РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: мобільний додаток - органайзер досягнень.

Мета кваліфікаційної роботи: розробка та створення програмного функціоналу, що надає змогу користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, виконано проєктування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні програмного додатка, що надає можливість користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання. Така взаємодія гаджету з власником смартфону дозволить користувачу контролювати саморозвиток і постійно вдосконалюватись.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки для смарт гаджетів, за допомогою яких користувач (людина) може спостерігати і контролювати стан свого життя, встановлювати різноманітні цілі та задачі та одержувати оцінку в результаті їх виконання, чи не виконання.

Список ключових слів: МОБІЛЬНИЙ ДОДАТОК, ПРОГРАМА, ОРГАНАЙЗЕР, ІНТЕРФЕЙС, ПРОГРАМУВАННЯ, БАЗА ДАНИХ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: mobile application - achievement organizer.

The purpose of the qualification work: development and creation of software functionality that allows the user to set (create) their tasks (goals) and determine the deadline.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, designs and develops the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of a software application that allows the user to set (create) their tasks (goals) and determine the deadline. This interaction of the gadget with the owner of the smartphone will allow the user to control self-development and constantly improve.

The relevance of this software product is determined by the high demand for such developments for smart gadgets, with which the user (person) can monitor and control the state of his life, set various goals and objectives and get an assessment of their performance or failure.

List of keywords: MOBILE APPLICATION, PROGRAM, ORGANIZER, INTERFACE, PROGRAMMING, DATABASE.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – бази даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СКБД – система керування базами даних;

ADT – Android Development Tools, набір інструментів для розробки під Android;

MVC – Model-View-Controller, Модель-Вид-Контролер.

ЗМІСТ

ВСТУП

Починаючи з кінця XX століття розвивається індустрія гаджетів. Кожного року на ринок технологій виходять все більш сильніші і функціональні смартфони, що дають змогу розробникам мобільних програм втілювати свої самі креативні ідеї. Ринок технологій розвивається, бо потреби людини ростуть. Людство за допомогою гаджетів може спостерігати за своїм психологічним і фізичним станом.

Для людей важливо постійно самовдосконалюватися. Важливою складовою саморозвитку є постановка цілей. Якщо не контролювати і не оцінювати дії людини, то це не буде призводити до покращення її вмінь і не буде наближати до поставленої цілі. Самовдосконалення важливо для покращення психологічного настрою людини і відтягування старіння мозку.

За допомогою смарт гаджетів користувач(людина) може спостерігати і контролювати стан свого життя. Нещодавно стали популярними носимі гаджети, а саме, різні фітнес трекери, що дозволяють стежити за фізичною активністю людей. За допомогою смартфона і мобільних додатків людина може встановлювати різноманітні цілі та задачі і одержувати оцінку в результаті їх виконання, чи не виконання. Така взаємодія з власником смартфона дозволить контролювати саморозвиток і постійно вдосконалюватись.

При виборі смартфона потрібно спиратися на характеристики гаджету. Одною з ключових характеристик є операційна система. На ринку мобільних гаджетів є дві конкуруючих операційних системи – це Android від компанії Google і IOS від компанії Apple. Більшу частину ринку займає операційна система Android.

Одним з найкращий фреймворків для розробки мобільних додатків є Xamarin від корпорації Microsoft, що має багато влаштованих бібліотек і добру взаємодію з окремими модулями смартфона. Xamarin створений на базі проекта Mono і має повну підтримку SQLLite для роботи з базами даних.

Завданням кваліфікаційної роботи є проектування та створення мобільного додатку - органайзеру досягнень «The Motivator», розробленого за допомогою технології Xamarin мовою програмування C# у IDA MS Visual Studio 2019.

Метою розробки мобільного додатку є розробка та створення програмного функціоналу, що надає змогу користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання.

Головною особливістю додатку є наявність статистики виконаних чи не виконаних задач та групові кімнати. Статистика розробленого програмного додатку показує успішність людини (користувача) та її витрачені зусилля на досягнення конкретних цілей.

В розробленому додатку реалізовано:

– особистий кабінет користувача, де відображаються отримані повідомлення та запрошення інших користувачів програми;

– створення здач, де користувач може ставити заплановані задачі;

– формування статистики, яка дозволяє користувачеві контролювати себе і стежити за своїм просуванням до цілі;

– спільні завдання і цілі для команди користувачів у групових кімнатах, які можна ставити для групового виконання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

У XXI сторіччі у всьому світі відбувається активний розвиток інформаційних технологій. Серед немалої кількості винаходів і нових можливостей значну популярність здобуває Інтернет. З'являються сервіси, сайти, веб-додатки, що покращують і полегшують життя людей у багатьох його аспектах.

Великий крок вперед у спілкуванні між людьми робить створення програмного забезпечення для персональних комп'ютерів, додатків для смартфонів, сайтів в Інтернеті, що дозволяють своїм користувачам обмінюватися текстовими, голосовими повідомленнями, відео-повідомленнями, файлами різних типів і форматів. Додатки з вищезазначеною функціональністю називають месенджерами.

На сьогодні існує декілька найпопулярніших додатків такого виду, а саме Telegram, Viber, WhatsApp. Кожен з них володіє дуже великою кількістю технологій, можливостей, унікальних технічних і дизайнерських рішень. Користувачі даних месенджерів, окрім вищеперелічених особливостей, мають змогу робити звичайні голосові дзвінки, що реалізовуються за допомогою Інтернет-технологій, а не стільникового зв'язку, а також відео-дзвінки:

1. Viber. Ізраїльський стартап з білоруським корінням створювався з думкою замінити Skype. Заснований на російський технологіях, Viber спочатку з'явився в Ізраїлі. Пункти розробки додатків розташовані в Білорусі, головний офіс - у Люксембурзі. У 2014 році стартап викупила японська компанія Rakuten.

Зараз Viber – найпопулярніший месенджер серед українців. За даними українського представництва компанії, месенджером користуються понад 20 млн осіб, тобто половина населення країни.

Однак назвати Viber найбільш безпечним месенджером важко. Було

багато прикладів, які це доводили. Так, у 2013 році фахівці з кібернетичної безпеки виявили, що за допомогою Viber можна отримати доступ до заблокованого смартфона на Android.

Ситуація поліпшилася у 2016 році, коли Viber ввів наскрізне шифрування повідомлень і розмов. Це спосіб передавання даних, при якому доступ до повідомлень мають тільки користувачі, задіяні у спілкуванні.

У 2017 році в месенджері з'явилися приховані чати, доступ до яких можна отримати за кодом, можливість писати повідомлення, які самі видаляються, а також контроль за спробами створення знімків листування.

Однак не варто забувати, що сервери цієї компанії розташовані в Росії. Хоча в компанії стверджують, що на них лежать лише дані російських користувачів, цю інформацію перевірити неможливо. Гарантій, що дані українських користувачів не зберігаються в тому ж місці, нема.

2. WhatsApp. WhatsApp є найпопулярнішим месенджером у світі, хоча спочатку він навіть не був месенджером. Це була програма, яка показувала статус контактів: хто знаходиться в мережі, хто телефонує, хто зайнятий.

Згодом додаток перетворився на вдосконалену альтернативу sms і перший месенджер, синхронізований з контактами в телефоні. Розробив його український емігрант Ян Кум. Над інтерфейсом та дизайном теж працювали українці: їх розробила команда дизайнерів з Дніпра.

WhatsApp посилив захист у 2016 році після інциденту між Apple та ФБР і запровадив наскрізне шифрування. Для його реалізації використовується бібліотека месенджера Signal, який прийнято вважати найбільш безпечним у світі.

Після цього компанія запевнила, що зміст повідомлень буде доступний тільки відправнику та одержувачу. До цього у неї було багато невдач - починаючи від злому даних і закінчуючи завантаженням вірусів через десктоп-версію додатка. Після впровадження наскрізного шифрування ситуація поліпшилася.

Однак експерти все одно не радять особливо довіряти месенджеру. Він

належить компанії Facebook, яка вирізняється агресивними механізмами отримання інформації про своїх користувачів.

3. Telegram. Про дітище засновника соціальної мережі "Вконтакте" Павла Дурова не чув тільки глухий. Галас навколо месенджера, який з'явився під девізом "Повернемо собі право на приватність", здійнявся саме через питання приватності.

Дуров оголосив конкурс, пообіцявши 200 тисяч доларів тому, хто зламає зашифроване листування. Розшифрувати його нікому не вдалося, проте користувач ресурсу "Хабрахабр" отримав 100 тисяч доларів за знайдену потенційну вразливість у чатах.

У Telegram є два варіанти шифрування: для звичайних і для секретних чатів.

За словами розробників, інформація звичайних чатів зберігається на кількох серверах по всьому світу і контролюється різними законами з надання доступу до неї. Ключі для розшифровки також зберігаються окремими блоками на різних серверах.

Жоден сервер Telegram не перебуває на території Росії, а сам Дуров виїхав з РФ у 2014 році, купивши собі громадянство іншої держави.

Однак лише секретні чати підтримують end-to-end-шифрування, тому чутливу інформацію рекомендується передавати через них.

1.2. Призначення розробки та область застосування

Огляд сучасних аналогів показав, що вони можуть запропонувати користувачам широке різноманіття корисних і зручних технологій, можливостей. Проте жоден з вищеназваних додатків не розглядає повноцінно потребу людей у організації, виокремленні завдань, що вони отримують від інших людей, згадують або придумують у процесі спілкування, що проявляється у відсутності зручних і простих засобів для зберігання певних завдань в них.

Метою розробки мобільного додатку є розробка та створення програмного функціоналу, що надає змогу користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання.

Головною особливістю додатку є наявність статистики виконаних чи не виконаних задач та групові кімнати. Статистика розробленого програмного додатку показує успішність людини (користувача) та її витрачені зусилля на досягнення конкретних цілей.

Практичне значення полягає у створенні програмного додатка, що надає можливість користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання. Така взаємодія гаджету з власником смартфону дозволить користувачу контролювати саморозвиток і постійно вдосконалюватись.

Подібні розробки для смарт гаджетів, за допомогою яких користувач (людина) може спостерігати і контролювати стан свого життя, встановлювати різноманітні цілі та задачі та одержувати оцінку в результаті їх виконання, чи не виконання можуть знайти широкий попит серед активних, сучасних користувачів.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного забезпечення мобільного додатку - органайзеру за допомогою технології Xamarin мовою програмування C# в середовищі розробки MS Visual Studio 2019» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Завданням кваліфікаційної роботи є проектування та створення мобільного додатку - органайзеру досягнень «The Motivator», розробленого за

допомогою технології Xamarin мовою програмування C# у IDA MS Visual Studio 2019.

Метою розробки мобільного додатку є розробка та створення програмного функціоналу, що надає змогу користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання.

Головною особливістю додатку є наявність статистики виконаних чи не виконаних задач та групові кімнати. Статистика розробленого програмного додатку показує успішність людини (користувача) та її витрачені зусилля на досягнення конкретних цілей.

Для досягнення мети роботи необхідно виконати наступні етапи:

1. Спроектувати та розробити мобільний додаток органайзеру досягнень особистих цілей «The Motivation».

2. Розробити функціонал функціонал додатку, що дозволить взаємодіяти з іншими користувачами додатку за допомогою групових кімнат

3. Розмежувати статус учасників кімнати на:

- адміністратор;
- користувач.

4. Створити графічний інтерфейс(GUI) особистого кабінету користувача.

5. Визначити основні напрямки завдань та їх пріоритетність.

6. Реалізувати можливість створення особистих завдань.

7. Для збереження даних мобільного додатку розробити базу даних.

Розроблений додаток повинен мати наступні характеристики:

– працювати під управлінням ОС Android, починаючи з версії 2.2 (Froyo) і вище;

– мати простий і доступний інтерфейс користувача;

– мати гнучку систему налаштувань.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблений програмний мобільний додаток призначений для користувачів операційної системи Android. В додатку передбачено взаємодія користувачів з іншими користувачами додатку. Функціями даного додатку є:

– особистий кабінет користувача де буде зберігатися особиста інформація користувача, запити на вступ в кімнату та отримані повідомлення;

– створення особистих завдань дає користувачу можливість ставити заплановані задачі. При розміщенні цілі потрібно обирати категорію, яка містить фіксовану кількість балів, які потрібні для складання статистики успішності;

– формування статистики дозволить користувачу контролювати себе і стежити за своїм просуванням до цілі. Кожна категорія містить свою кількість балів і на основі виконаних чи невиконаних завдань в кінці тижня буде формуватися оцінка успішності користувача взагалі і по конкретним категоріям;

– спільні завдання і цілі для команди користувачів. За допомогою кімнат їх адміністратор може розміщувати завдання для виконання їх всіма учасниками кімнати. Кожен користувач програми може створювати кімнати та вступати в інші кімнати.

1.5.2. Вимоги до інформаційної безпеки

На серверному рівні даного додатку повинен бути реалізований захист від втручань сторонніх користувачів у базу даних веб-додатку, зокрема SQL ін’єкцій, за допомогою підготовлених запитів.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для повноцінної роботи додатку потрібно мати мінімальні характеристики:

– версію операційної системи не нижче Android 6.0;
– оперативну пам'ять не менше 2 Гб;
– не менше 2 ядер;
– не менше 32 Гб вбудованої пам’яті;
– діагональ екрана повинна бути не менше 4.7 дюймів;
– частота процесора не менше 1.7 ГГц.

Максимальні характеристики для роботи додатку:

– версію операційної системи не вище Android 10.0;
– оперативну пам'ять не більше 12 Гб;
– не більше 16 ядер;
– не більше 256 Гб вбудованої пам’яті;
– діагональ екрана повинна бути не більше 7 дюймів;
– частота процесора не більше 2.8 ГГц.

1.5.4. Вимоги до інформаційної та програмної сумісності

Мобільний додаток написаний за допомогою мови програмування C# на фраємворкі Xamarin. Він має розширення .app і може бути використаний на смартфонах, що працюють на операційній системі Android. Для стабільної роботи програми версія операційної системи повинна бути не нижче ніж Android 7.0. Такою операційною системою оснащені всі смартфони починаючи з 2016 року і деякі більш старіші телефони, що отримали підтримку сьомої версії Android.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою розробки мобільного додатку є розробка та створення програмного функціоналу, що надає змогу користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання.

Головною особливістю додатку є наявність статистики виконаних чи не виконаних задач та групові кімнати. Статистика розробленого програмного додатку показує успішність людини (користувача) та її витрачені зусилля на досягнення конкретних цілей.

В розробленому додатку реалізовано:

– особистий кабінет користувача, де відображаються отримані повідомлення та запрошення інших користувачів програми;

– створення здач, де користувач може ставити заплановані задачі;

– формування статистики, яка дозволяє користувачеві контролювати себе і стежити за своїм просуванням до цілі;

– спільні завдання і цілі для команди користувачів у групових кімнатах, які можна ставити для групового виконання. За допомогою кімнат їх адміністратор може розміщувати завдання для виконання їх всіма учасниками кімнати. Кожен користувач програми може створювати кімнати та вступати в інші кімнати.

