{Binding
IsEndWorkingTimeChanged}"
                            Value="True"/>

                    </MultiTrigger.Conditions>

                    <Setter Property="InputTransparent" Value="False"/>

                </MultiTrigger>

            </Button.Triggers>

        </Button>

    </StackLayout>

</pages:BaseContentPage>

```

CreateAccountFirstPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<pages:BaseContentPage xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodels="clr-
namespace:TeamConnect.ViewModels"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:controls="clr-namespace:TeamConnect.Controls"
    x:Class="TeamConnect.Views.CreateAccountFirstPage"

    x:DataType="viewmodels:CreateAccountFirstPageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"

```

```

        Title="{xct:Translate CreateAccount}">
<StackLayout Padding="24, 100, 24, 0"
    Spacing="16">

    <controls:CustomEntry Style="{DynamicResource estyle_i1}"
        Placeholder="{xct:Translate Name}"
        Text="{Binding Name}"/>
    <controls:CustomEntry Style="{DynamicResource estyle_i1}"
        Placeholder="{xct:Translate Surname}"
        Text="{Binding Surname}"/>

    <Button Style="{DynamicResource bstyle_i3}"
        Text="{xct:Translate Next}"
        Command="{Binding NextTapCommand}"
        InputTransparent="True"
        Margin="0, 156, 0, 0">

        <Button.Triggers>

            <Trigger TargetType="Button"
                Property="InputTransparent"
                Value="False">

                <Setter      Property="Style"      Value="{DynamicResource
bstyle_i1}"/>

            </Trigger>

            <MultiTrigger TargetType="Button">

                <MultiTrigger.Conditions>

                    <BindingCondition Binding="{Binding Name,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                        Value="True"/>

                    <BindingCondition Binding="{Binding Surname,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                        Value="True"/>

                </MultiTrigger.Conditions>

                <Setter Property="InputTransparent" Value="False"/>

            </MultiTrigger>

        </Button.Triggers>

    </Button>

</StackLayout>

</pages:BaseContentPage>

```

CreateAccountSecondPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<pages:BaseContentPage xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

```

```

xmlns:viewmodels="clr-
namespace:TeamConnect.ViewModels"
xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
xmlns:controls="clr-namespace:TeamConnect.Controls"
x:Class="TeamConnect.Views.CreateAccountSecondPage"

x:DataType="viewmodels:CreateAccountSecondPageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"
    Title="{xct:Translate CreateAccount}">

<StackLayout Padding="24, 100, 24, 0"
    Spacing="16">

    <controls:CustomEntry Style="{DynamicResource estyle_i1}"
        Placeholder="{xct:Translate Email}"
        Text="{Binding Email}"
        IsError="{Binding IsEmailError}"
        ErrorText="{Binding EmailError}"/>

    <controls:CustomEntry Style="{DynamicResource estyle_i1}"
        Placeholder="{xct:Translate Password}"
        Text="{Binding Password}"
        PasswordIcon="{DynamicResource ic_eye_off}"
        NotPasswordIcon="{DynamicResource ic_eye}"
        IsPassword="True"
        IsPasswordButtonVisible="True"
        IsError="{Binding IsPasswordError}"
        ErrorText="{Binding PasswordError}"/>

    <controls:CustomEntry Style="{DynamicResource estyle_i1}"
        Placeholder="{xct:Translate ConfirmPassword}"
        Text="{Binding ConfirmPassword}"
        PasswordIcon="{DynamicResource ic_eye_off}"
        NotPasswordIcon="{DynamicResource ic_eye}"
        IsPassword="True"
        IsPasswordButtonVisible="True"
        IsError="{Binding IsConfirmPasswordError}"
        ErrorText="{Binding ConfirmPasswordError}"/>

    <Button Style="{DynamicResource bstyle_i3}"
        Text="{xct:Translate CreateAccount}"
        Command="{Binding CreateAccountTapCommand}"
        InputTransparent="True"
        Margin="0, 84, 0, 0">

        <Button.Triggers>

            <Trigger TargetType="Button"
                Property="InputTransparent"
                Value="False">

                <Setter Property="Style" Value="{DynamicResource
bstyle_i1}"/>

            </Trigger>

            <MultiTrigger TargetType="Button">

                <MultiTrigger.Conditions>

                    <BindingCondition Binding="{Binding Email,
Converter={StaticResource isNotNullOrEmptyConverter}}">

```

```

Value="True"/>
        <BindingCondition Binding="{Binding Password,
Converter={StaticResource isNotNullOrEmptyConverter}}"
Value="True"/>
        <BindingCondition Binding="{Binding ConfirmPassword,
Converter={StaticResource isNotNullOrEmptyConverter}}"
Value="True"/>
    </MultiTrigger.Conditions>
    <Setter Property="InputTransparent" Value="False"/>
</MultiTrigger>
</Button.Triggers>
</Button>
</StackLayout>
</pages:BaseContentPage>

```

LeavePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<page:BaseDetailPage xmlns:page="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:viewmodels="clr-namespace:TeamConnect.ViewModels"
    xmlns:datatemplates="clr-
namespace:TeamConnect.Views.DataTemplates"
    x:Class="TeamConnect.Views.LeavePage"
    x:DataType="viewmodels:LeavePageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"
    Title="{xct:Translate Leave}">
    <page:BaseDetailPage.ToolbarItems>
        <ToolbarItem IconImageSource="{DynamicResource ic_calendar}"
            Command="{Binding SelectDateTapCommand}"/>
        <ToolbarItem IconImageSource="{DynamicResource ic_reset}"
            Command="{Binding ResetDateTapCommand}"/>
    </page:BaseDetailPage.ToolbarItems>
    <Grid>
        <CollectionView ItemsSource="{Binding Leaves}"
            IsGrouped="True">
            <CollectionView.ItemsLayout>
                <LinearItemsLayout Orientation="Vertical"
                    ItemSpacing="10"/>
            </CollectionView.ItemsLayout>
            <CollectionView.Header>

```

```

        <StackLayout HeightRequest="{OnPlatform iOS=10}" />
    </CollectionView.Header>

    <CollectionView.GroupHeaderTemplate>

        <datatemplates:LeaveGroupDataTemplate />

    </CollectionView.GroupHeaderTemplate>

    <CollectionView.ItemTemplate>

        <datatemplates:LeaveDataTemplate />
    </CollectionView.ItemTemplate>

    <CollectionView.Footer>

        <StackLayout HeightRequest="20" />

    </CollectionView.Footer>
</CollectionView>

<ActivityIndicator Style="{DynamicResource activityIndicator}"
    IsVisible="False">

    <ActivityIndicator.Triggers>

        <DataTrigger TargetType="ActivityIndicator"
            Binding="{Binding PageState}"
            Value="Loading">

            <Setter Property="IsVisible" Value="True" />

        </DataTrigger>

    </ActivityIndicator.Triggers>
</ActivityIndicator>

<Label Style="{DynamicResource tstyle_i1_2}"
    Text="{xct:Translate NoLeave}"
    HorizontalOptions="Center"
    VerticalOptions="Center"
    IsVisible="False">

    <Label.Triggers>

        <DataTrigger TargetType="Label"
            Binding="{Binding PageState}"
            Value="NoResult">

            <Setter Property="IsVisible" Value="True" />

        </DataTrigger>

    </Label.Triggers>
</Label>

<Button WidthRequest="64"
    HeightRequest="64"

```

```

        ImageSource="{DynamicResource ic_plus}"
        CornerRadius="32"
        BackgroundColor="Blue"
        HorizontalOptions="End"
        VerticalOptions="End"
        Margin="24"
        Command="{Binding AddLeaveTapCommand}"/>

</Grid>

</page:BaseDetailPage>

```

LoginPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<pages:BaseContentPage xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodels="clr-namespace:TeamConnect.ViewModels"
    xmlns:controls="clr-namespace:TeamConnect.Controls"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    x:Class="TeamConnect.Views.LoginPage"
    x:DataType="viewmodels:LoginPageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"
    Title="{xct:Translate Login}">

    <StackLayout Padding="24, 100, 24, 0"
        Spacing="16">

        <controls:CustomEntry Style="{DynamicResource estyle_i1}"
            Placeholder="{xct:Translate Email}"
            Text="{Binding Email}"
            IsError="{Binding IsEmailError}"
            ErrorText="{xct:Translate InvalidEmail}"/>

        <controls:CustomEntry Style="{DynamicResource estyle_i1}"
            Placeholder="{xct:Translate Password}"
            Text="{Binding Password}"
            PasswordIcon="{DynamicResource ic_eye_off}"
            NotPasswordIcon="{DynamicResource ic_eye}"
            IsPassword="True"
            IsPasswordButtonVisible="True"
            IsError="{Binding IsPasswordError}"
            ErrorText="{xct:Translate InvalidPassword}"/>