2.2. Опис застосованих математичних методів

В даному програмному забезпеченні ні під час розробки, ні під час тестування та його роботи, не використовуються та не розглядаються ніякі математичні методи

2.3. Опис використаної архітектури та шаблонів проектування

Модель-вигляд-контролер (або модель – представлення –контролер, англ. model-view-controller, MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності [8].

Багатоплатформовий текстовий редактор SublimeText має Інтерфейс, який насправді, це не така вже незначна деталь, як може здатися на перший погляд. Якщо ви користуєтеся інструментом більшу частину робочого дня, в ньому має радувати все. Не тільки швидкість, можливості і зручність, але і естетична частина, теж повинна бути в порядку.

Підсвічування синтаксиста – це найперше, на що звертаєш увагу в будь-якому редакторі. Sublime Text за замовчуванням підтримує величезну кількість мов і пропонує на вибір близько 20 колірних схем.

Повноекранний режим – в цьому режимі робоча область програми займає весь екран. Дуже корисно, якщо ви хочете, щоб вас нічого не відволікало.

Мінікарта – цього не зустрічав ще ні де. У вузькій колонці мінікарти вміщується приблизно 5-6 екранів, що дозволяє швидко переміщатися по коду. Це не заміна і не аналог закладок, а просто ще один зручний спосіб навігації.

Мультипанель – ще одна особливість, притаманна Далеко не всім редакторам, це можливість паралельно працювати з декількома файлами в одному вікні. Часом, це набагато зручніше ніж кілька незалежних вікон.

Автозбереження – для того, щоб не натискати «Зберегти» кожен раз, коли вам необхідно перевірити внесені зміни, в Sublime Text передбачена функція автозбереження. Редактор буде виконувати за вас цю операцію кожен раз, коли вікно програми або вкладка з відкритим файлом втратять фокус.

Кодкомпліт – автозавершення чого завгодно. Якщо надрукувати частину імені відомої функції, ST доповнить її. Якщо відповідних збігів не знайдеться, рядок буде доповнена першій наявній нагоді значенням.

Макроси – дозволяють записати і відтворити найпростіші дії: набір текст, копіювання, вставка і так далі.

Перевірка орфографії – дуже тямуща перевірка орфографії. Підсвічує слова, в яких були допущені орфографічні помилки, але тільки в текстових рядках і в коментарях.

Закладки – значно спрощують навігацію, особливо, коли ви працюєте з великим файлом. Закладка запам'ятовує не просто номер рядка, а також виділену область і положення курсору.

Налаштування – практично всі параметри редактора налаштовуються вручну в текстових файлах. Мабуть, не найзручніший спосіб на світлі, але зате відразу видно всі параметри і опис їх призначення. Також, це дозволяє легко переносити налаштування з одного комп'ютера на інший [9].

TabbedPage – це інструмент групування форм, який дозволяє відображувати вкладки головного меню в нижній частині екрану смартфона. При переході з форми на форму, вкладки не перезавантажуються, що дає змогу процесору постійно їх не викликати. Такий інструмент зазвичай використовується для групування вкладок головного меню, коли користувачеві потрібно мати доступ до багатьох вкладок одночасно. Переходи між вкладками виконується за допомогою натискання на значок вкладки (рис. 2.1).

Рис. 2.1. Реалізація TabbedPage в програмі

За допомогою інструментів NavigationPage вдалося втілити шаблонну навігацію за принципом Model-View-View-Model (MVVM). MVVM дозволило переходити між вкладками по принципу стека. Це інструмент групування вкладок використовується при виклику модальних вікон , тобто користувач не може повернутися на попередню вкладку доки не закриє поточне активне вікно (рис. 2.2).

Рис. 2.2. Реалізація NavigationPage в програмі

2.4. Опис використаних технологій та мов програмування

Мобільна індустрія постійно розвивається і разом з нею розвиваються технології, що дозволяють створювати більш складніші мобільні додатки. При створенні мобільних програм використовують технологію Software Development Kit(SDK).

SDK – комплект засобів розробки, який дозволяє фахівцям з про-грамного забезпечення створювати додатки для певного пакету про-грам, програмного забезпечення базових засобів розробки, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем і інших платформ. SDK використовує переваги кожної платформи і скорочує час на

інтеграцію.

Існує різні версії SDK, які використовуються для написання коду для різних версій Android. В даний час велике поширення отримали ве-рсії 2017 і 2019. Підтримується майже повна зворотна сумісність версій.

Крім розробки на мові C# підтримується можливість більш низькорівнева розробка з використанням Android NDK (Native Development Kit) на мові C/C++.

Операційні системи Android і IOS є найбільш популярними і тому під них створюють все більш нові технологій. Одними с таких технологій є:

– java 2 Micro Edition (J2ME);
– windows Phone SDK;
– iPhone SDK;
– android SDK.

В першу чергу J2ME набір специфікацій і технологій, призначених для різних типів портативних пристроїв. Існують два основних напрямки: Connected Device Configuration (CDC) і Connected Limited Device Configuration (CLDC). Напрямок визначає тип конфігурації центральних бібліотек Java, а так же параметрів віртуальної машини Java. Спеціальні режими дозволяють визначати функціональність конфігурацій для різних типів пристроїв. Режим Mobile Information Device Profile (MIDP) призначений для CLDC портативних пристроїв з можливістю комуніку-вати. Режим MIDP визначається функціонал роботи інтерфейсу призначеної для користувача, збереження налаштувань, роботу в мережі і модель програми. CLDC і MIDP закладають основу реалізації J2ME.

Для Windows Phone SDK код розробляється та описується на мові XAML. XML – це файли з мовою розмітки XAML.

Платформа Windows Phone створена для мобільних пристроїв. Вона містить в собі технологічну складову і повністю опрацьовану концепцію дизайну інтерфейсу і взаємодії з користувачем під назвою Metro-дизайн або стиль Metro.

Вся розробка мобільних програм під Windows Phone ведеться в середовищі MS Visual Studio 2019. Середовище для розробки і налагодження додатків. Для мобільних додатків під Windows Phone налагодження відбувається за допомогою емулятора Windows Phone.

Для iPhone SDK розробка під операційну систему iOS можлива тільки під Mac OS. Для написання програм під iPhone використовується використовувати Objective C. При цьому є можливість писати так само і на C і на C ++ (для цього необхідно змінювати розширення файлів з .m на .mm). При цьому повністю піти від Objective C не вдасться, майже весь API розрахований саме на Objective C, виключення складають наприклад OpenGL, так само повністю доступні стандартні бібліотеки C.

Для Android SDK розробки мобільних програм під Android можна використовувати середу Visual Studio з встановленим фраємворком Xamarin. Розробка ведеться на мові програмування C#. Є можливість налагодження з використанням емулятора вбудованого в Visual Studio 2017 та вище, або безпосередньо на мобільному пристрої з ОС Android[1].

Архітектуру Android можна розділити на шість рівнів:

– обладнання;
– ядра Linux;
– нативних бібліотек;
– середовища виконання Android;
– каркаса додатків (Application Framework);
– додатків.

Операційна система Android на даний момент має 10 версій. Починаючи з першої версії операційна система постійно вдосконалюється. Кожне оновлення несло розширення функціоналу та влаштування нових технологій (табл. 2.1)[8].

Таблиця 2.1

Порівняння версій ОС Android

Версія Android	Нововведення
Android 6.0	Змінився дизайн, покращилася функція блокування методом розпізнавання особи за допомогою фронтальної камери, булла додана можливість блокування гаджета за допомогою відбитків пальців і можливість швидких платежів через систему Android Pay. Оновлено популярний додаток Google Now.
Android 8.0	– нові іконки, які можуть бути виконані в єдиному стилі; – з'явилися індикатори нових повідомлень у іконок додатків; – системні ікони також змінилися; – з'явилися канали повідомлень; – з'явилася нативна підтримка багатьох кодеків для бездротової передачі звуку (LDAC, aptX, aptX HD, SBC і AAC); – стала доступна підтримка фізичних клавіатур; – робота програм у фоновому режимі обмежена;
Android 9.0	Недавні версії Android робили акцент на збільшенні часу автономної роботи. Android Pie продовжує цю тенденцію з режимом Adaptive Battery. Він використовує машинне навчання, щоб виз начити, які додатки використовуються частіше, і потім адаптує використання батареї так, щоб найбільш використовувані додатки отримували пріоритет. Режим надає гнучкі можливості для контролю витрат заряду акумулятора і дозволить тонко його налаштувати. Наприклад, можна відключити відображення часу на смартфонах з «завжди включеним дисплеєм».

Продовж. табл.2.1

Версія Android	Нововведення
Android 10.0	В Android 10 можна перейти в темну тему інтерфейсу (Dark Mode) і відключити повідомлення в додатках без входу в них. Також в десятій версії ОС з'явилася функція Live Caption, яка накладає субтитри на будь-які аудіо- і відеозаписи. Нова навігацію жестами, за допомогою якої можна видаляти повідомлення або відкривати додаткове меню. Ще однією цікавою новинкою стала вбудована функція запису екрану.

Мобільний додаток – це набір технологій реалізованих на платформі розробки додатку з використанням мови програмування (рис.2.1).

Рис. 2.1. Склад мобільної програми

На основі поставлених задач було обране середовище програмування Microsoft Visual Studio, фреймворк Xamarin та мова програмування C#.

Для створення повноцінного додатку потрібна база даних. В фреймворкі

Xamarin є вбудована підтримка однієї з поширених систем управління базами даних(БД) – SQLite. Для цього в пакеті android.database.sqlite визначено набір класів, які дозволяють працювати з базами даних SQLite. І кожен додаток може створити свою базу даних.

ОС Android за замовчуванням вже містить ряд вбудованих баз SQLite, які використовуються стандартними програмами для списку ко-нтактів, для зберігання фотографій з камери, музичних альбомів і т.д.

Основну функціональність по роботі з базами даних надає пакет android.database. Функціональність безпосередньо для роботи з SQLite знаходиться в пакеті android.database.sqlite.

База даних в SQLite представлена різноманітними класами:

– android.database.sqlite.SQLiteDatabase дозволяє виконувати запити до БД, виконувати з нею різні маніпуляції;

– android.database.sqlite.SQLiteCursor надає запит і дозволяє повертати набір рядків, які відповідають цьому запиту;

– android.database.sqlite.SQLiteQueryBuilder дозволяє створювати SQL-запити;

– android.database.sqlite.SQLiteStatement, які дозволяють за допомогою плейсхолдеров вставляти в вираження динамічні дані;

– android.database.sqlite.SQLiteOpenHelper дозволяє створити базу даних з усіма таблицями, якщо їх ще не існує.

У SQLite застосовується інтегрована система типів даних:

– integer, представляє ціле число, аналог типу int в C#;

– real, представляє число з плаваючою точкою, аналог float і double вC#;

– text, представляє набір символів, аналог String і char в C#;

– blob, представляє масив бінарних даних, наприклад, зображення, аналог типу int в C#.

Зберігаються дані повинні подавати відповідні типи в C#.

Для створення або відкриття нової бази даних з коду Activity в Android можна викликати метод openOrCreateDatabase ()[2].

Аналізуючи вище описане для розробки мобільного додатку потрібен фраємворк Xamarin від корпорації Microsoft. Xamarin підтримує роботу з базами даних і новітні технології розробки програм.

Існує кілька вагомих причин, за якими Xamarin використовується багатьма компаніями:

– єдиний стек технологій для розробки на всіх платформах;

– xamarin використовує мову C# і .NET Framework, щоб створювати додатки для будь-якої мобільної платформи;

– продуктивність близька до нативної. Багатоплатформовий додаток, створений за допомогою Xamarin, можна класифікувати як нативне, на відміну від традиційних гібридних рішень, що базуються на вебтехнологіях. Показники продуктивності можна порівняти з показниками Java під Android та Objective-C або Swift для розробки додатків під iOS;

– власний призначений для користувача досвід. Використовуючи платформо-залежні елементи UI, Xamarin дозволяє створювати інтерфейс[3];

– повна підтримка обладнання усуває всі проблеми сумісності обладнання, використовуючи плагіни і різні API для роботи з функціями загальних пристроїв на всіх платформах;

– проста підтримка платформ, Xamarin спрощує підтримку і оновлення програмного забезпечення. Можна просто внести зміни в один вихідний файл, і вони будуть застосовуватися до додатків як iOS, так і Android;

– повна екосистема розвитку Xamarin поставляється в одному наборі з повним пакетом інструментів розробки: власна система IDE;

– підтримка роботи з базами даних SQLLite.

Мова C# зараховується до об’єктно-орієнтованих, а точніше об’єктних мов програмування. «Мова заснована на суворій компонентній архітектурі і реалізує передові механізми забезпечення безпеки коду» – так прийнято характеризувати його. C # живе за принципом «будь-яка сутність є об’єкт». C # відноситься до сім’ї мов C подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує

поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML[4].

Розробка мобільних додатків на мові програмування C# реалізовується в середовищі Visual Studio 2019 та на платформі Xamarin (рис. 2.2).

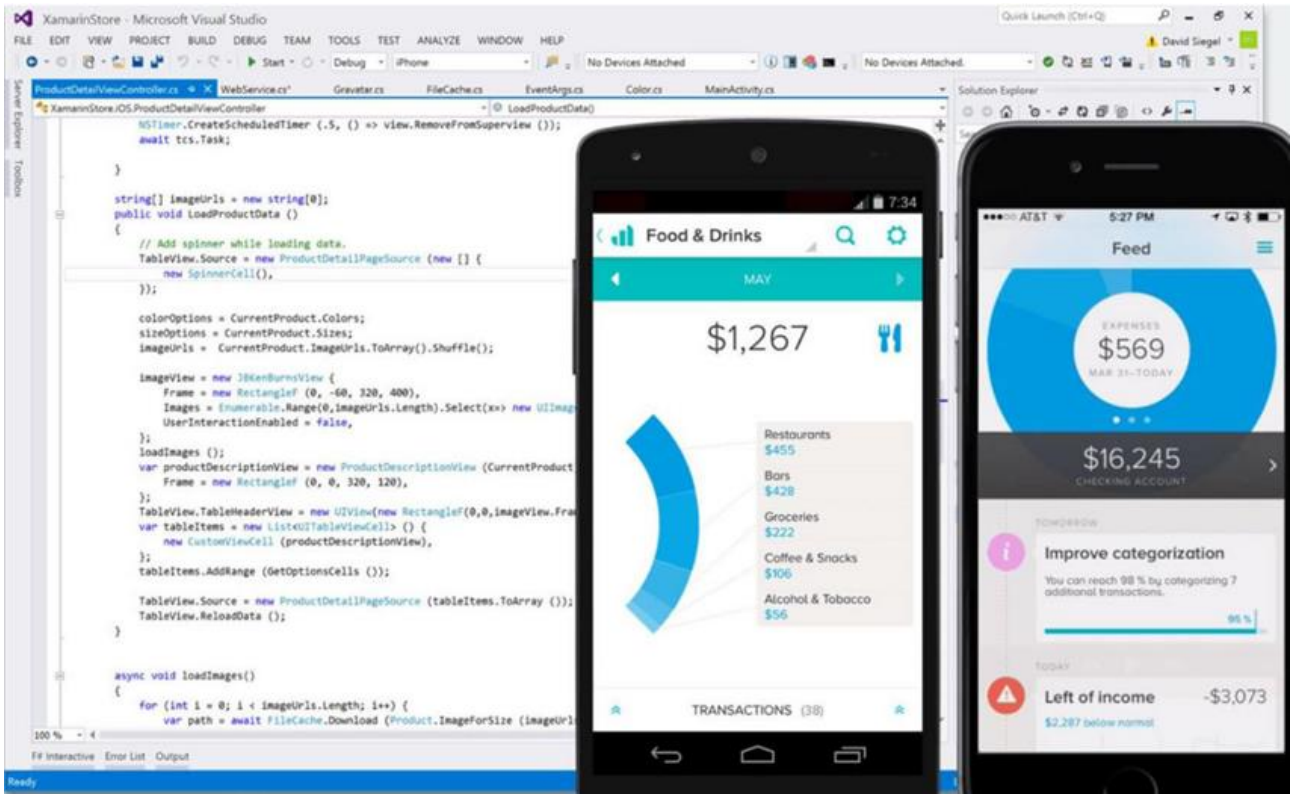

Рис. 2.2. Xamarin Forms

Починаючи з Visual Studio 2017 фраємворк Xamarin поставляється, як інструмент програмування мобільних додатків, але функціонал Visual Studio 2019 більш кращий і дозволяє взаємодіяти з базами даних за допомогою БД SQLLite.