        <Button Style="{DynamicResource bstyle_i1}"
            Text="{xct:Translate Login}"
            Command="{Binding LogInTapCommand}"
            InputTransparent="True"
            Margin="0, 84, 0, 0">

            <Button.Triggers>

                <Trigger TargetType="Button"
                    Property="InputTransparent"
                    Value="True">

                    <Setter Property="Style" Value="{DynamicResource
bstyle_i3}"/>

```

```

        </Trigger>

        <MultiTrigger TargetType="Button">
            <MultiTrigger.Conditions>
                <BindingCondition Binding="{Binding Email,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                                Value="True"/>
                <BindingCondition Binding="{Binding Password,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                                Value="True"/>
            </MultiTrigger.Conditions>
            <Setter Property="InputTransparent" Value="False"/>
        </MultiTrigger>
    </Button.Triggers>
</Button>

<Button Style="{DynamicResource bstyle_i2}"
        Text="{xct:Translate SignUp}"
        Command="{Binding SignUpTapCommand}"/>

</StackLayout>

</pages:BaseContentPage>

```

MainMasterPage.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns:viewmodels="clr-namespace:TeamConnect.ViewModels"
    xmlns:ff="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
    x:Class="TeamConnect.Views.MainMasterPage"
    x:DataType="viewmodels:MainMasterPageViewModel"
    IsGestureEnabled="{OnPlatform iOS=False}"
    IsPresented="{Binding IsPresented}">
    <MasterDetailPage.Master>
        <pages:BaseContentPage
            appcolor_i4}"
            BackgroundColor="{DynamicResource
            Title="{xct:Translate Menu}">
        <StackLayout Spacing="0">
            <Grid ColumnDefinitions="Auto, *"
                Padding="24"
                ColumnSpacing="10">
                <Frame Grid.Column="0"
                    HasShadow="False"
                    IsClippedToBounds="True"

```

```

        Padding="0"
        HeightRequest="64"
        WidthRequest="64"
        CornerRadius="32"
        HorizontalOptions="Start"
        VerticalOptions="CenterAndExpand">
<ff:CachedImage Source="{Binding User.Photo}"
        LoadingPlaceholder="{DynamicResource
pic_profile}"
        ErrorPlaceholder="{DynamicResource
pic_profile}">

        <ff:CachedImage.Triggers>

            <DataTrigger TargetType="ff:CachedImage"
                Binding="{Binding User.Photo,
Converter={StaticResource isNullOrEmptyConverter}}"
                Value="True">

                <Setter Property="Source"
Value="{DynamicResource pic_profile}"/>
            </DataTrigger>

        </ff:CachedImage.Triggers>

    </ff:CachedImage>

</Frame>

<StackLayout Grid.Column="1"
    Spacing="0"
    VerticalOptions="CenterAndExpand">

    <Label Style="{DynamicResource tstyle_i1_2}">

        <Label.FormattedText>

            <FormattedString>

                <Span Text="{Binding User.Name}"/>
                <Span Text=" "/>
                <Span Text="{Binding User.Surname}"/>

            </FormattedString>

        </Label.FormattedText>

    </Label>

    <Label Style="{DynamicResource tstyle_i3_1}"
        Text="{Binding User.Email}"
        LineBreakMode="TailTruncation"/>

</StackLayout>

</Grid>

<Frame x:Name="team"
    HasShadow="False"
    CornerRadius="0"
    BackgroundColor="{DynamicResource appcolor_i4}"
    xct:TouchEffect.Command="{Binding TeamTapCommand}"

```

```

xct:TouchEffect.PressedBackgroundColor="{DynamicResource appcolor_i5}"
    Padding="24, 20">

    <StackLayout Orientation="Horizontal"
        Spacing="16">

        <Image Source="{DynamicResource ic_team_gray}"
            VerticalOptions="CenterAndExpand">

            <Image.Triggers>

                <DataTrigger TargetType="Image"
                    Binding="{Binding DetailPage}"
                    Value="Team">

                    <Setter Property="Source"
                        Value="{DynamicResource ic_team_blue}"/>

                </DataTrigger>

            </Image.Triggers>

        </Image>
        <Label Style="{DynamicResource tstyle_i4_1}"
            Text="{xct:Translate Team}"
            VerticalOptions="CenterAndExpand">

            <Label.Triggers>

                <DataTrigger TargetType="Label"
                    Binding="{Binding DetailPage}"
                    Value="Team">

                    <Setter Property="TextColor"
                        Value="{DynamicResource tcolor_i4}"/>

                </DataTrigger>

            </Label.Triggers>

        </Label>

    </StackLayout>

</Frame>

<Frame HasShadow="False"
    CornerRadius="0"
    BackgroundColor="{DynamicResource appcolor_i4}"
    xct:TouchEffect.Command="{Binding LeaveTapCommand}"

    xct:TouchEffect.PressedBackgroundColor="{DynamicResource appcolor_i5}"
        Padding="24, 20">

    <StackLayout Orientation="Horizontal"
        Spacing="16">

        <Image Source="{DynamicResource ic_plane_gray}"
            VerticalOptions="CenterAndExpand">

            <Image.Triggers>

```

```

        <DataTrigger TargetType="Image"
                    Binding="{Binding DetailPage}"
                    Value="Leave">

            <Setter                                     Property="Source"
Value="{DynamicResource ic_plane_blue}"/>

        </DataTrigger>

    </Image.Triggers>

</Image>

<Label Style="{DynamicResource tstyle_i4_1}"
        Text="{xct:Translate Leave}"
        VerticalOptions="CenterAndExpand">

    <Label.Triggers>

        <DataTrigger TargetType="Label"
                    Binding="{Binding DetailPage}"
                    Value="Leave">

            <Setter                                     Property="TextColor"
Value="{DynamicResource tcolor_i4}"/>

        </DataTrigger>

    </Label.Triggers>
</Label>

</StackLayout>

</Frame>

<Frame HasShadow="False"
        CornerRadius="0"
        BackgroundColor="{DynamicResource appcolor_i4}"
        xct:TouchEffect.Command="{Binding TeamTimeTapCommand}"
xct:TouchEffect.PressedBackgroundColor="{DynamicResource appcolor_i5}"
        Padding="24, 20">

    <StackLayout Orientation="Horizontal"
                Spacing="16">

        <Image Source="{DynamicResource ic_clock_gray}"
                VerticalOptions="CenterAndExpand">

            <Image.Triggers>

                <DataTrigger TargetType="Image"
                            Binding="{Binding DetailPage}"
                            Value="TeamTime">

                    <Setter                                     Property="Source"
Value="{DynamicResource ic_clock_blue}"/>

                </DataTrigger>

            </Image.Triggers>

```

```

</Image>

<Label Style="{DynamicResource tstyle_i4_1}"
      Text="{xct:Translate TeamTime}"
      VerticalOptions="CenterAndExpand">

    <Label.Triggers>

        <DataTrigger TargetType="Label"
                      Binding="{Binding DetailPage}"
                      Value="TeamTime">

            <Setter Property="TextColor"
Value="{DynamicResource tcolor_i4}"/>

        </DataTrigger>

    </Label.Triggers>

</Label>

</StackLayout>

</Frame>

<Frame HasShadow="False"
      CornerRadius="0"
      BackgroundColor="{DynamicResource appcolor_i4}"
      xct:TouchEvent.Command="{Binding LogOutTapCommand}"
xct:TouchEvent.PressedBackgroundColor="{DynamicResource appcolor_i5}"
      Padding="24, 20"
      VerticalOptions="EndAndExpand">

    <StackLayout Orientation="Horizontal"
                  Spacing="16">

        <Image Source="{DynamicResource ic_logout}"
              VerticalOptions="CenterAndExpand"/>

        <Label Style="{DynamicResource tstyle_i4_1}"
              Text="{xct:Translate LogOut}"
              VerticalOptions="CenterAndExpand"/>

    </StackLayout>

</Frame>

</StackLayout>

</pages:BaseContentPage>

</MasterDetailPage.Master>

</MasterDetailPage>

```

NewRequestPage.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<pages:BaseContentPage xmlns:pages="clr-namespace:TeamConnect.Views"
                        xmlns="http://xamarin.com/schemas/2014/forms"

```

```

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
xmlns:viewmodels="clr-
namespace:TeamConnect.ViewModels"
    x:Class="TeamConnect.Views.NewRequestPage"
    x:DataType="viewmodels.NewRequestPageViewModel"
    BackgroundColor="{DynamicResource tcolor_i3}"
    Title="{xct:Translate NewRequest}"
<StackLayout Padding="24, 34, 24, 24"
    Spacing="16">

    <Label Style="{DynamicResource tstyle_i1_1}"
        Text="{xct:Translate RequestType}"/>

    <Frame HasShadow="False"
        Padding="16, 18"
        CornerRadius="16"
        BorderColor="{DynamicResource appcolor_i2}"
        BackgroundColor="Transparent">