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Інструменти Visual Studio дозволяють розробляти, як консольні додатки, так і додатки з графічним інтерфейсом(GUI),

в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight. Visual Studio дозволяє створювати власні проекти під різні платформи (рис. 2.3)[5].

MS Visual Studio включає в себе редактор коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований налагоджувач може працювати, як налагоджувач рівня вихідного коду, так і машинного рівня. Інші інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних[6].

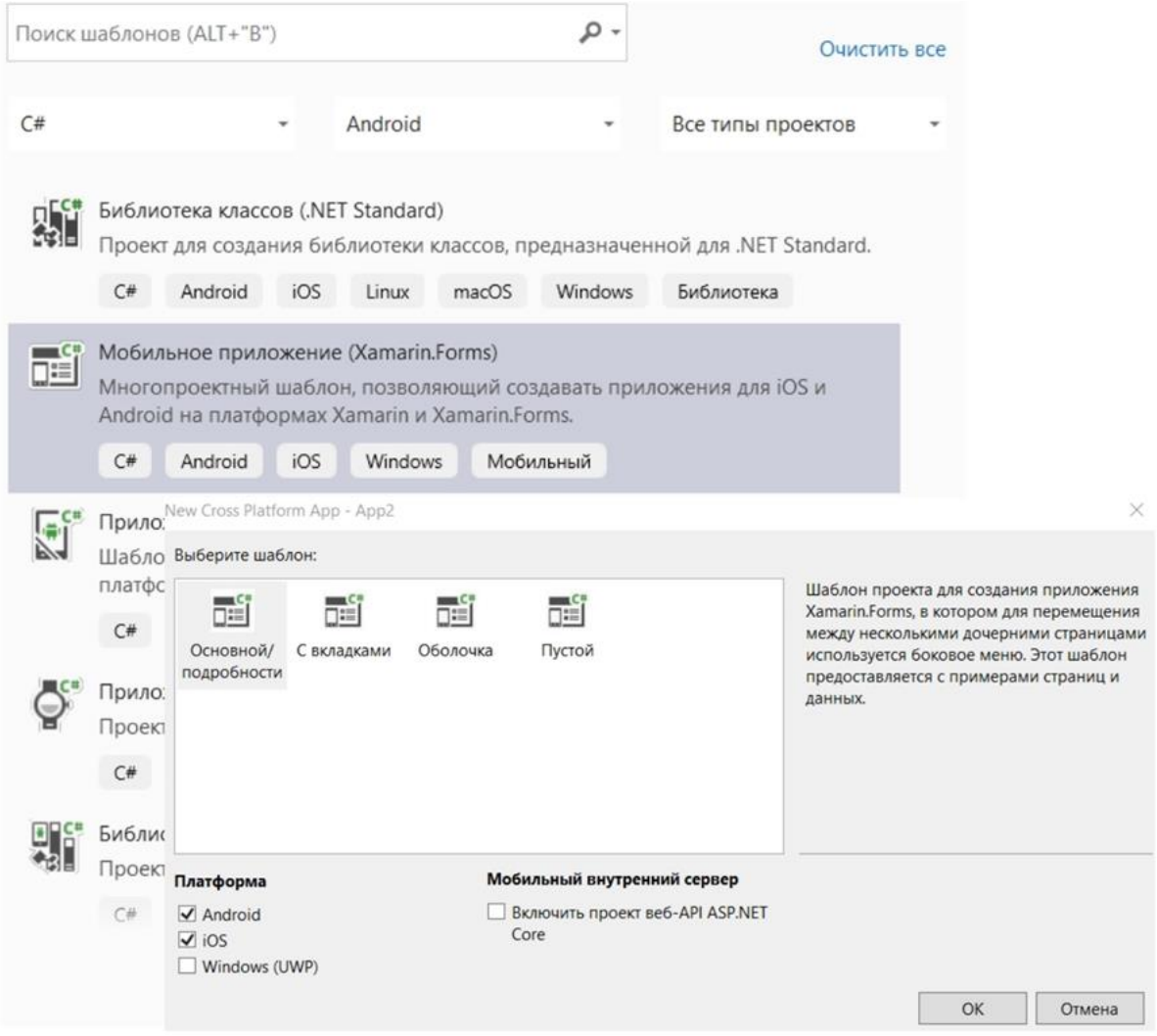

Рис. 2.3. Вікно створення проекту

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server)[7].

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Опис логічної структури додатку

Підчас логічного проектування була створена UML діаграма, що відображає структуру та залежність процесів в програмі (рис. 2.4).

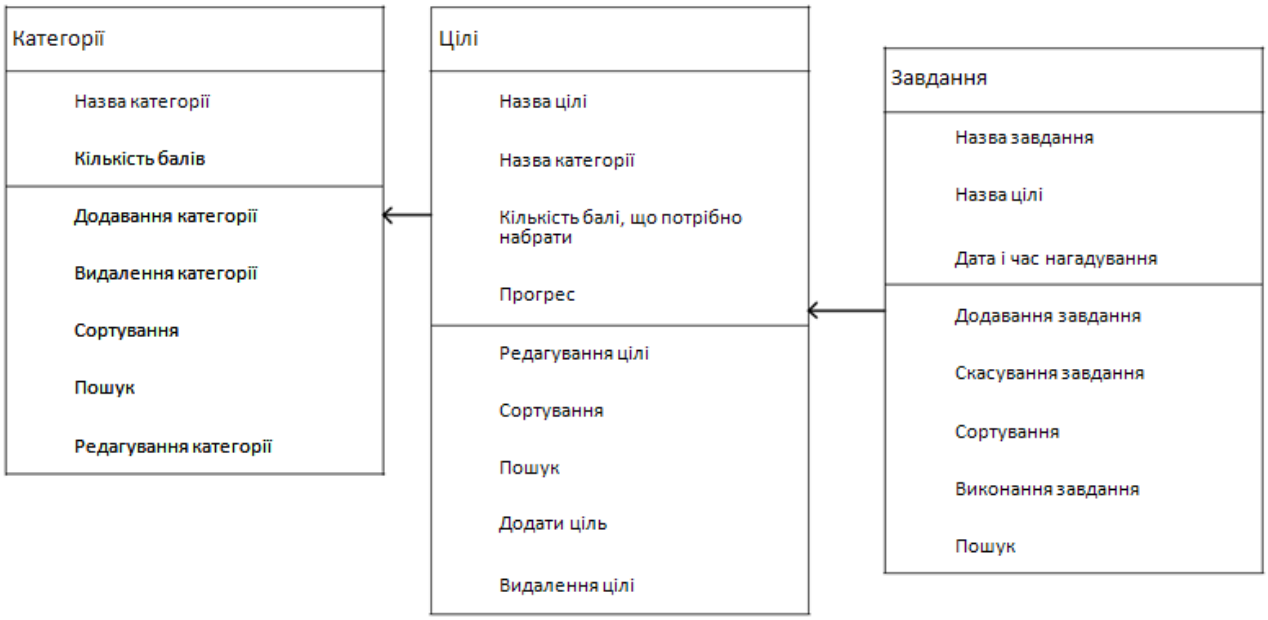

Рис. 2.4. UML діаграма додатку «The Motivation»

Щоб відобразити послідовності всіх дії в програмі використовується діаграма послідовностей (SEQUENCE DIAGRAM).

SEQUENCE DIAGRAM – це різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень. В ролі об’єктів виступають користувач, смартфон, програмний додаток та сервер з базою даних. Ці об’єкти зображені на рис. 2.5.

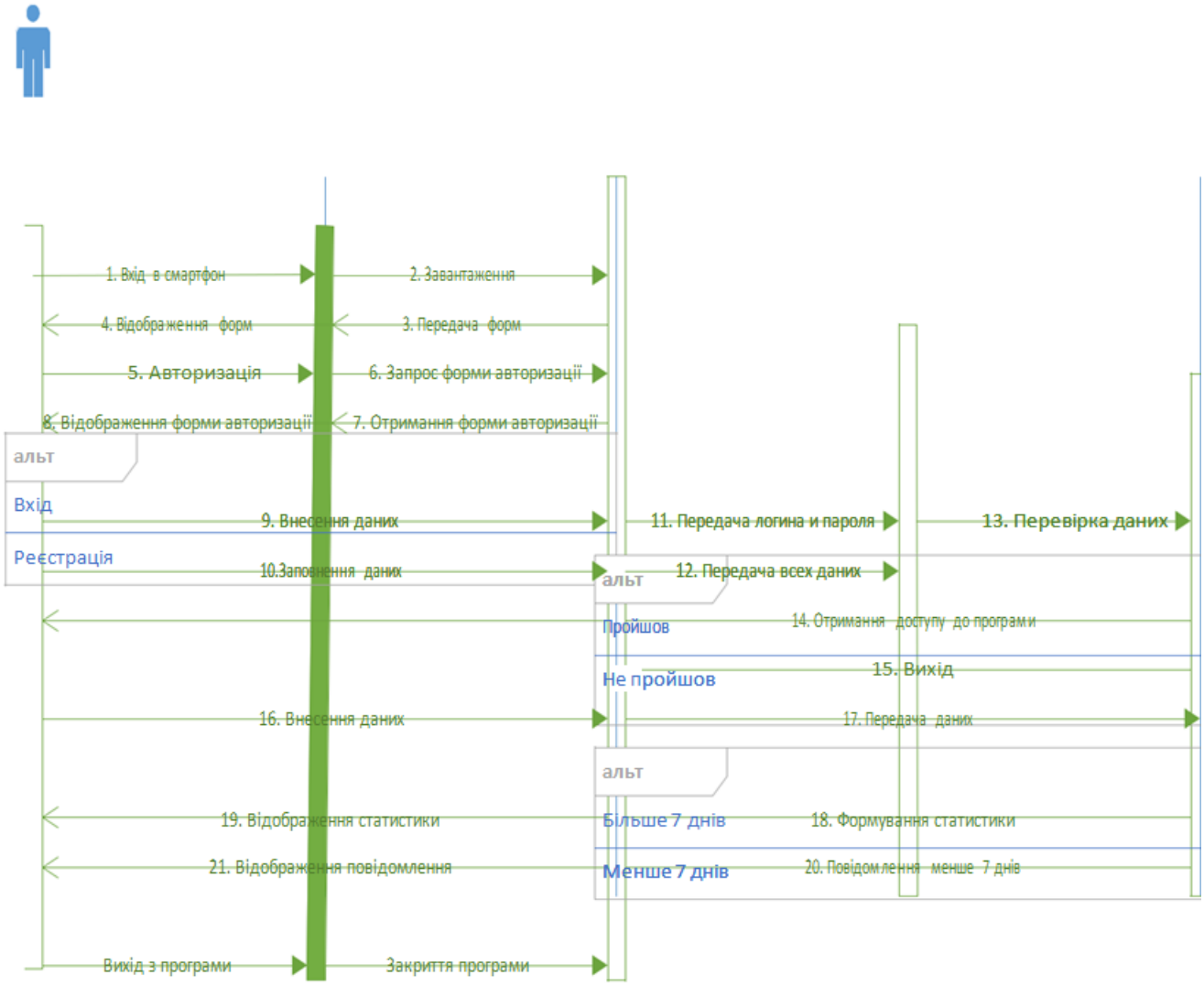

Рис. 2.5. SEQUENCE DIAGRAM додатку «The Motivation»

Для відображення відношень між актором та прецедентом використовується UseCase діаграма, що зображена на рис. 2.6

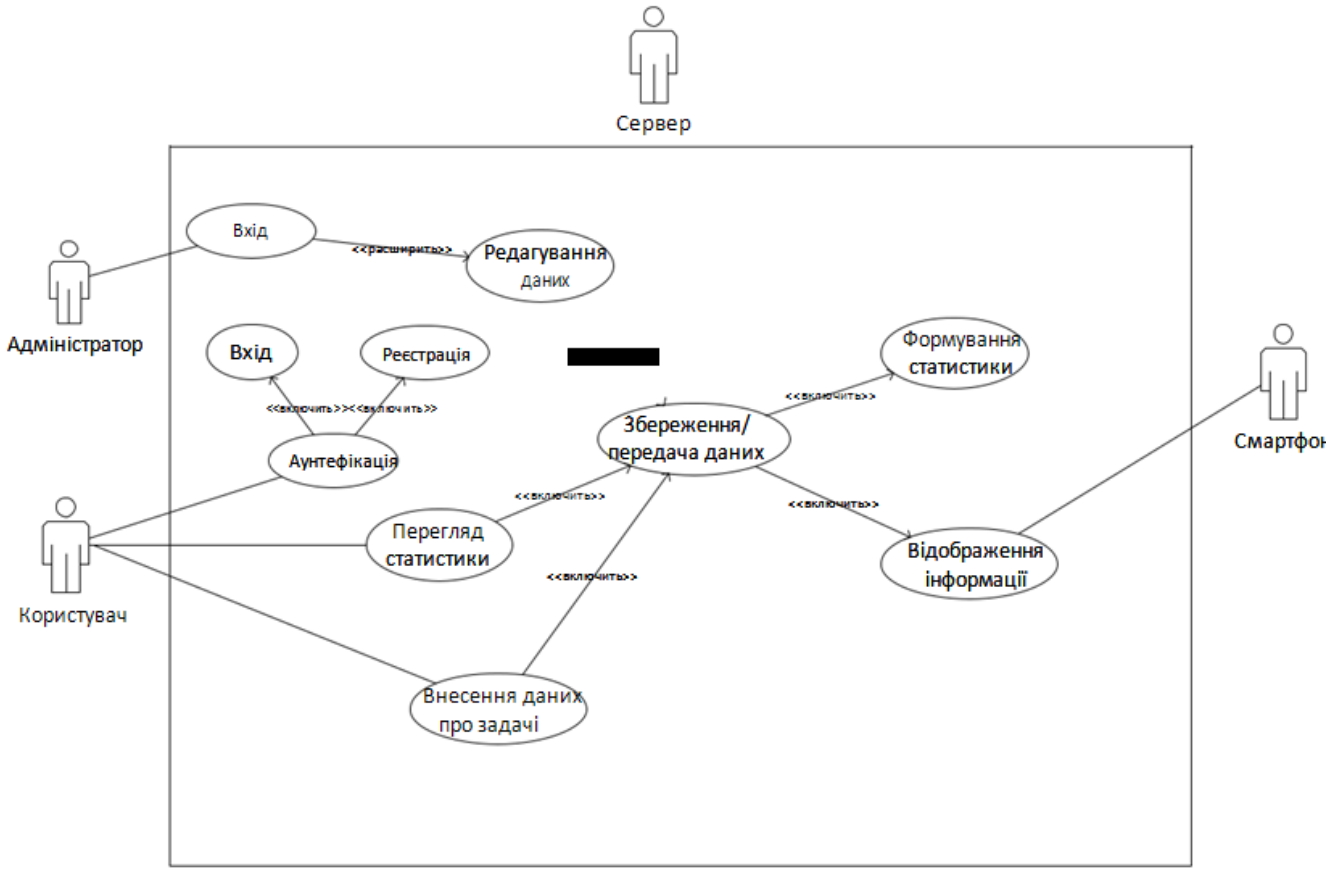

Рис. 2.6. UseCase діаграма додатку «The Motivation»

Діаграма діяльності додатку наведена на рис. 2.7.