        <Frame.GestureRecognizers>

            <TapGestureRecognizer Command="{Binding
SelectRequestTypeCommand}"/>

        </Frame.GestureRecognizers>

        <Label Text="{xct:Translate SelectType}"
            Style="{DynamicResource tstyle_i2_1}">

            <Label.Triggers>

                <DataTrigger TargetType="Label"
                    Binding="{Binding SelectedRequestType,
Converter={StaticResource isNotNullOrEmptyConverter}}"
                    Value="True">

                    <Setter Property="Text" Value="{Binding
SelectedRequestType}"/>

                    <Setter Property="TextColor" Value="{DynamicResource
tcolor_i1}"/>

                </DataTrigger>

            </Label.Triggers>

        </Label>

    </Frame>

    <Label Style="{DynamicResource tstyle_i1_1}"
        Text="{xct:Translate Duration}"
        Margin="0, 34, 0, 0"/>

    <Frame HasShadow="False"
        Padding="16, 18"
        CornerRadius="16"
        BorderColor="{DynamicResource appcolor_i2}"
        BackgroundColor="Transparent">

        <Frame.GestureRecognizers>

```

```

        <TapGestureRecognizer                                Command="{Binding
SelectStartDateCommand}"/>
    </Frame.GestureRecognizers>

    <Label Text="{xct:Translate From}"
        Style="{DynamicResource tstyle_i2_1}">

        <Label.Triggers>

            <DataTrigger TargetType="Label"
                Binding="{Binding IsStartDateChanged}"
                Value="True">

                <Setter Property="Text" Value="{Binding StartDate,
StringFormat='{0:dd MMM yyyy}'}"/>

                <Setter Property="TextColor" Value="{DynamicResource
tcolor_i1}"/>

            </DataTrigger>

        </Label.Triggers>

    </Label>

</Frame>

<Frame HasShadow="False"
    Padding="16, 18"
    CornerRadius="16"
    BorderColor="{DynamicResource appcolor_i2}"
    BackgroundColor="Transparent">

    <Frame.GestureRecognizers>
        <TapGestureRecognizer                                Command="{Binding
SelectEndDateCommand}"/>
    </Frame.GestureRecognizers>

    <Label Text="{xct:Translate To}"
        Style="{DynamicResource tstyle_i2_1}">

        <Label.Triggers>

            <DataTrigger TargetType="Label"
                Binding="{Binding IsEndDateChanged}"
                Value="True">

                <Setter Property="Text" Value="{Binding EndDate,
StringFormat='{0:dd MMM yyyy}'}"/>

                <Setter Property="TextColor" Value="{DynamicResource
tcolor_i1}"/>

            </DataTrigger>

        </Label.Triggers>

    </Label>

</Frame>

<Button Style="{DynamicResource bstyle_i3}"

```

```

        Text="{xct:Translate CreateRequest}"
        Command="{Binding CreateRequestTapCommand}"
        InputTransparent="True"
        VerticalOptions="EndAndExpand">

<Button.Triggers>

    <Trigger TargetType="Button"
        Property="InputTransparent"
        Value="False">

        <Setter      Property="Style"      Value="{DynamicResource
bstyle_i1}"/>

    </Trigger>

    <MultiTrigger TargetType="Button">

        <MultiTrigger.Conditions>

            <BindingCondition Binding="{Binding
SelectedRequestType, Converter={StaticResource isNotNullOrEmptyConverter}}"
                Value="True"/>

            <BindingCondition Binding="{Binding
IsStartDateChanged}"
                Value="True"/>

            <BindingCondition Binding="{Binding
IsEndDateChanged}"
                Value="True"/>

        </MultiTrigger.Conditions>

        <Setter Property="InputTransparent" Value="False"/>

    </MultiTrigger>

</Button.Triggers>

</Button>

</StackLayout>

</pages:BaseContentPage>

```

TeamPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<page:BaseDetailPage xmlns:page="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:viewmodels="clr-namespace:TeamConnect.ViewModels"
    xmlns:datatemplates="clr-
namespace:TeamConnect.Views.DataTemplates"
    x:Class="TeamConnect.Views.TeamPage"
    x:DataType="viewmodels:TeamPageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"
    Title="{xct:Translate Team}">

```

```

<Grid RowDefinitions="Auto, *"
      RowSpacing="0">

    <StackLayout Grid.Row="0"
                  HeightRequest="60"
                  Padding="24, 0">

        <Label Style="{DynamicResource tstyle_i1_3}"
                Text="{Binding Team.Name}"
                VerticalOptions="CenterAndExpand"/>

    </StackLayout>

    <CollectionView Grid.Row="1"
                    ItemsSource="{Binding Users}">

        <CollectionView.ItemsLayout>

            <LinearItemsLayout Orientation="Vertical"
                               ItemSpacing="10"/>

        </CollectionView.ItemsLayout>

        <CollectionView.Header>

            <StackLayout HeightRequest="{OnPlatform iOS=20}"/>

        </CollectionView.Header>

        <CollectionView.ItemTemplate>

            <datatemplates:TeamDataTemplate/>

        </CollectionView.ItemTemplate>

        <CollectionView.Footer>

            <StackLayout HeightRequest="20"/>

        </CollectionView.Footer>

    </CollectionView>

    <ActivityIndicator Style="{DynamicResource activityIndicator}"
                        IsVisible="False">

        <ActivityIndicator.Triggers>

            <DataTrigger TargetType="ActivityIndicator"
                          Binding="{Binding PageState}"
                          Value="Loading">

                <Setter Property="IsVisible" Value="True"/>

            </DataTrigger>

        </ActivityIndicator.Triggers>

    </ActivityIndicator>

    <StackLayout Grid.Row="0"
                  Grid.RowSpan="2">

```

```

        HorizontalOptions="Center"
        VerticalOptions="Center">

<Label Style="{DynamicResource tstyle_i1_2}"
    Text="{xct:Translate NoTeam}"
    HorizontalOptions="CenterAndExpand"
    IsVisible="False">

    <Label.Triggers>

        <DataTrigger TargetType="Label"
            Binding="{Binding PageState}"
            Value="NoResult">

            <Setter Property="IsVisible" Value="True"/>

        </DataTrigger>

        <DataTrigger TargetType="Label"
            Binding="{Binding PageState}"
            Value="TeamNotCreated">

            <Setter Property="IsVisible" Value="True"/>

        </DataTrigger>

    </Label.Triggers>

</Label>

<Button Style="{DynamicResource bstyle_i1}"
    Text="{xct:Translate CreateNewTeam}"
    Command="{Binding CreateTeamTapCommand}"
    IsVisible="False"
    HorizontalOptions="CenterAndExpand"
    Padding="10, 0">

    <Button.Triggers>

        <DataTrigger TargetType="Button"
            Binding="{Binding PageState}"
            Value="TeamNotCreated">

            <Setter Property="IsVisible" Value="True"/>

        </DataTrigger>

    </Button.Triggers>

</Button>

</StackLayout>

<Button Grid.Row="1"
    WidthRequest="64"
    HeightRequest="64"
    ImageSource="{DynamicResource ic_plus}"
    CornerRadius="32"
    BackgroundColor="Blue"
    HorizontalOptions="End"
    VerticalOptions="End"
    Margin="24"

```

```

        Command="{Binding AddMemberTapCommand}">

        <Button.Triggers>

            <DataTrigger TargetType="Button"
                Binding="{Binding PageState}"
                Value="TeamNotCreated">

                <Setter Property="IsVisible" Value="False"/>
            </DataTrigger>

        </Button.Triggers>

    </Button>

</Grid>

</page:BaseDetailPage>

```

TeamTimePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<pages:BaseDetailPage xmlns:pages="clr-namespace:TeamConnect.Views"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:viewmodels="clr-namespace:TeamConnect.ViewModels"
    xmlns:datatemplates="clr-namespace:TeamConnect.Views.DataTemplates"
    x:Class="TeamConnect.Views.TeamTimePage"
    x:DataType="viewmodels:TeamTimePageViewModel"
    BackgroundColor="{DynamicResource appcolor_i4}"
    Title="{xct:Translate TeamTime}">

    <pages:BaseDetailPage.ToolbarItems>

        <ToolbarItem IconImageSource="{DynamicResource ic_calendar}"
            Command="{Binding SelectDateTapCommand}"/>

    </pages:BaseDetailPage.ToolbarItems>

    <Grid RowDefinitions="Auto, *"
        RowSpacing="0">

        <!-- DATE -->
        <StackLayout Grid.Row="0"
            HeightRequest="60"
            Padding="24, 0">

            <Label Style="{DynamicResource tstyle_i1_3}"
                VerticalOptions="CenterAndExpand">