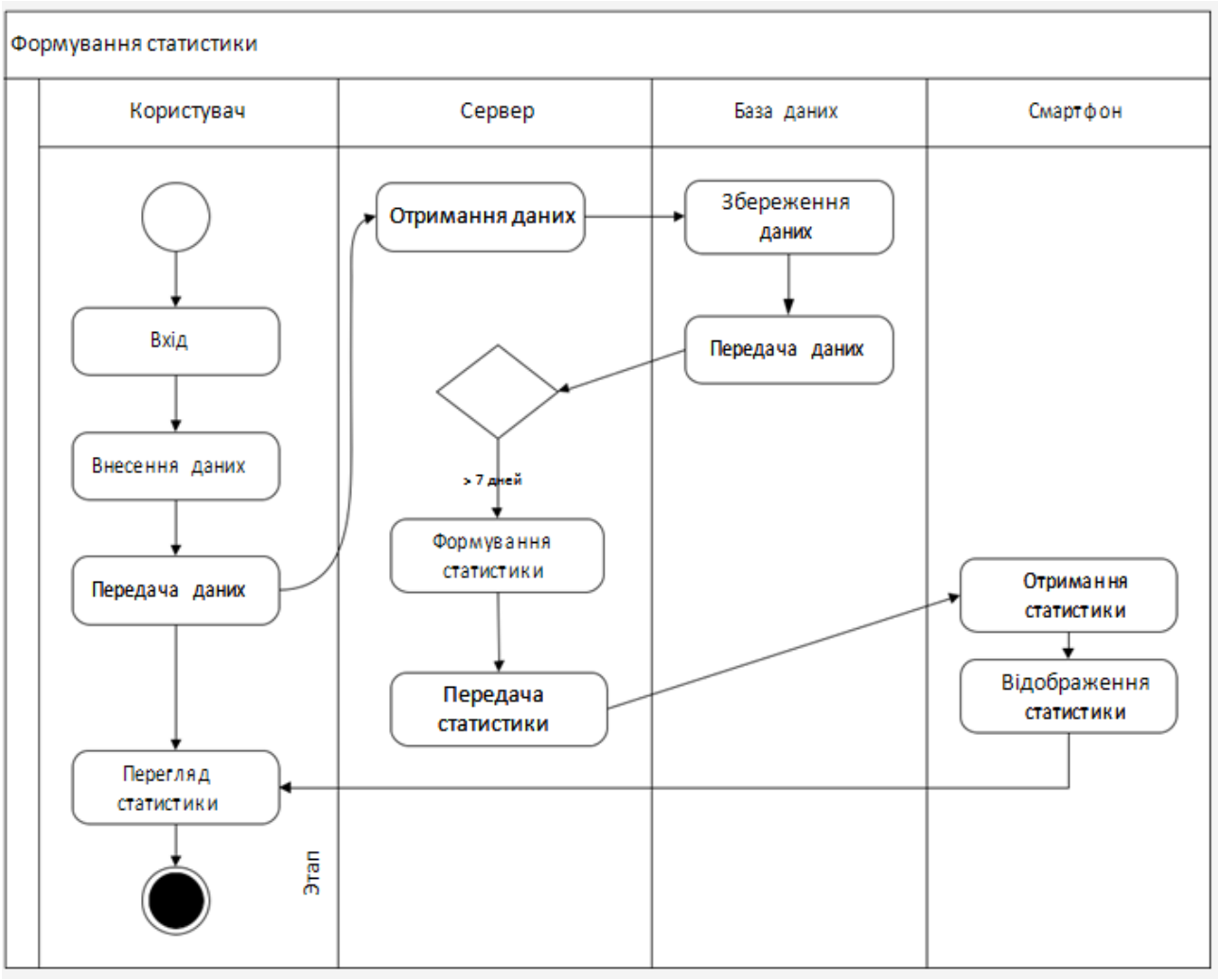

Рис. 2.7. Діаграма діяльності додатку «The Motivation»

Діаграма комунікації (англ. Communication diagram, в UML 1.x - діаграма кооперації, англ. collaboration diagram) - діаграма, на якій зображуються взаємодії між частинами композитної структури або ролями кооперації. На відміну від діаграми послідовності, на діаграмі комунікації явно вказуються відносини між об'єктами, а час як окремий вимір не використовується (застосовуються порядкові номери викликів).

Діаграма комунікації моделює взаємодії між об'єктами або частинами в термінах впорядкованих повідомлень. Комунікаційні діаграми представляють комбінацію інформації, взятої з діаграм класів, послідовності і варіантів використання, описуючи відразу і статичну структуру і динамічну поведінку системи.

Діаграма комунікації показує багато в чому ту ж інформацію, що і діаграма послідовності, але через іншого способу подання інформації якісь речі на одній діаграмі бачити простіше, ніж на інший. Діаграма комунікацій наочніше показує, з якими елементами взаємодіє кожен елемент, а діаграма послідовності ясніше показує в якому порядку відбуваються взаємодії.

На рис. 2.8 зображена Collaboration Diagram додатку « The Motivation».

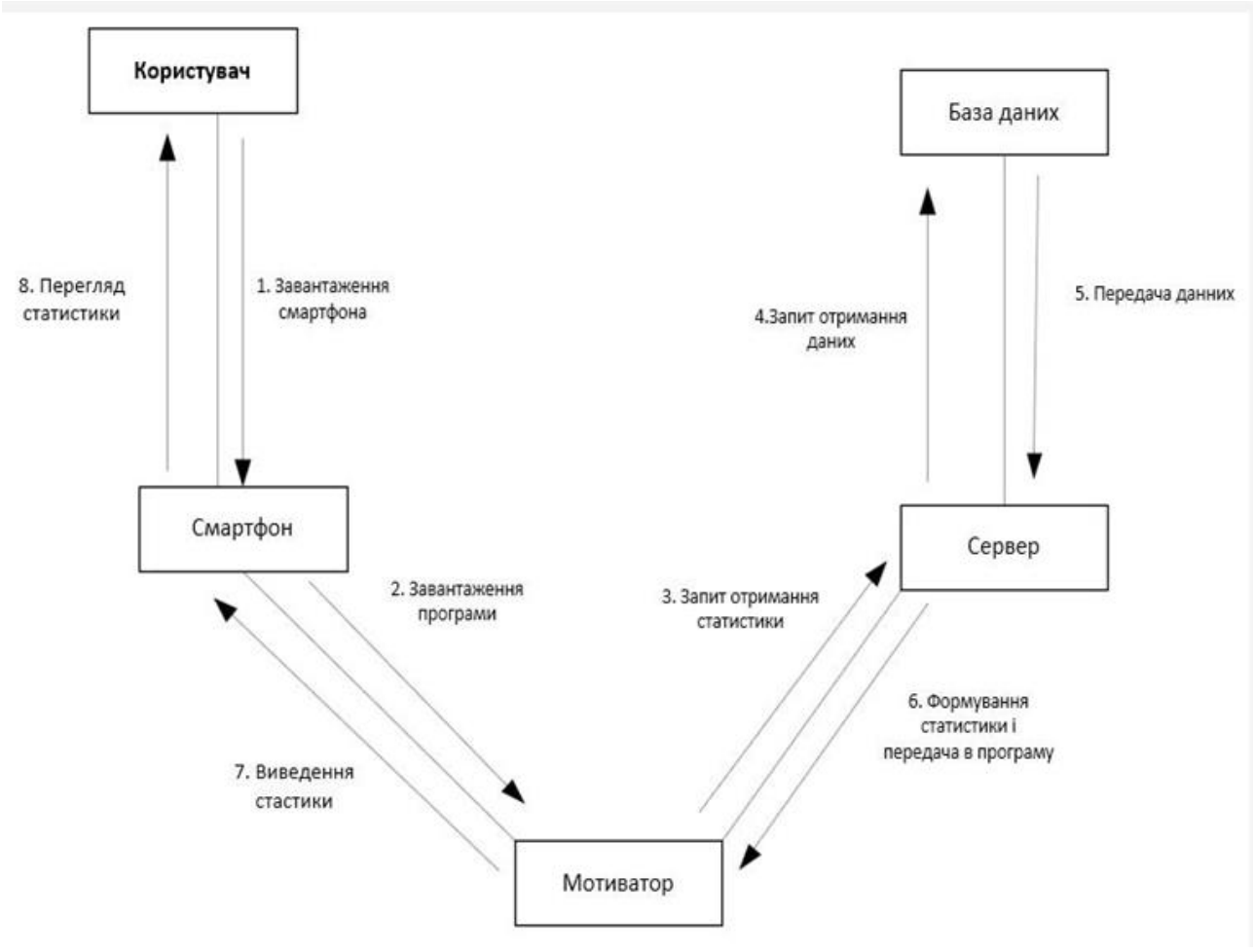

Рис. 2.8. Collaboration Diagram додатку « The Motivation»

2.5.2. Опис розробки бази даних проекту

Всі данні зберігаються у таблицях бази даних. Таблиця – це об’єкт, призначений для збереження даних у вигляді записів (рядків) та полів (стовпців). Звичайно кожна таблиця використовується для збереження

відомостей по одному конкретному питанню, наприклад о предметах або спеціальностях.

Кожне поле таблиці має свій тип. При проектуванні таблиць використовувались наступні типи даних та позначення:

– C – символьне значення до 255 символів;

– T – час;

– D – дата;

– F – дійсне число;

– N – ціле число.

При встановленні зв’язку між таблицями використалися наступні позначення:

– PK – (Primary key) первинний ключ – унікальне поле,

– FK – (Foreign key) зовнішній ключ – набор атрибутів одного відношення, яке є потенційним ключем другого відношення.

Нижче наведений опис сутностей у вигляді реляційних таблиць (табл. 2.2-2.6)

Таблиця 2.2

Схема відносин Користувач (User)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор користувача	IdUser	N	Первинний ключ
Ім’я	Name	C(15)	Обов’язкове поле
Прізвище	Surname	C(20)	Обов’язкове поле
Телефон	Phone	C(10)	Обов’язкове поле
Логін	Login	C(30)	Обов’язкове поле
Пароль	Password	C(30)	Обов’язкове поле
Електронна адреса	Email	C(30)	Обов’язкове поле

Таблиця 2.3

Схема відносин Цілі (Target)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор цілі	TargetId	N	Первинний ключ
Ідентифікатор категорії	IdCategories	N	Зовнішній ключ
Ідентифікатор користувача	IdUser	N	Зовнішній ключ
Назва цілі	Name_Target	C(20)	Обов’язкове поле
Кількість балів, що потрібно набрати	Point_Target	N	Обов’язкове поле
Прогрес	Complate_Poi	F	Обов’язкове поле

Таблиця 2.4

Схема відносин Категорії (Categories)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор категорії	IdCategories	N	Первинний ключ
ID Користувач	IdUser	N	Зовнішній ключ
Назва категорії	Name_categories	C(20)	Обов’язкове поле
Кількість балів	Points	N	Обов’язкове поле

Таблиця 2.5

Схема відносин Кімнти (Room)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор кімнати	IdRoom	N	Первинний ключ
Назва	Name_Room	C(20)	Обов’язкове поле
Кількість учасників	Num_Us	N	Обов’язкове поле

Таблиця 2.6

Схема відносин Завдання (Tasks)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор завдання	IdTask	N	Первинний ключ
Ідентифікатор цілі	TargetId	N	Зовнішній ключ
Ідентифікатор користу вача	IdUse r	N	Зовнішній ключ
Назва завдання	Name_Task	C(20)	Обов’язкове поле
Дата	Date	N	Обов’язкове поле
Час	Time	F	Обов’язкове поле
Статус	Status	N	Обов’язкове поле

В ході проектування бази даних було прийнято рішення спроектувати додаток таким чино, щоб можна було взаємодіяти з іншими користувачами і дані, які заносить користувач, зберігалися на сервері. Таким чином база даних додатка спроектована під систему керування базами даних MySQL і розмінюється на віддаленому сервері. При проектування використалась реляційна модель бази даних.

Для налаштовування бази даних використовується сервіс від хостингу phpMyAdmin. За допомогою цього сервісу була побудована база даних (рис. 2.9), а ER-діаграма додатку зображена на рис. 2.10.