                <Label.FormattedText>

                    <FormattedString>

                        <Span Text="{Binding Date,
StringFormat='{0:dddd}'}"/>

                        <Span Text=", "/>
                        <Span Text="{Binding Date, StringFormat='{0:dd MMM
yyyy}'}"/>

```

```

        </FormattedString>

    </Label.FormattedText>

</Label>

</StackLayout>

<!-- TEAM TIMES -->
<CollectionView Grid.Row="1"
    ItemsSource="{Binding TeamTimes}"
    IsGrouped="True">

    <CollectionView.ItemsLayout>

        <LinearItemsLayout Orientation="Vertical"
            ItemSpacing="10"/>

    </CollectionView.ItemsLayout>

    <CollectionView.GroupHeaderTemplate>

        <datatemplates:TeamTimeGroupDataTemplate/>

    </CollectionView.GroupHeaderTemplate>

    <CollectionView.ItemTemplate>

        <datatemplates:TeamTimeDataTemplate/>

    </CollectionView.ItemTemplate>

    <CollectionView.Footer>

        <StackLayout HeightRequest="20"/>

    </CollectionView.Footer>

</CollectionView>

<ActivityIndicator Grid.Row="1"
    Style="{DynamicResource activityIndicator}"
    IsVisible="False">

    <ActivityIndicator.Triggers>

        <DataTrigger TargetType="ActivityIndicator"
            Binding="{Binding PageState}"
            Value="Loading">
            <Setter Property="IsVisible" Value="True"/>
        </DataTrigger>

    </ActivityIndicator.Triggers>

</ActivityIndicator>

<Label Grid.Row="1"
    Style="{DynamicResource tstyle_i1_2}"
    Text="{xct:Translate NoTeamTime}"
    HorizontalOptions="Center"
    VerticalOptions="Center"

```

```

        IsVisible="False">

        <Label.Triggers>

            <DataTrigger TargetType="Label"
                Binding="{Binding PageState}"
                Value="NoResult">

                <Setter Property="IsVisible" Value="True"/>

            </DataTrigger>

        </Label.Triggers>

    </Label>

</Grid>

</pages:BaseDetailPage>

```

App.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<prism:PrismApplication                                xmlns:prism="clr-
namespace:Prism.Unity;assembly=Prism.Unity.Forms"
                                xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                                xmlns:styles="clr-
namespace:TeamConnect.Resources.Styles"
                                x:Class="TeamConnect.App">

    <prism:PrismApplication.Resources>
        <styles:DefaultResources>

            <styles:DefaultResources.MergedDictionaries>

                <styles:LightModeResources/>

            </styles:DefaultResources.MergedDictionaries>

        </styles:DefaultResources>

    </prism:PrismApplication.Resources>

</prism:PrismApplication>

```

App.xaml.cs

```

namespace TeamConnect
{
    public partial class App : PrismApplication
    {
        private IAuthorizationService _authorizationService;
        private IAuthorizationService AuthorizationService =>
        _authorizationService ??= Container.Resolve<IAuthorizationService>();

        public App()
        {

```

```

    }

    #region -- Overrides --

    protected override void OnInitialized()
    {
        InitializeComponent();

        InitializeLocalizationManager();

        if (AuthorizationService.IsAuthorized)
        {
            NavigationService.NavigateAsync($"{nameof(MainMasterPage)}/{nameof(NavigationPage)}/{nameof(TeamPage)}");
        }
        else
        {
            NavigationService.NavigateAsync(
                $"{nameof(NavigationPage)}/{nameof(LoginPage)}");
        }
    }

    protected override void RegisterTypes(IContainerRegistry
containerRegistry)
    {
        // Packages
        containerRegistry.RegisterInstance(UserDialogs.Instance);
        containerRegistry.RegisterPopupNavigationService();

        // Services

        containerRegistry.RegisterInstance<IRepository>(Container.Resolve<Repository>());

        containerRegistry.RegisterInstance<ISettingsManager>(Container.Resolve<SettingsManage
r>());

        containerRegistry.RegisterInstance<IRestService>(Container.Resolve<RestService>());

        containerRegistry.RegisterInstance<IAuthorizationService>(Container.Resolve<Authoriza
tionService>());

        containerRegistry.RegisterInstance<IUserService>(Container.Resolve<UserService>());

        containerRegistry.RegisterInstance<ITeamService>(Container.Resolve<TeamService>());

        containerRegistry.RegisterInstance<ILeaveService>(Container.Resolve<LeaveService>());

        containerRegistry.RegisterInstance<IMapService>(Container.Resolve<MapService>());

        // Pages
        containerRegistry.RegisterForNavigation<NavigationPage>();
        containerRegistry.RegisterForNavigation<LoginPage,
LoginPageViewModel>();
        containerRegistry.RegisterForNavigation<CreateAccountFirstPage,
CreateAccountFirstPageViewModel>();
        containerRegistry.RegisterForNavigation
<CreateAccountSecondPage, CreateAccountSecondPageViewModel>();

        containerRegistry.RegisterForNavigation<CompleteRegistrationFirstPage,
CompleteRegistrationFirstPageViewModel>();

```

```

        containerRegistry.RegisterForNavigation<CompleteRegistrationSecondPage,
CompleteRegistrationSecondPageViewModel>();
        containerRegistry.RegisterForNavigation<MainMasterPage,
MainMasterPageViewModel>();
        containerRegistry.RegisterForNavigation<TeamPage,
TeamPageViewModel>();
        containerRegistry.RegisterForNavigation<CreateTeamPopupPage,
CreateTeamPopupPageViewModel>();
        containerRegistry.RegisterForNavigation<AddMembersPopupPage,
AddMembersPopupPageViewModel>();
        containerRegistry.RegisterForNavigation<LeavePage,
LeavePageViewModel>();
        containerRegistry.RegisterForNavigation<NewRequestPage,
NewRequestPageViewModel>();
        containerRegistry.RegisterForNavigation<TeamTimePage,
TeamTimePageViewModel>();
    }

    protected override void OnStart()
    {
    }

    protected override void OnSleep()
    {
    }

    protected override void OnResume()
    {
    }

    #endregion

    #region -- Private helpers --

    private void InitializeLocalizationManager()
    {
        var enCulture = new CultureInfo("en");

        CultureInfo.DefaultThreadCurrentCulture = enCulture;
        CultureInfo.DefaultThreadCurrentUICulture = enCulture;
        CultureInfo.CurrentCulture = enCulture;
        CultureInfo.CurrentUICulture = enCulture;

        LocalizationResourceManager.Current.PropertyChanged += (sender,
e) => Strings.Culture = LocalizationResourceManager.Current.CurrentCulture;

        LocalizationResourceManager.Current.Init(Strings.ResourceManager);
        LocalizationResourceManager.Current.CurrentCulture = enCulture;
    }

    #endregion
}

```

Constants.cs

```

namespace TeamConnect
{
    public static class Constants
    {
        public const string DATABASE_NAME = "teamconnect.db3";
    }
}

```

```

public const int DEFAULT_ID = 0;
public const int DEFAULT_WORKING_TIME = 8;

public static class Navigation
{
    public const string USER = nameof(USER);
    public const string EMAIL = nameof(EMAIL);
    public const string DETAIL_PAGE = nameof(DETAIL_PAGE);
    public const string MEMBER_NOT_ADDED = nameof(MEMBER_NOT_ADDED);
}

public static class GoogleAPI
{
    public const string API_KEY = "AIzaSyDCu7o1IVRZysBWkCR95qIZH52KL-
1okhM";
    public const string BASE_TIMEZONE_API_URI =
"https://maps.googleapis.com/maps/api/timezone/json";
    public const string LOCATION = "?location=";
    public const string TIMESTAMP = "&timestamp=";
    public const string KEY = "&key=";
}
}

```

BorderlessEntryRenderer.cs

```

[assembly: ExportRenderer(typeof(BorderlessEntry),
typeof(BorderlessEntryRenderer))]
namespace TeamConnect.Droid.Renderers
{
    public class BorderlessEntryRenderer : EntryRenderer
    {
        public BorderlessEntryRenderer(Context context)
            : base(context)
        {
        }

        #region -- Overrides --

        protected override void
        OnElementChanged(ElementChangedEventArgs<Entry> e)
        {
            base.OnElementChanged(e);

            if (Control != null)
            {
                Control.SetPadding(0, 0, 0, 0);
                Control.Background = null;
            }
        }

        #endregion
    }
}

```

BorderlessEntryRenderer.cs

```

[assembly: ExportRenderer(typeof(BorderlessEntry),
typeof(BorderlessEntryRenderer))]

```

```

namespace TeamConnect.iOS.Renderers
{
    public class BorderlessEntryRenderer : EntryRenderer
    {
        #region -- Overrides --
        protected override void
OnElementChanged(ElementChangedEventArgs<Entry> e)
        {
            base.OnElementChanged(e);

            if (Control != null)
            {
                Control.Layer.BorderWidth = 0;

                var color =
(Color)Xamarin.Forms.Application.Current.Resources["appcolor_i1"];
                Control.TintColor = color.ToUIColor();
                Control.BorderStyle = UITextBorderStyle.None;
            }
        }
    }
}
#endregion
}

```

ДОДАТОК Б
Слайди презентації

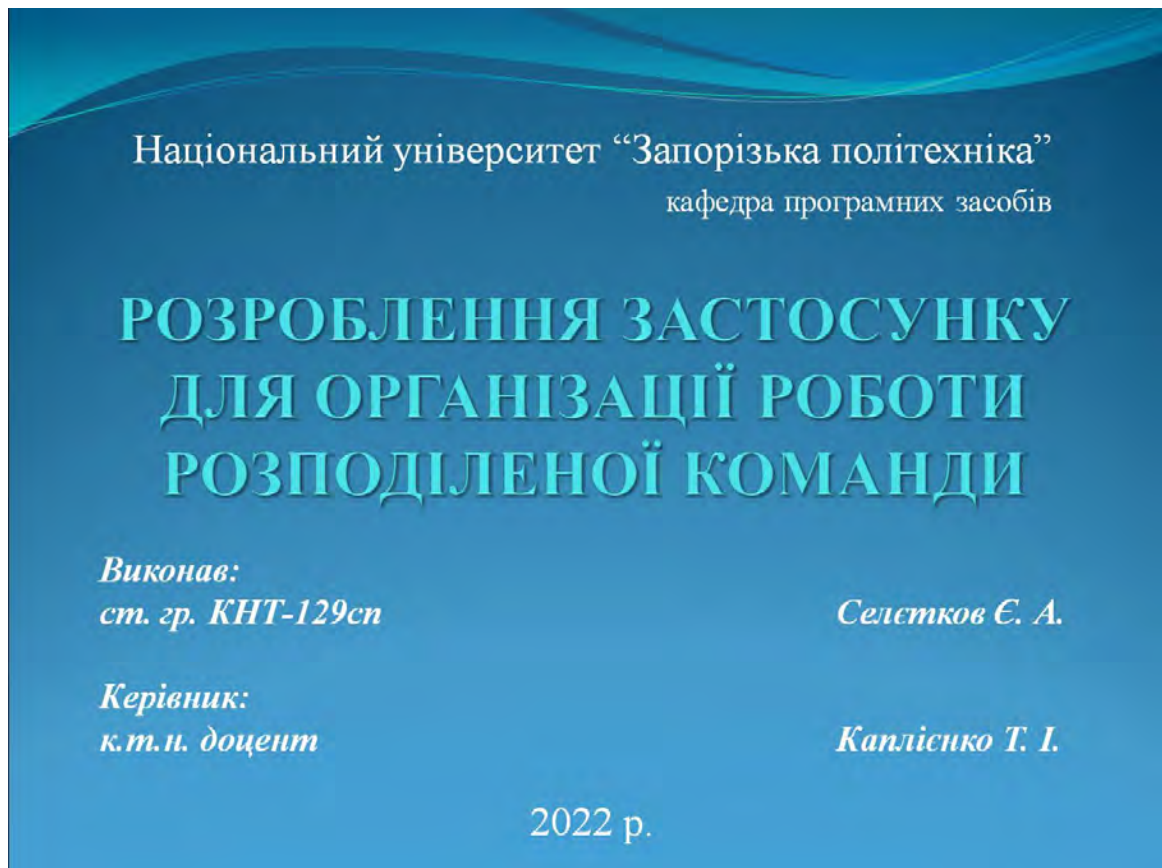


Рисунок Б.1 – Слайд 1 презентації

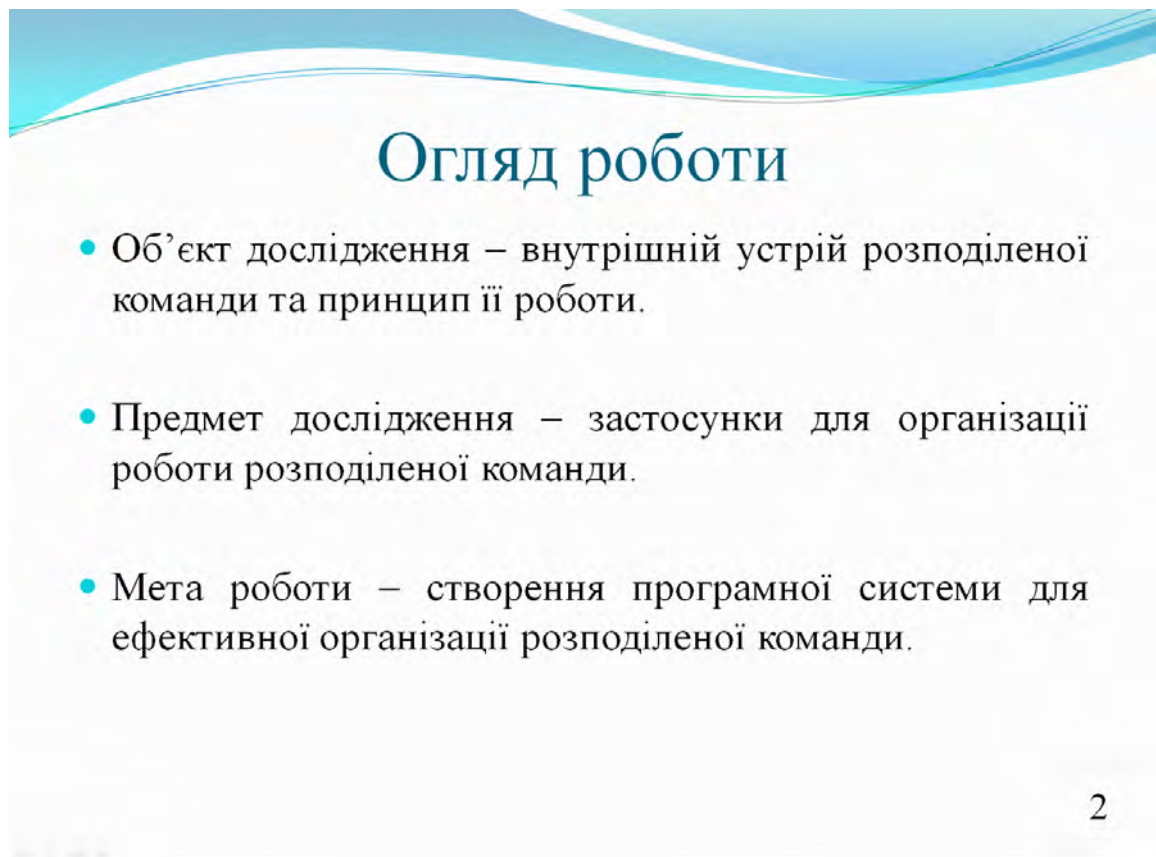


Рисунок Б.2 – Слайд 2 презентації

Аналіз предметної області

Віддалена робота та використання територіально розподілених проєктних команд в сучасних умовах є життєвою необхідністю. Особливо це стосується сфери ІТ-аутсорсингу.

- Територіально розподілена команда – це група співробітників компанії, які виділені для певного проєкту і від сумісної роботи яких залежить результат проєкту. Між собою співробітники, як однієї команди, так і різних, здійснюють комунікації шляхом спілкування за допомогою інформаційних технологій. При цьому вони можуть знаходитися як в одному будинку, так і за тисячі кілометрів один від одного.

Майже все, що стосується роботи та проєкту знаходиться у віртуальному просторі, а фізичний офіс є лише допоміжним пунктом. Тому для успішної роботи головним є якість комунікацій та взаємодії учасників команд.

3

Рисунок Б.3 – Слайд 3 презентації

Огляд існуючих аналогів

Розглянемо деякі програмні засоби для організації роботи розподіленої команди.

- Jira - це потужний інструмент управління роботою, що підходить для різних випадків, від управління вимогами і сценаріями тестування, до agile-розробки програмного забезпечення.
- Trello – це візуальний інструмент керування робочим процесом, який дає змогу командам формувати ідеї, планувати роботу та організовано й продуктивно керувати нею.
- Slack – корпоративний месенджер.
- GitHub – це система управління проєктами та версіями коду, а також платформа соціальних мереж, створена для розробників.
- Hurma (скорочено від Human Resource Management) – продукт для автоматизації задач HR-відділу.