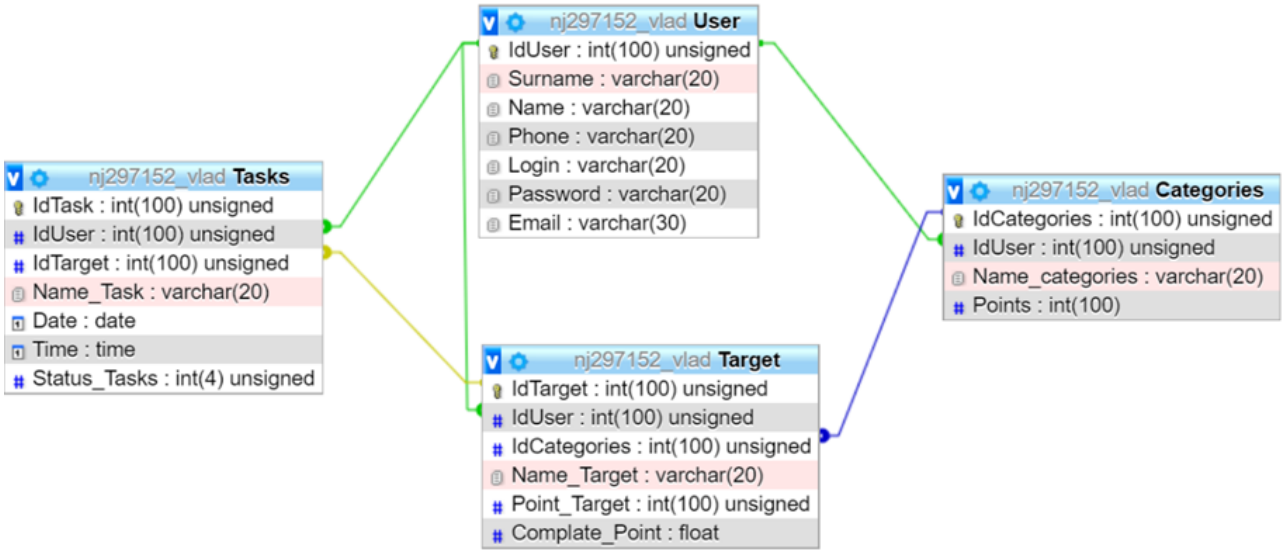

Рис. 2.9 Схема БД проекту мобільної додатку «The Motivation»

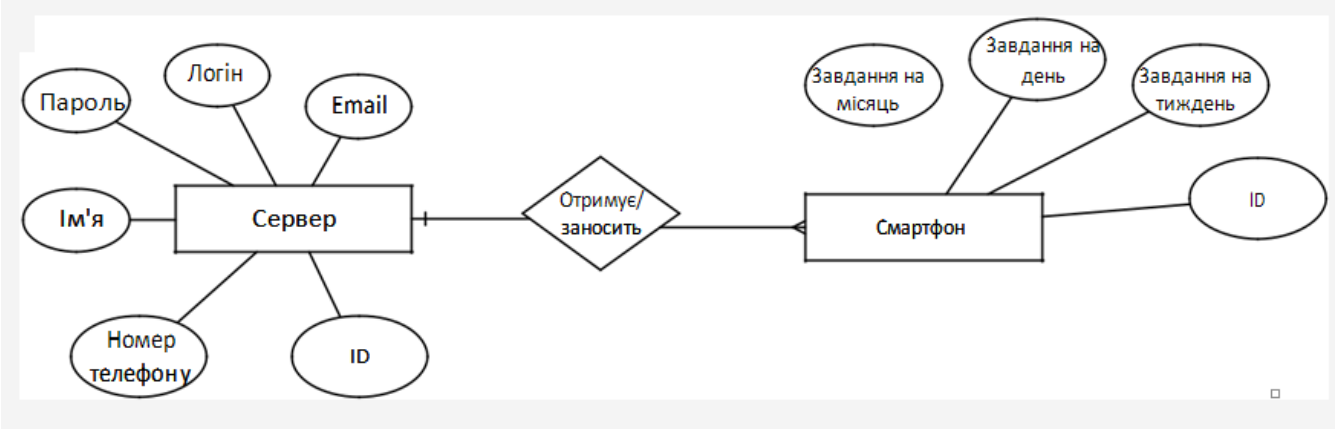

Рис. 2.10. ER-діаграма додатку «The Motivation»

Для роботи з базою даних була обрана технологія Ado.Net. За допомогою збережених процедур, що розміщуються на сервері, було реалізовано:

1. Додавання даних:
   - insert_User – додавання користувачів;
   - insert_Tasks – додавання завдань;
   - insert_Target – додавання цілей;
   - insert_Categories – додавання категорій.
2. Вивід даних:
   - select_User – вводу користувачів;

– select_Terget – виводу цілей;

– select_Tasks – виводу завдань;

– select_Categories – виводу категорій.

3. Видалення даних:

– dell_Categories – видалення категорій;

– dell_Target – видалення цілей;

– dell_Tasks – видалення завдань.

4. Пошук:

– search_Target – пошук по цілям;

– search_Categories – пошук по категоріям;

– search_Tasks – пошук по завданням.

5. Оновлення:

– update_Email – оновлення електронної адреси;

– update_Phone – номеру телефона;

– update_Status_0, Update_Status_1 – оновлення статусу виконання завдання;

– update_Complate_Point, Update_Complate_Point_NOT – оновлення прогресу цілі.

6. Перевірка:

– check_User – перевірка наявності користувача в базі;

– check_Login – перевірка коректності вводу даних при вході в особистий кабінет;

– check_Phone – перевірка збіжності номера телефону.

2.5.3. Опис файлової структури проекту

Для відображення обчислювальних вузлів під час роботи програми, використовується діаграма розгортання, яка зображена на рис. 2.11.

Рис. 2.11. Діаграма розгортання додатку «The Motivation»

Діаграма компонентів додатку наведена на рис. 2.12.

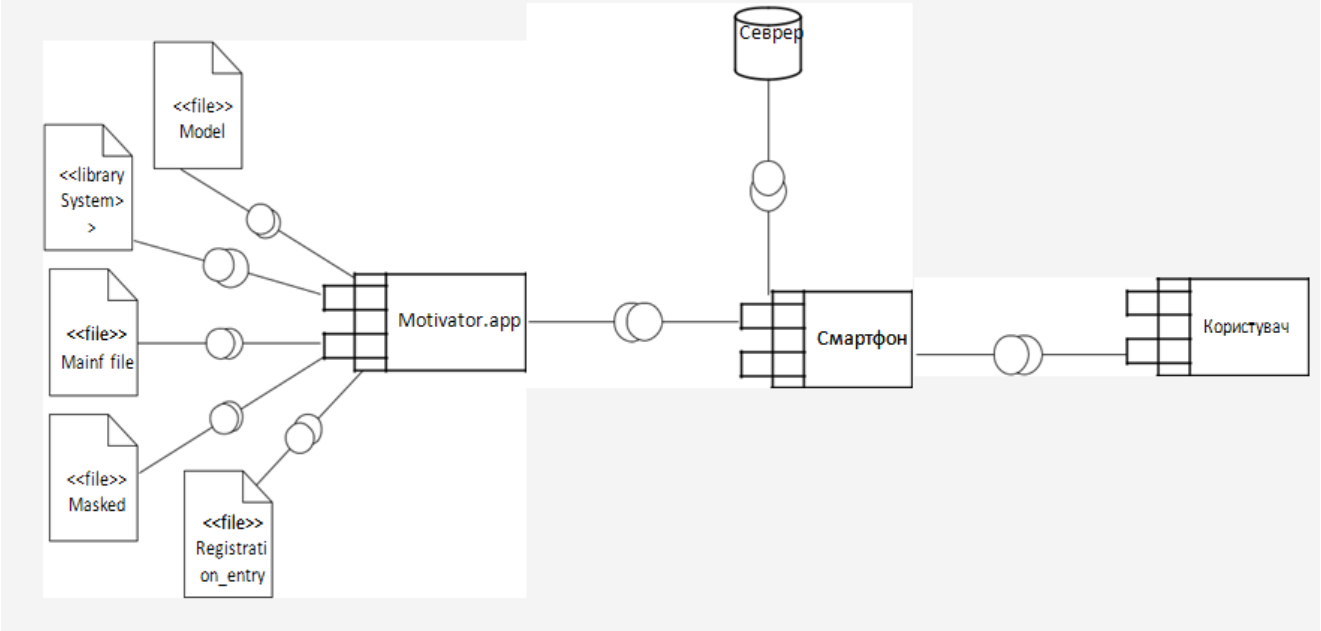

Рис. 2. 12. Component Diagram додатку «The Motivation»

Після створення проекту з’являються кореневі файли, що зберігають початкову структуру проекту (рис. 2.13). В процесі створення додатку початкова структура буде змінюватися, так як будуть з’являтися нові файли, що

зберігатимуть набори форм, інтерфейсів та класів.

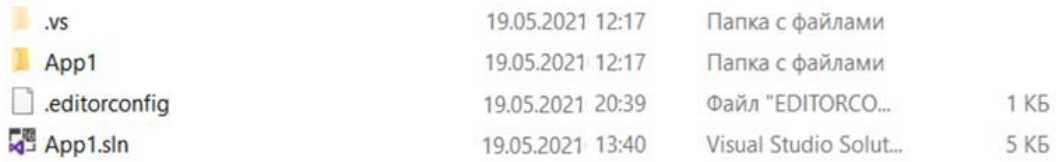

Рис. 2.13. Файл запуску проекту

Під час проектування програми була створена файлова структура файлів та папок проекту, яка наведена рис. 214.

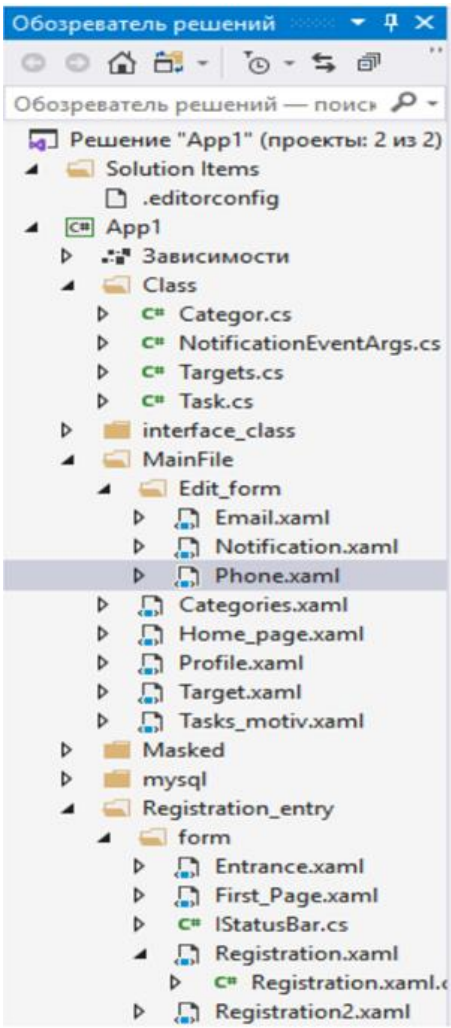

Рис. 2.14. Файлова структура проекту The_Motivation.app

Файлова структура проєкту включає в себе папки:

– class – для зберігання класів обробки локальних даних;

– mainFile – для зберігання головних навігаційних вкладок програми;

– edit_form – для зберігання вкладок редагування;

– masked – для зберігання класів, що надають методи вводу користувацьких даних з обмеженням;

– mysql – для зберігання класів обробки даних з серверу;

– registration_entry – для зберігання вкладок реєстрації та входу. Для навігації в програмі використовуються вбудовані інструменти

– переходу між формами(Page) «TabbedPage» та «NavigationPage».

Всі xml вкладки включають файли з програмним кодом, які мають розширення *.cs.

Для обробки локальних даних використовуються класи. В усіх класах допомогою методів GET() та SET() отримуємо та вносимо данні.

Для роботи з базою даних створені класи, що відображають дані (рис. 2.15).

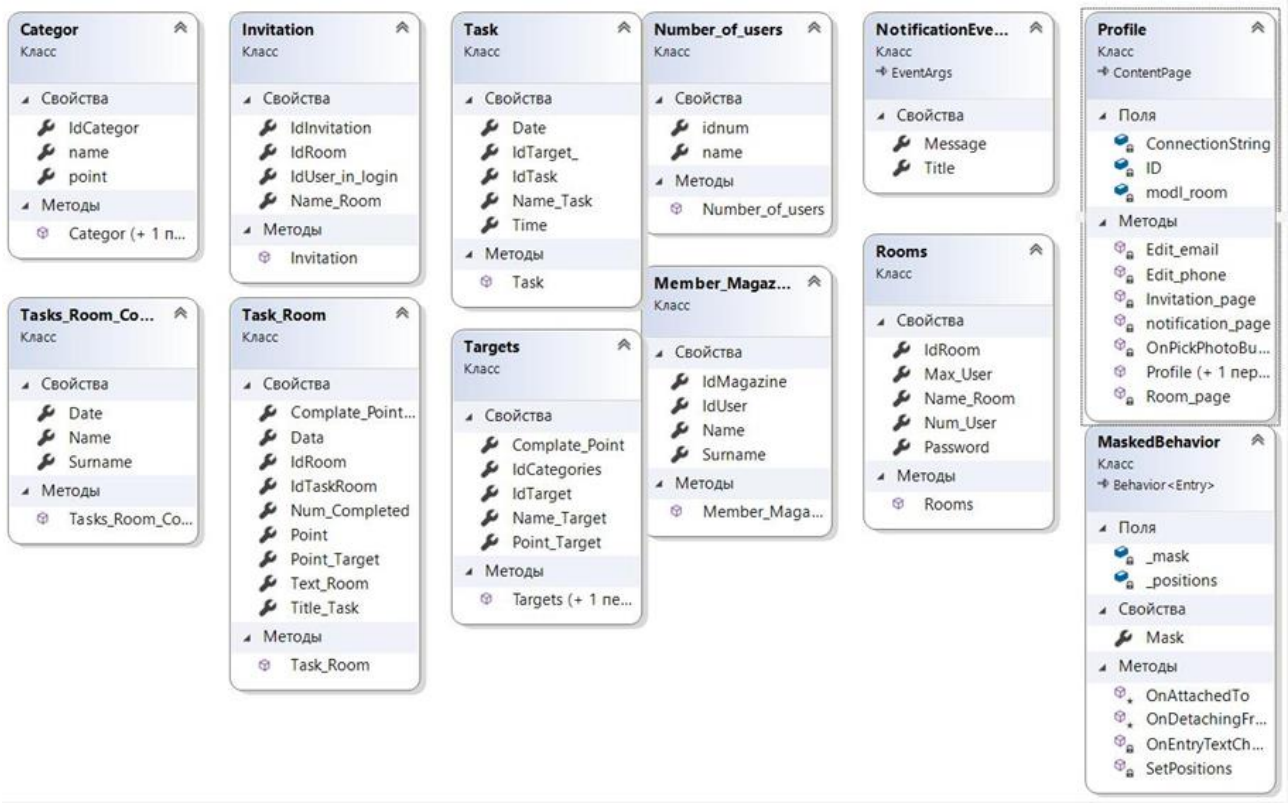

Рис. 2.15. Діаграма класів додатку «The Motivation

Для обробки та відображення даних створено класи Categor – клас для відображення даних по категоріям, Targets – для відображення даних по цілям, Task – для відображення даних по завданням.

В оголошенні класу Categor у якості атрибутів, виступають дані типу string: name – назва категорії, point – кількість балів за категорію, IdCategor – ідентифікатор категорії.

При визначені конструкторів класу Categor були обрані параметри:

– categor(string name, string point, string ID) – конструктор з параметрами класу Categor;

– categor() – конструктор без параметрів класу Categor.

В класі Targets у якості атрибутів, виступають дані типу string: IdTarget – ідентифікатор цілі, IdCategories– ідентифікатор категорії, Name_Target – назва цілі, Point_Target – кількість балів, що потрібно на-брати, Complate_Point – кількість набраних балів.

При визначені конструкторів класу Targets були обрані параметри:

– targets(string IdTarget, string IdCategories, string Name_Target, string Point_Target,string Complate_Point) – конструктор з параметрами класу Targets;

– targets() – конструктор без параметрів класу Targets.

При оголошенні класу Task у якості атрибутів, виступають дані типу string: IdTask – ідентифікатор завдання, IdTarget – ідентифікатор цілі, Name_Task – текст завдання, Date – дата нагадування, Time – час нагадування.

При визначені конструкторів класу Task були обрані параметри:

– task(string IdTask, string IdTarget, string Name_Task, string Date, string Time) – конструктор з параметрами класу Task;

– task() – конструктор без параметрів класу Task.

Для роботи формами входу та реєстрації використовує класи, що з’являються при створенні Page. Ці класи мають таку ж назву, як і форма.

Клас Entrance, що відображає вікно входу у якості атрибутів, висту-пають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

При визначені конструкторів класу Entrance були обрані параметри:

– entrance() – конструктор без параметрів класу Entrance;

– async void Login_password() – функція автентифікації;

– async void Registration() – функція переходу на форму реєстрації. При оголошенні класу Registration, що відображає перше вікно реєстрації у якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

При визначені конструкторів класу Registration буди обрані параметри:

– registration () – конструктор без параметрів класу Registration;

– void Further() – функція перевірки даних.

Для оголошеного класу Registration2, що відображає останнє вікно реєстрації у якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL, login – логін, password – пароль, email – електронна адреса.

При визначені конструкторів класу Registration2 були обрані параметри:

– registration2 (string login, string password, string email) – конструктор з параметрами класу Registration2;

– check_phone(string s) – функція перевірки коректності вводу номеру телефону;

– async void Further2() – функція перевірки коректності всіх даних та реєстрації нового користувача.

При роботі з вкладками головного меню використовуємо Page, що прив’язані до інструменту навігації TabbedPage. TabbedPage дозволяє відображати багато вкладок і переходити на іншу форму не закриваючи попередню.

В класі Categories, що відображає вікно відображення категорій у якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

При визначені конструкторів класу Categories були обрані параметри:

– Categories (int id) – конструктор з параметрами класу Categories;

– void Add_Categories () – функція додавання категорії;

– void Sort_Name() – функція сортування категорій по назві;

– void Sort_Point() – функція сортування категорій по балам;

– void Search() – функція пошуку категорій по всім полям;

– void Update() – функція обновлення даних;

– void Dell() – функція видалення категорії.

Для класу Profiley, що відображає вікно профілю користувача якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

При визначені конструкторів класу Profiley були обрані параметри:

– profiley (int id) – конструктор з параметрами класу Profiley;

– void OnPickPhotoButtonClicked() – функція завантаження картинки з галереї смартфону;

– void Edit_email() – функція переходу до форми редагування електронної адреси;

– void Edit_phone() – функція переходу до форми редагування номера телефону.

При оголошенні класу Target, що відображає вікно цілей у якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

При визначені конструкторів класу Target були обрані параметри:

– target (int id) – конструктор з параметрами класу Target;

– void Add_ Target() – функція додання цілі;

– void Sort_Name() – функція сортування цілей по назві;

– void Sort_Categories() – функція сортування цілей по назві категорії;

– void Sort_Point() – функція сортування цілей по балам;

– void Search() – функція пошуку цілей по всім полям;

– void Update() – функція обновлення даних;

– void Dell() – функція видалення цілі.

В оголошенні класу Tasks_motiv, що відображає вікно завдань у якості атрибутів, виступають дані типу string: ConnectionString – рядок

підключення до бази даних MySQL.

При визначені конструкторів класу Tasks_motiv були обрані параметри:

– tasks_motiv (int id) – конструктор з параметрами класу Tasks_motiv;

– void Add_ Tasks () – функція додання завдання;

– void Sort_Name() – функція сортування завдань по назві;

– void Sort_Targets() – функція сортування завдань по назві цілі;

– void Sort_Date() – функція сортування завдань по даті;

– void Search() – функція пошуку завдання по всім полям;

– void Update() – функція обновлення даних;

– void Dell() – функція видалення категорії;

– void Add_Completed() – функція виконання завдання;

– void Not_execute() – функція скасування завдання.

При роботі з вкладками редагування використовуємо Page, що прив’язані до інструмента навігації NavigationPage. Вкладки редагування використовуються, як модальні, тому при їх виклику батьківське вікно повинно буде не активне і не з’являтися доки дочірня вкладка не

Клас Email, що відображає вікно редагування електронної адреси у якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

При визначені конструкторів класу Email були обрані параметри:

– email(int id) – конструктор з параметрами класу Email;

– void Chek_Email() – функція перевірки наявності введеного електронного адреса в базі даних;

– void Edit_email() – функція занесення нового електронного адресу.

Для оголошення класу Phone, що відображає вікно редагування номера телефону у якості атрибутів, виступають дані типу string: ConnectionString – рядок підключення до бази даних MySQL.

В оголошенні та визначені функції члени класу Phone:

– phone (int id) – конструктор з параметрами класу Phone;

– void Chek_Phone() – функція перевірки наявності введеного номера

телефону в базі даних;

– void Edit_phone() – функція занесення нового номера телефону.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

До вхідних даних веб-додатку відносять всі види даних, що потребують зберігання у базі. По-перше, це інформація про зареєстрованих користувачів. Таблиця з всіма користувачами є необхідною для реєстрації, авторизації, зміні паролю, а також пов’язана зі зберіганням повідомлень користувачів. По-друге, це власне повідомлення. У цій таблиці також будуть зберігатися завдання користувачів.

Крім тексту завдання, дати й часу його отримання, в блоці з повідомленням знаходиться ім’я та прізвище користувача учасника, який також бере участь у виконання даного завдання.

До вихідних даних програми відноситиметься інформація, що виводитиметься в результаті обробки запитів користувача.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки даного додатку використовувався ПК з наступними характеристиками:

- тип процесора: процесор з частотою 2.2 ГГц;
- ОЗУ об'ємом 4 Гб;
- 4 Гб доступного простору на жорсткому диску;
- жорсткий диск з частотою обертання 5400 об / хв.;
- дозвіл екрану 1024x768;
- клавіатура, маніпулятор “миша”.
- вихід до мережі Інтернет.

2.7.2. Використані програмні засоби

Додаток був розроблений в середовищі програмування MS Visual Studio 2019 на платформі Xamarin, за допомогою мови програмування C#. Мобільний додаток розроблений під операційну систему Android 7.0 та вище. Для розробки мобільного програмного додатку були додатково використані програмні технології для роботи з базою даних SQLlite та віддалений сервер MySQL. Зв'язок з компонентами операційної системи Android був досягнутий за допомогою технології Mono, що вбудована в платформу Xamarin.

2.7.3. Виклик та завантаження програми

Для роботи розробленої програми потрібно мати смартфон, що працює на операційній системі Android та мати підключення до мобільного інтернету для взаємодії з іншими користувачами. Також в налаштування смартфона дозволити взаємодію додатку з модулем оповіщення телефону [9].

Для роботи даного додатку необхідно мобільний пристрій (смартфон), що відповідає таким вимогам:

– операційна система Android версії Android 5.0 Lollipop і вище;
– оптимальна роздільна здатність дисплею – 1280*720.

Для користування додатком достатньо встановити app файл і дозволити програмі присилати вам повідомлення з нагадуваннями.

Готовий мобільний додаток займає близько 50 Мб пам’яті телефону. Для функціонування програми потрібен відкритий доступ до інтернету.

2.7.4. Опис інтерфейсу користувача

Підчас завантаження програми з’являється вікно входу до особистого кабінету (рис. 2.16).

Рис. 2.16. Екран форми входу в особистий кабінет

Мобільний додаток створений, як користувацька програма, яка має кабінети користувачів та можливість реєстрування нових кабінетів. Перший та другий екран реєстрації наведено на рис. 2.17. Передбачена перевірка на існування користувача з введеним логіном або електронним адресом (рис 2.18).

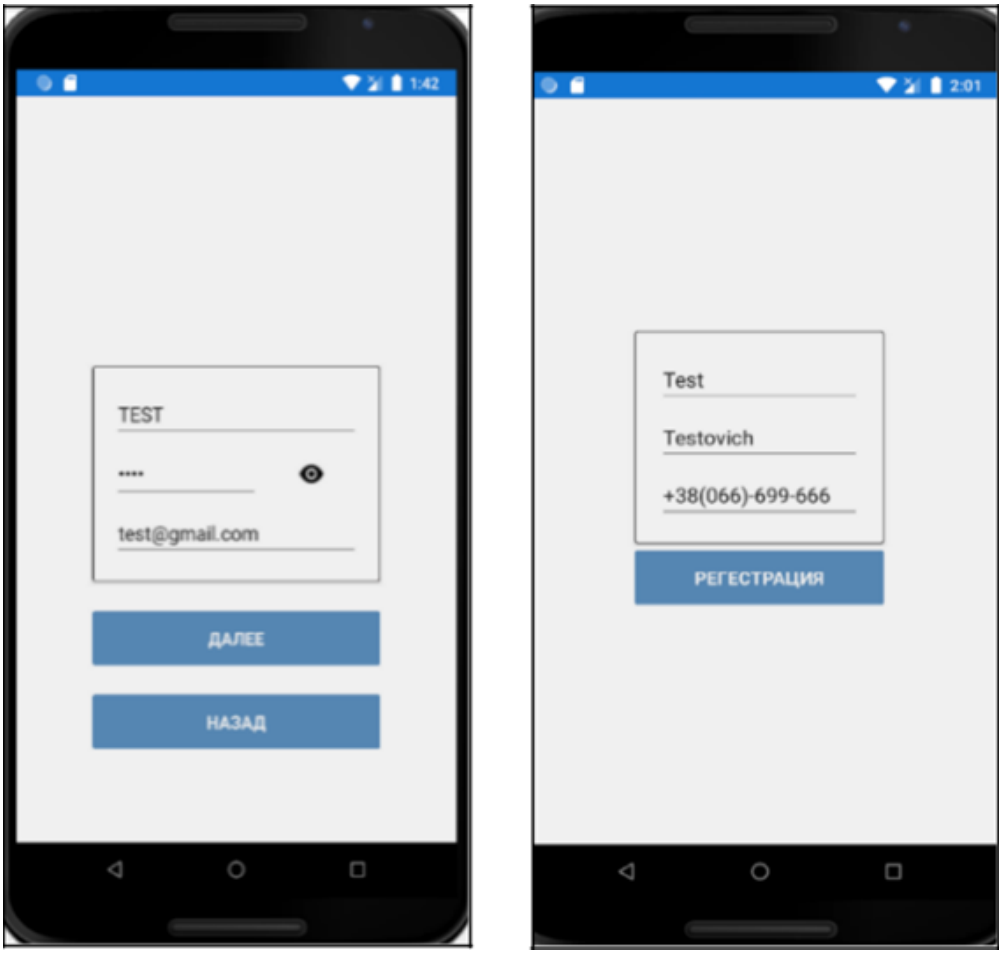

Рис. 2.17. Екрани реєстрації

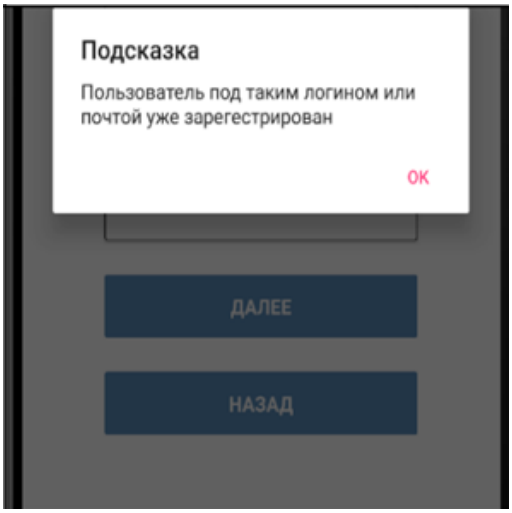

Рис. 2.18. Перевірка наявності користувача

Передбачена перевірка на наявність пустих рядків і реалізована перевірка

на правильність логіну та паролю (рис. 2.19).

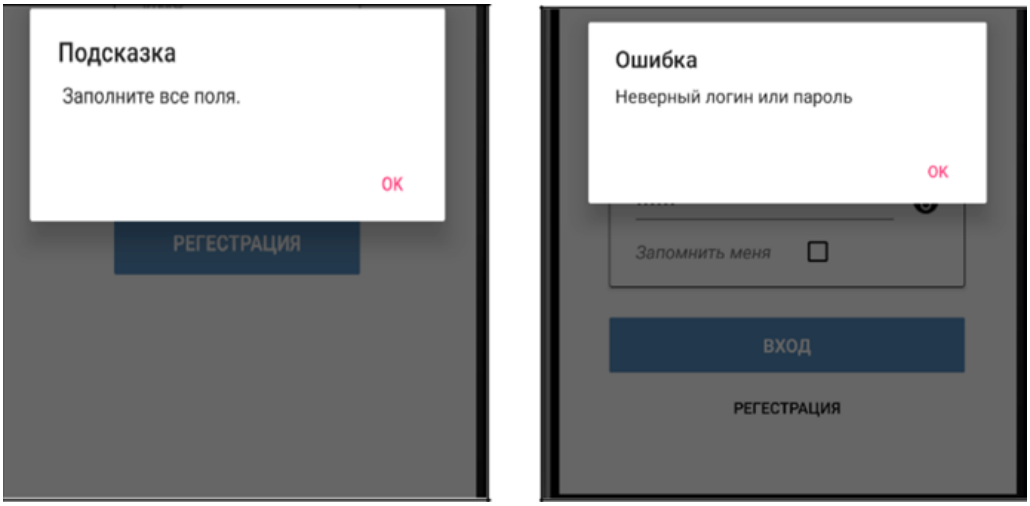

Рис. 2.19. Перевірка на пусті рядки та правильність логіну та паролю

Після успішного входу в особистий кабінет з’являється вікно профілю (рис 2.20).