4

Рисунок Б.4 – Слайд 4 презентації

Порівняння мов програмування

Порівняння Kotlin, Java, Objective C та C#

Мови програмування	ОС	Кросплатформовість	ОС для розробки	Простота вивчення
Kotlin	Android та iOS	+	Windows, Linux, MacOS	Просто вчити
Java	Android	-	Windows, Linux, MacOS	Відносно просто вчити
Swift	iOS	-	MacOS	Просто вчити
Objective C	iOS	-	MacOS	Складно вчити
C#	Android та iOS	+	Windows, Linux, MacOS	Просто вчити

5

Рисунок Б.5 – Слайд 5 презентації

Порівняння кросплатформових фреймворків

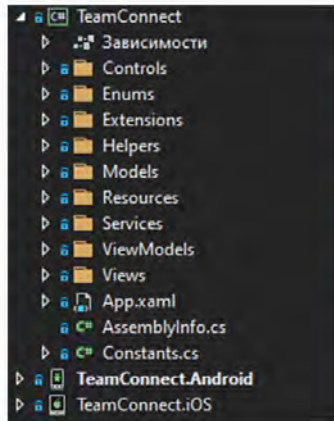
Порівняння Xamarin, Flutter, React Native

Фреймворк	Популярність	Платформи	Можливі архітектурні шаблони
Xamarin	Не дуже популярний	Android, iOS, Windows, MacOS	MVVM, MVC
Flutter	Дуже популярний	Android, iOS	MVVM, MVC, MVP, Redux
React Native	Популярний	Android, iOS, Windows, MacOS, вебзастосунки	MVVM, MVC, MVP, MVI, Redux

6

Рисунок Б.6 – Слайд 6 презентації

Структура проекту



Для реалізації застосунку було обрано шаблон проектування MVVM. Програмний код умовно ділиться на три частини:

- View містить інтерфейс користувача. У Xamarin.Forms представлений у вигляді коду на мові розмітки XAML;
- Model – класи з даними, а також логіка пов'язана з ними;
- ViewModel використовується для зв'язування Model та View. Має логіку для отримання даних із Model та передавання їх у View, а також оновлення даних у Model при взаємодії з View.

7

Рисунок Б.7 – Слайд 7 презентації

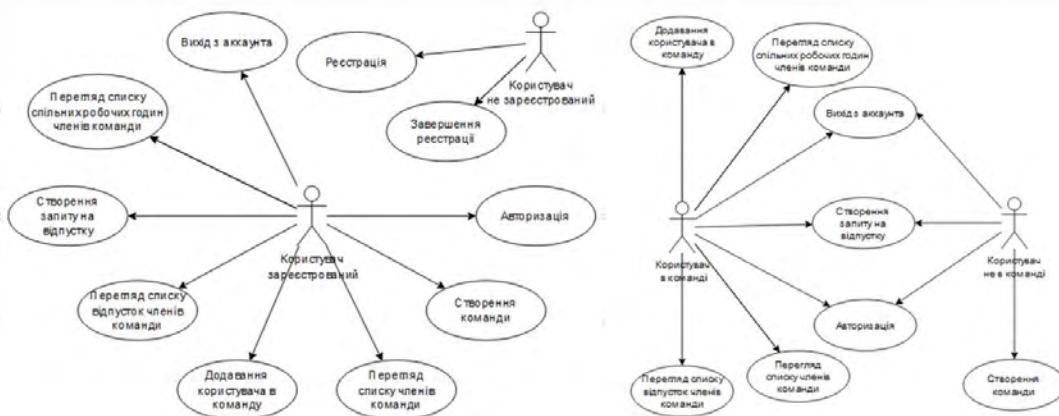
Підключені бібліотеки та фреймворки

- Prism.Unity.Forms – MVVM, DI та зручна навігація;
- sqlite-net-pcl – робота з базою даних SQLite;
- Newtonsoft.Json – робота з json;
- Xamarin.Forms.Maps – робота з картою Apple для iOS та картою Google для Android;
- Nager.Date – визначення святкових днів та вихідних для певної країни;
- Xamarin.CommunityToolkit – різні інструменти для спрощення розробки на Xamarin.Forms;
- Xamarin.FFImageLoading та Xamarin.FFImageLoading.Forms – елемент керування для взаємодії з зображеннями;
- Acr.UserDialogs – стандартні для Android, iOS та Windows спливаючі вікна;
- Rg.Plugin.Popup – створення власних спливаючих сторінок;
- Prism.Plugin.Popups – взаємодія Rg.Plugin.Popup з навігацією Prism.Unity.Forms.