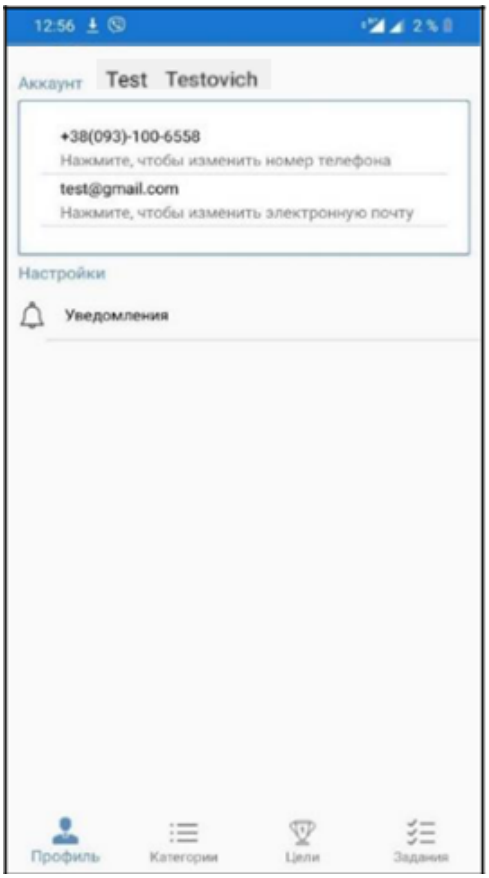

Рис. 2.20. Екран профілю користувача

На екрані профілю передбачено редагування особистих даних та завантаження фотографії профілю (рис. 2.21).

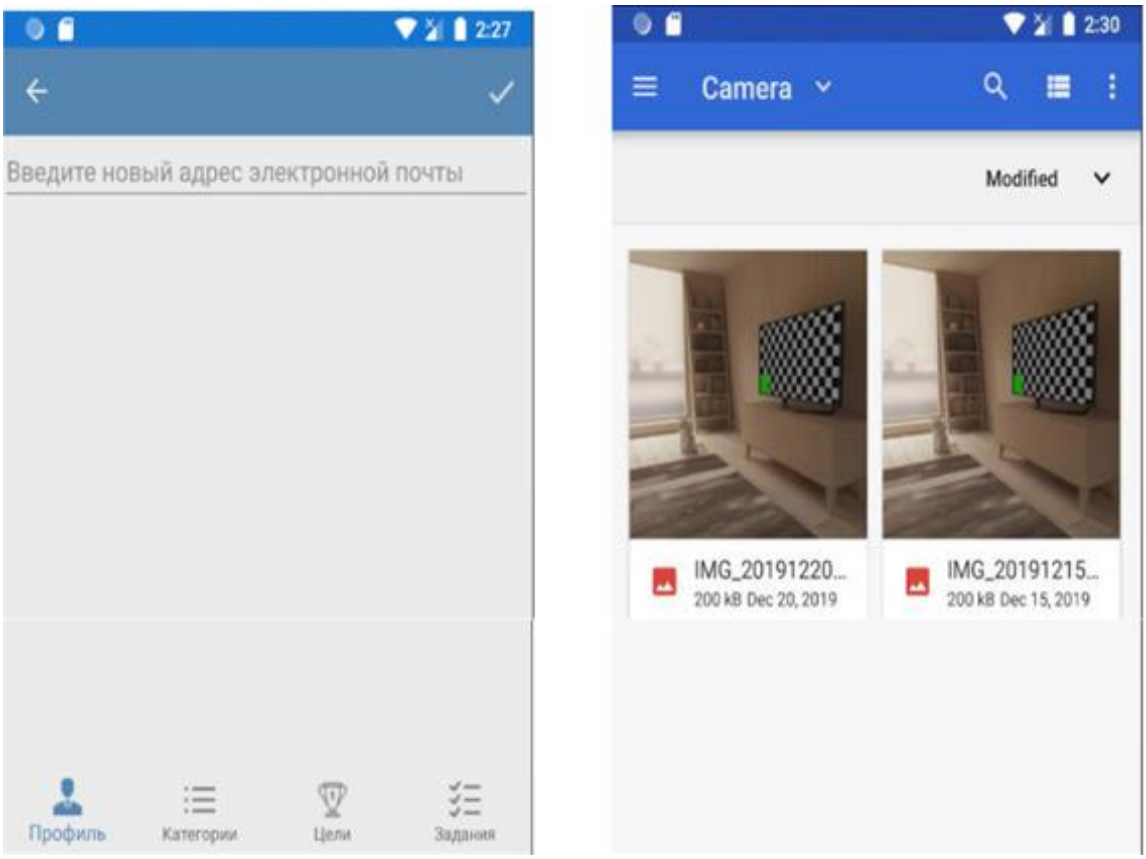

Рис. 2.21. Екран редагування та завантаження фотографії

В даному додатку передбачена можливість створення категорій завдань та завдання балів за виконання завдань з цих категорій. Реалізація додавання, сортування та пошуку показана на рисунках 2.22, 2.23.

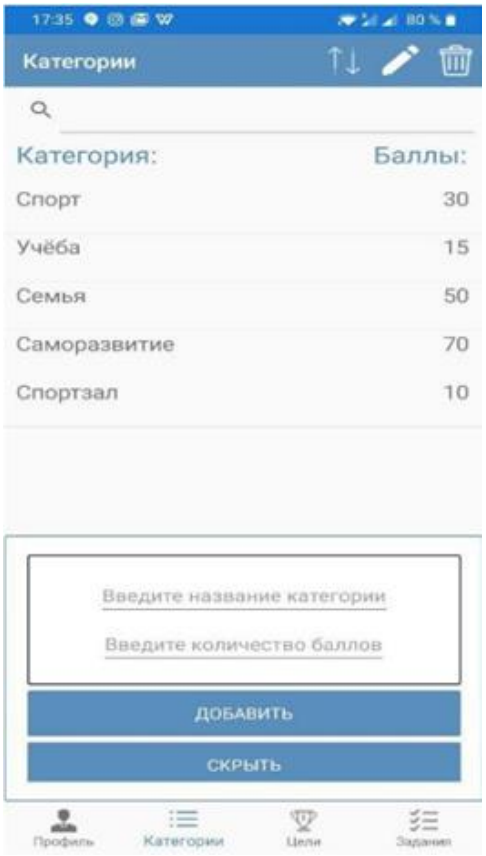

Рис. 2.22. Екран «Категории» в режимі додавання

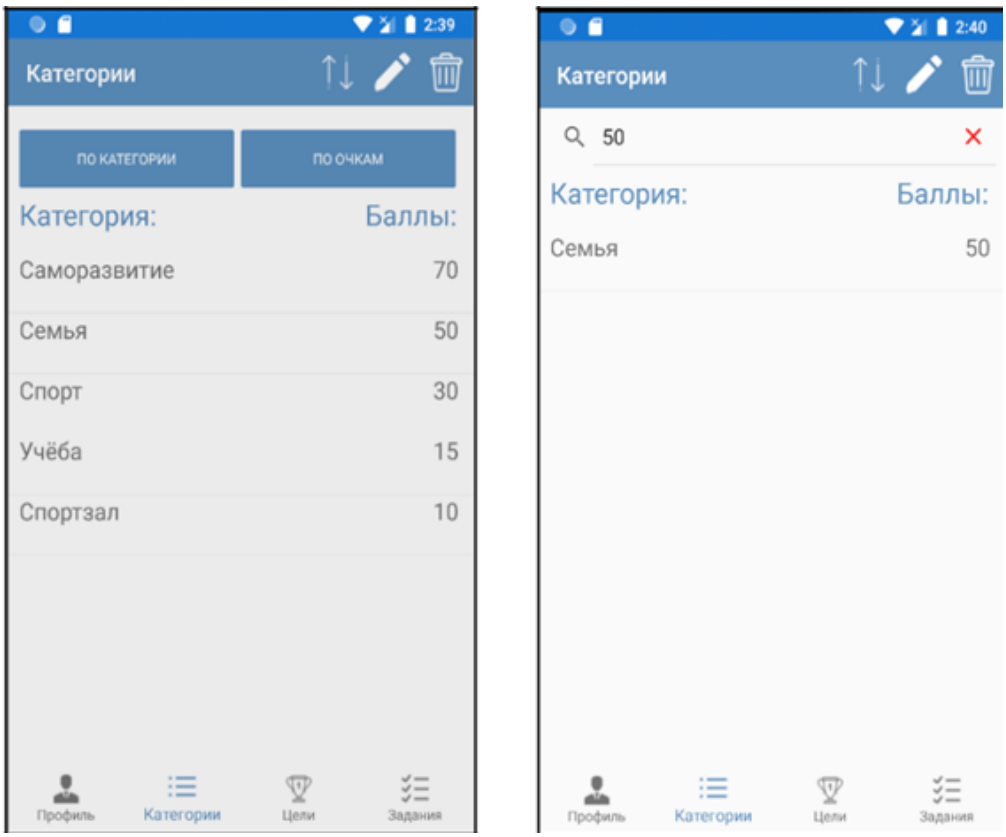

Рис. 2.25. Екран категорії в режимі сортування та пошуку

Також в програмі аналогічно категоріям реалізовані додавання, видалення, пошук та сортування на вкладках «Цели» та «Задания» (рис. 2.26, 2.27).

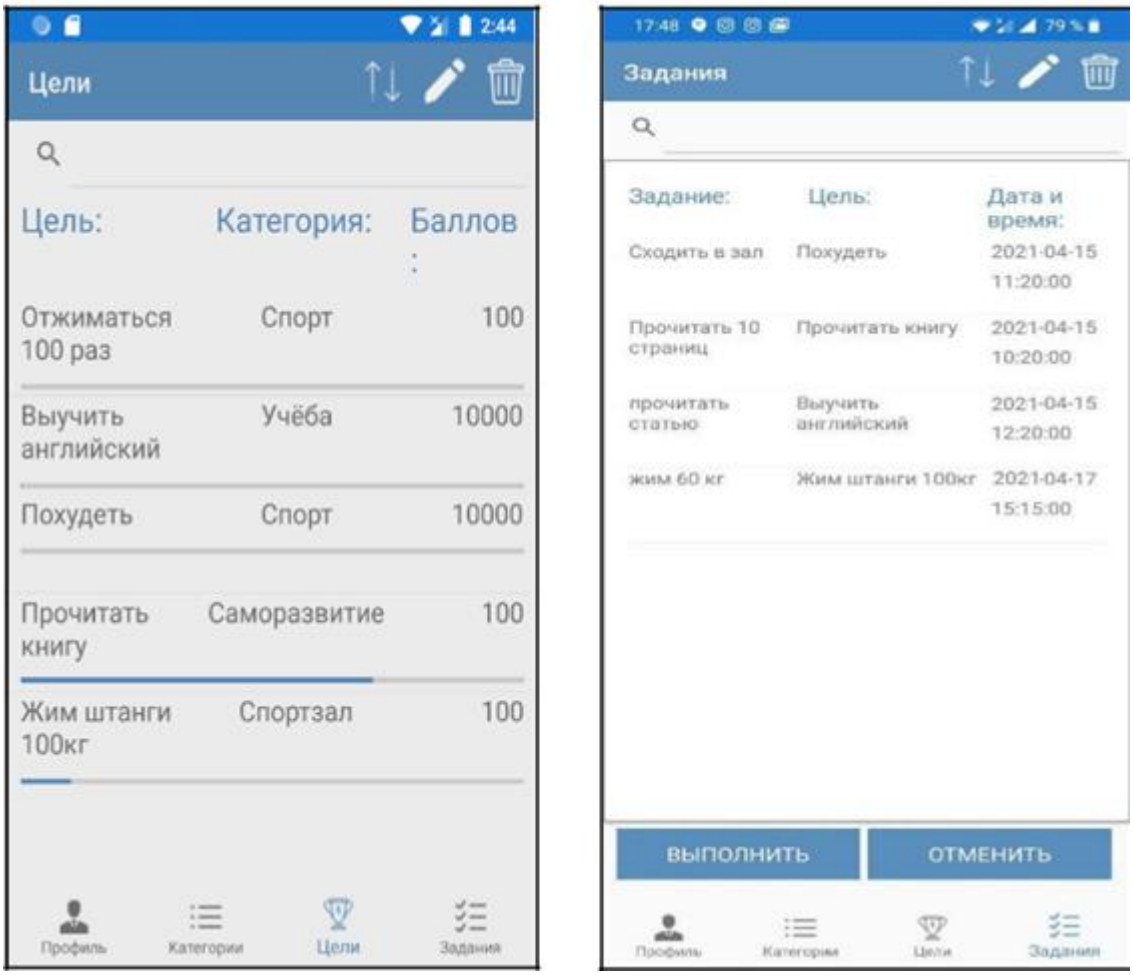

Рис. 2.26. Екран «Цели» в режимі відображення даних

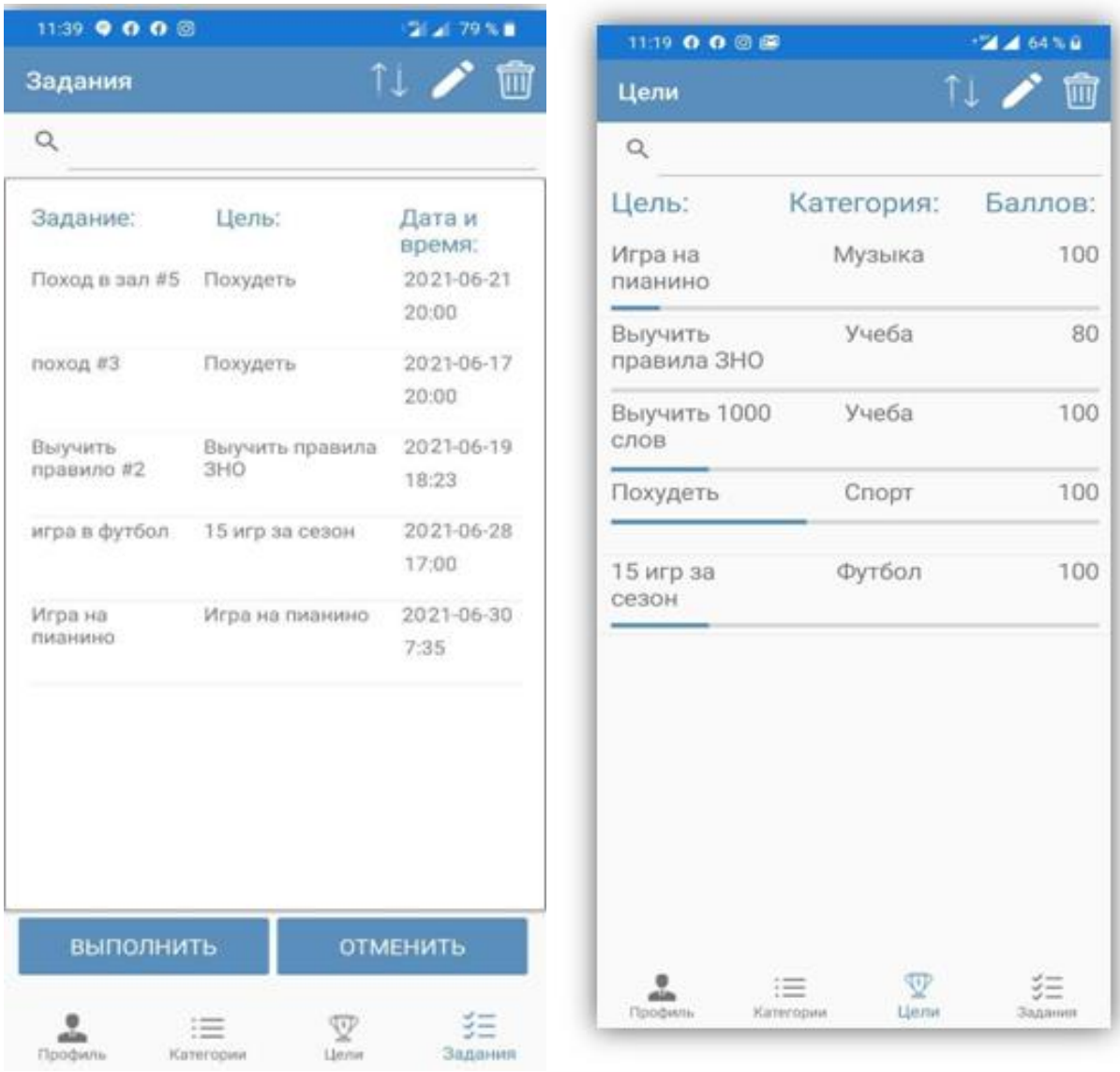

Рис. 2.27 Вікно завдань та стан їх виконання

В даному додатку передбачена можливість створення кімнат для колективного виконання завдання. Головне меню кімнати складається з «Мои комнаты» та «Комнаты» . Адміністратор кімнати може розміщувати завдання та додавати учасників, а також контролювати статус виконання (рис. 2.27 - 2.29).