8

Рисунок Б.8 – Слайд 8 презентації

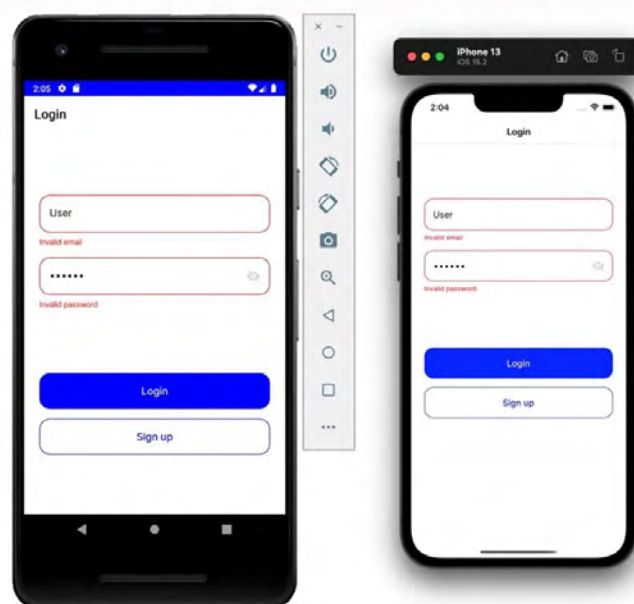
Use-case діаграма користувача



9

Рисунок Б.9 – Слайд 9 презентації

Сторінка входу



10

Рисунок Б.10 – Слайд 10 презентації

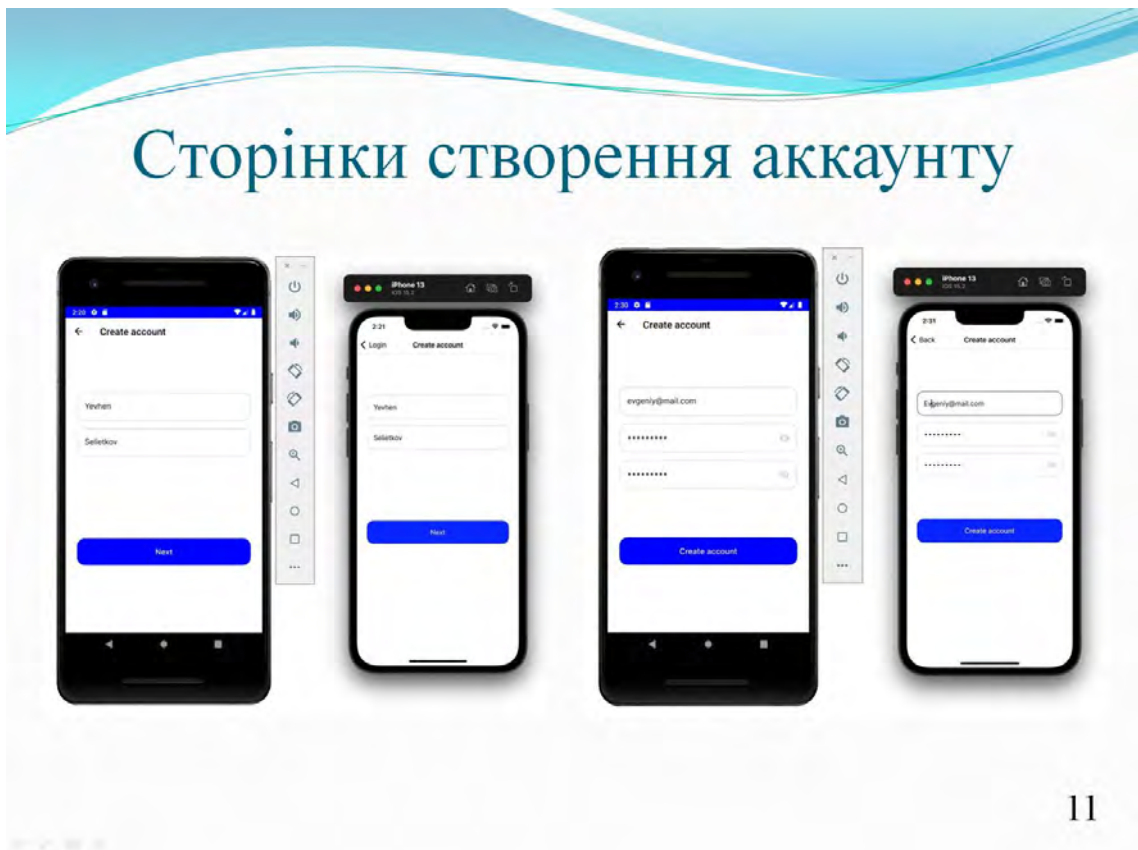


Рисунок Б.11 – Слайд 11 презентації

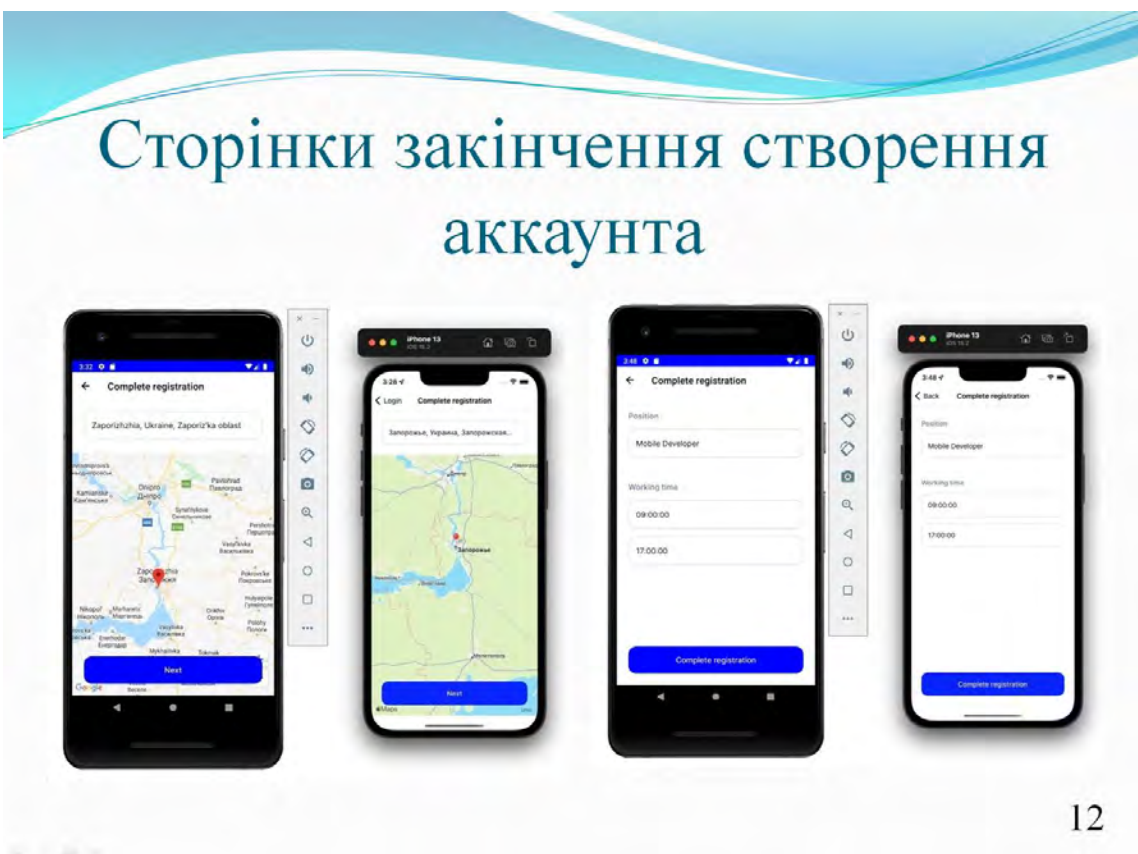


Рисунок Б.12 – Слайд 12 презентації

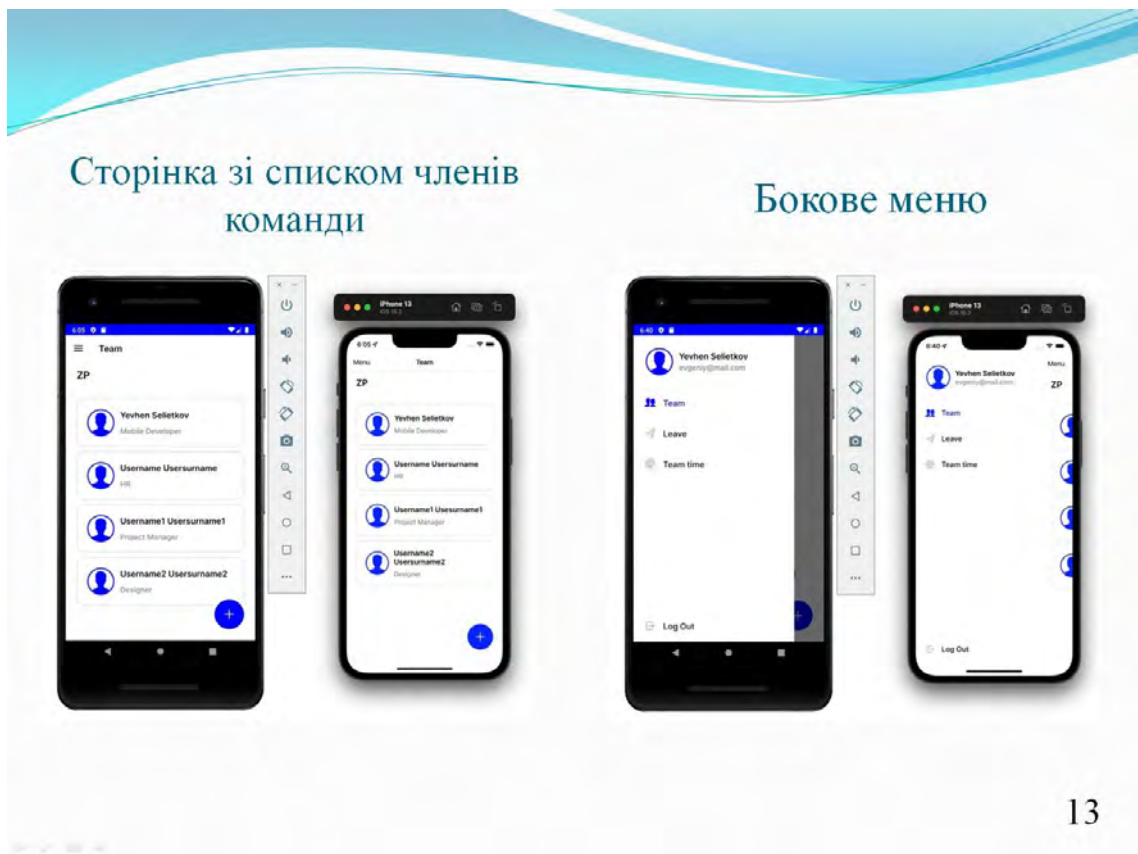


Рисунок Б.13 – Слайд 13 презентації

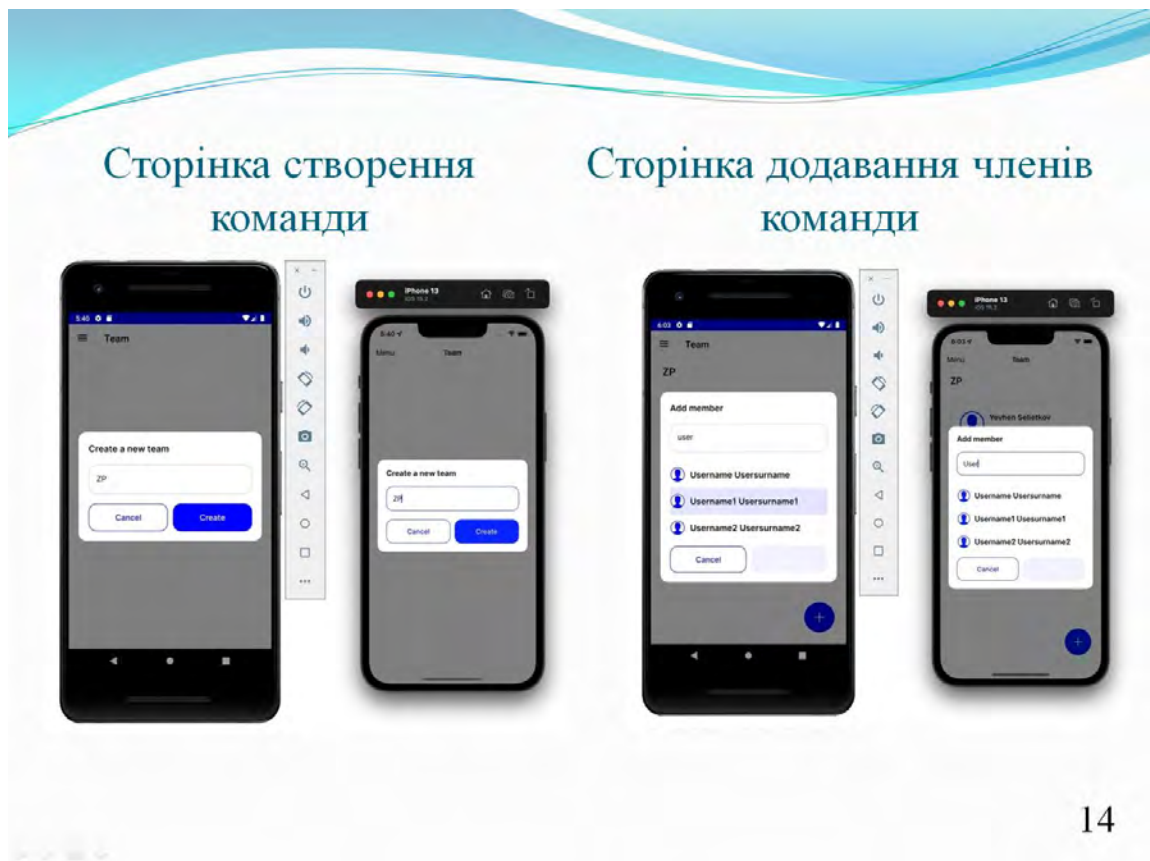


Рисунок Б.14 – Слайд 14 презентації

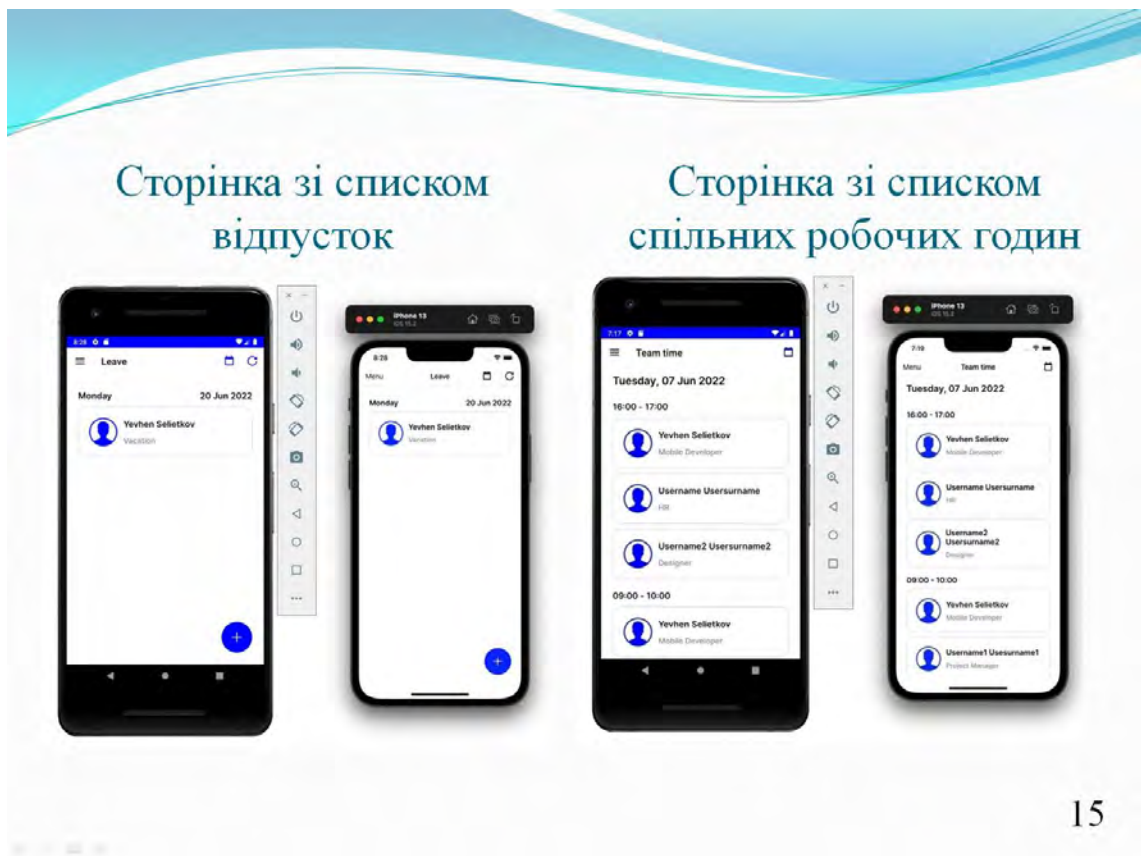


Рисунок Б.15 – Слайд 15 презентації

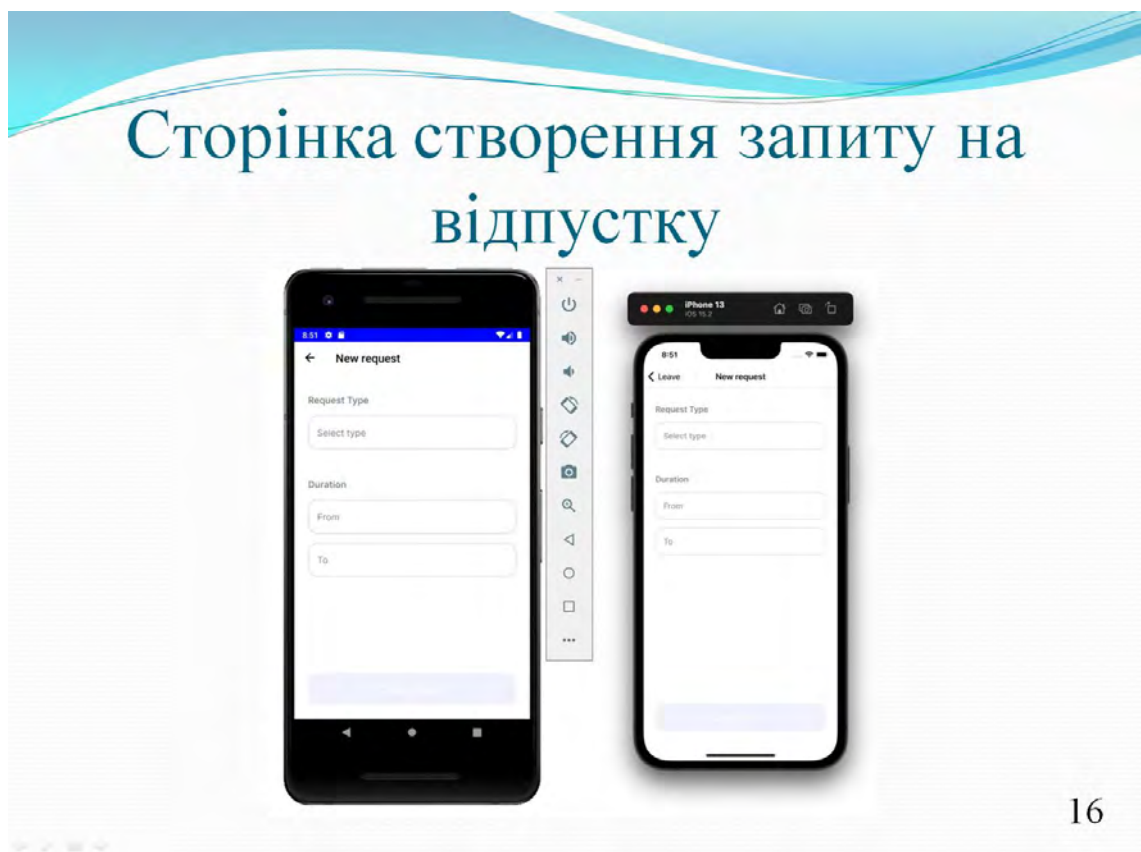


Рисунок Б.16 – Слайд 16 презентації

Таблиця тестування функціоналу

Функції	Зареєстрований користувач	Не зареєстрований користувач	Користувач не в команді	Користувач в команді
Реєстрація	-	+	-	-
Завершення реєстрації	-	+	-	-
Авторизація	+	-	+	+
Створення команди	+	-	+	-
Перегляд списку членів команди	+	-	-	+
Додавання користувача в команду	+	-	-	+
Перегляд списку відпусток членів команди	+	-	+	+
Створення запиту на відпустку	+	-	-	+
Перегляд списку спільних робочих годин членів команди	+	-	+	+
Вихід з аккаунта	+	-	+	+

17

Рисунок Б.17 – Слайд 17 презентації

Висновки

Для розробки застосунку був обраний архітектурний шаблон проектування MVVM, як найпопулярніший для розробки мобільних застосунків.

Під час написання коду було створено 4 моделі, 8 сервісів, 12 сторінок та стільки ж вьюмоделей, які відповідають сторінкам.

Були проведені всі тести функціоналу для кожного типу користувача. Всі тести були пройдені успішно.

Створений застосунок планується розвивати й далі. Список функціоналу, який планується реалізувати:

- серверна частина застосунку;
- можливість створення декількох команд;
- новий тип користувача адміністратор;
- можливість надати права адміністратора користувачу;
- можливість підтверджувати та відмовляти в запиті на відпустку адміністратором;
- можливість підтверджувати вступ у команду;
- можливість редагувати аккаунт.

18

Рисунок Б.18 – Слайд 18 презентації