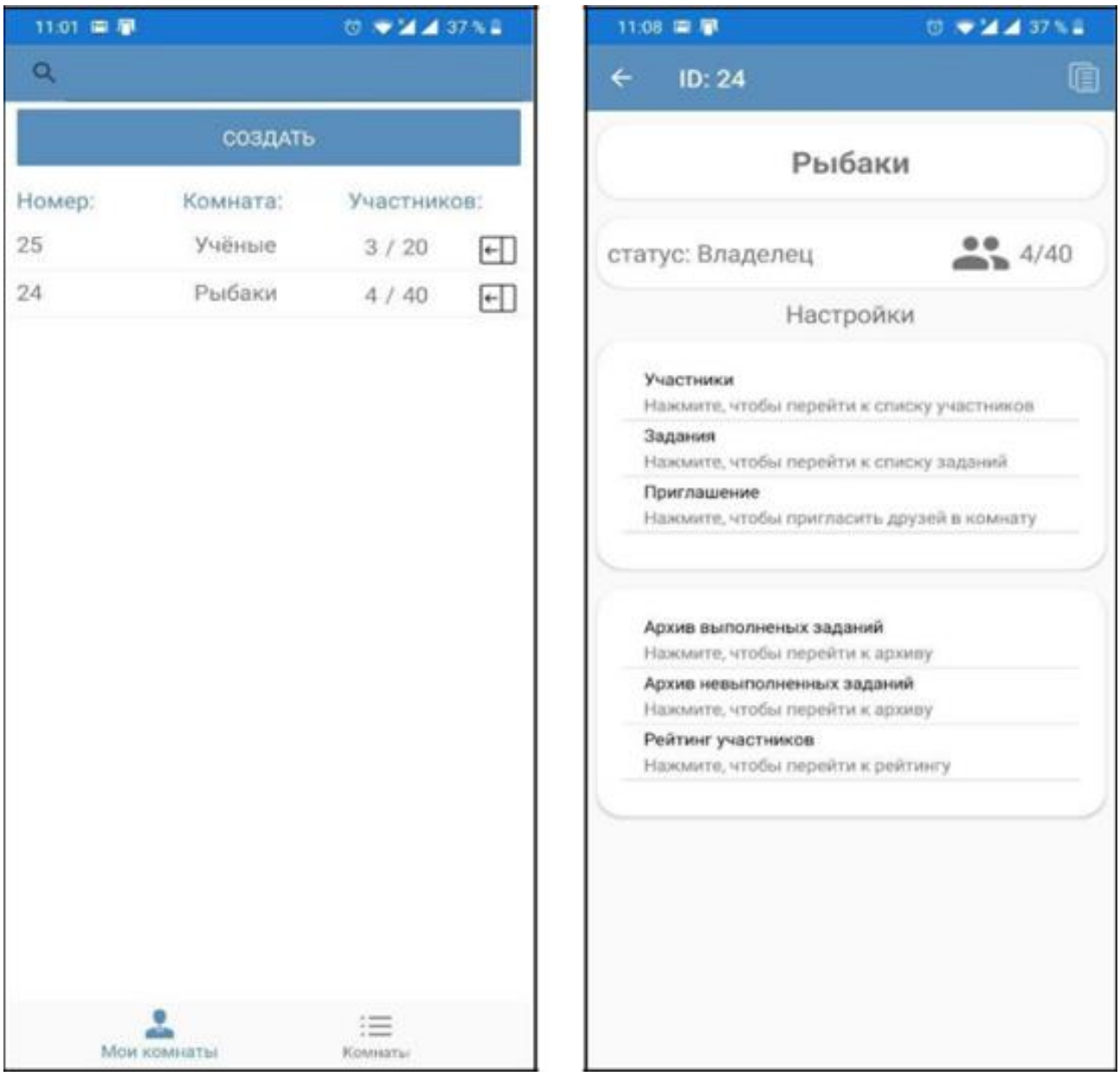

Рис. 2.27. Екран «Кімнат» в режимі відображення та відкритті

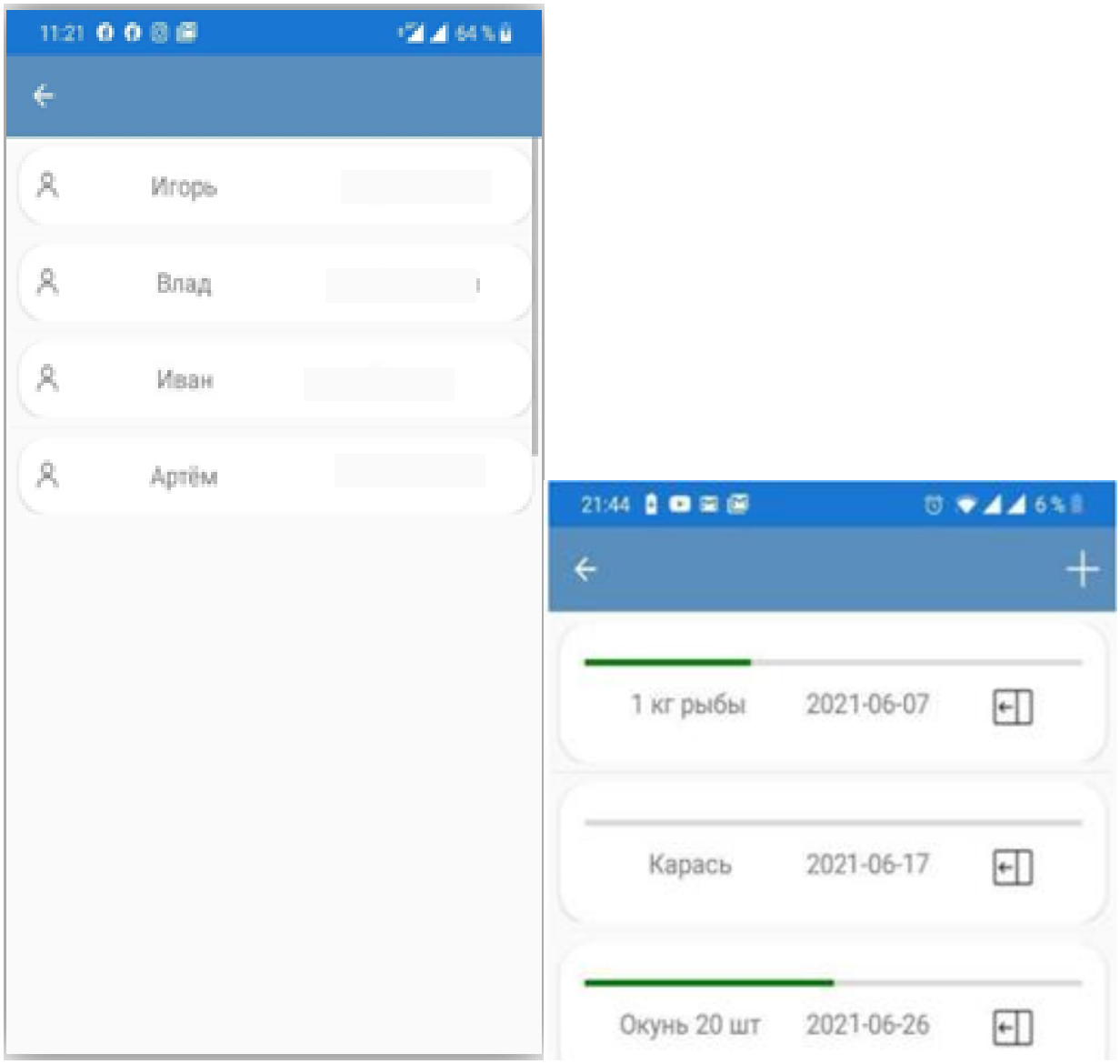

Рис. 2.28. Учасники групового завдання та стан виконання

Після відкриття детальної інформації про завдання адміністратору кімнати дається можливість перегляду списку учасників кімнати, що виконали та не виконали завдання (рис. 2.29).

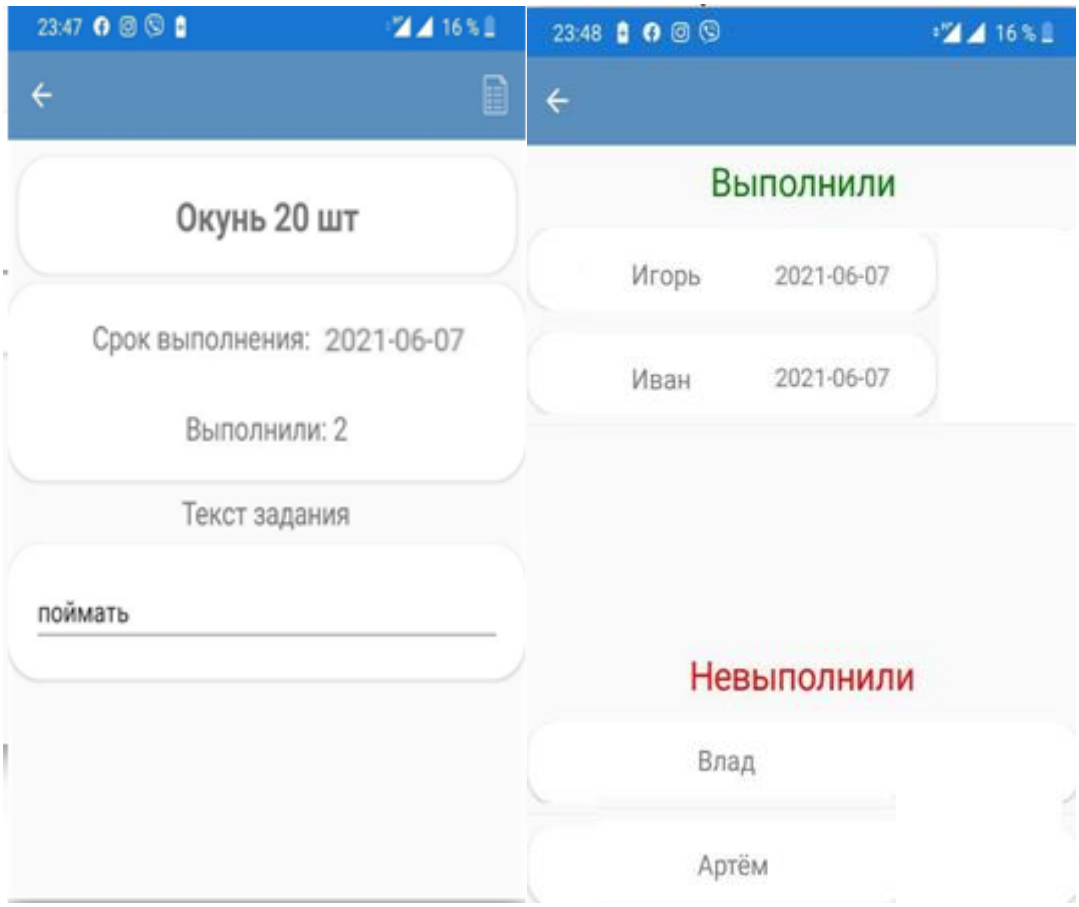

Рис. 2.29. Результат виконання групового завдання

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані розробки програмного забезпечення:

- передбачуване число операторів – 860;
- коефіцієнт складності програми – 1,25;
- коефіцієнт кореляції програми в ході її розробки – 0,1;
- середня годинна заробітна плата програміста, грн/год – 40;
- вартість машино-години ЕОМ, грн/год – 7.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_и + t_a + t_п + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

$t_п$ – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \tag{3.2}$$

де q – передбачуване число операторів,

С– коефіцієнт складності програми,

р– коефіцієнт кореляції програми в ході її розробки.

$$Q = 860 \cdot 1{,}25 \cdot (1 + 0{,}1) = 1182;$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \tag{3.3}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1.2 ... 1.5,

К – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 0,8.

$$t_u = \frac{1182 \cdot 1{,}2}{85 \cdot 0{,}8} = 22, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \tag{3.4}$$

$$t_a = \frac{1182}{20 \cdot 0{,}8} = 73, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}; \quad (3.5)$$

$$t_n = \frac{1182}{25 \cdot 0{,}8} = 59, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}; \quad (3.6)$$

$$t_{отл} = \frac{1182}{4 \cdot 0{,}8} = 369, \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \quad (3.7)$$

$$t_{отл}^{к} = 1{,}2 \cdot 369 = 443, \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}; \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K}; \quad (3.9)$$

$$t_{др} = \frac{1182}{15 \cdot 0{,}8} = 98, \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}; \quad (3.10)$$

$$t_{до} = 0{,}75 \cdot 98 = 73$$, людино-годин.

$$t_{д} = 98 + 73 = 172$$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +22 + 73 +59 + 443 +172 = 821, людино-годин.

В результаті ми розрахували, що в загальній складності необхідно 821 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}$$, грн, (3.11)

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}$$, грн, (3.12)

де t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 821 \cdot 40 = 32843$$,грн.

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}$$, грн, (3.13)

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 443 \cdot 7 = 3104 \text{, грн.}$$

$$К_{ПО} = 32843 + 3104 = 35947 \text{ , грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, мес. ,} \qquad (3.14)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{821}{1 \cdot 176} = 4{,}6 \text{ міс.}$$

Таким чином, очікувана тривалість розробки складе 4,6 місяця, а витрати на створення програмного забезпечення 35947 грн.

ВИСНОВКИ

Метою розробки програмного мобільного додатку «The Motivation», є створення функціоналу, що надає змогу користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання.

Головною особливістю додатку є наявність статистики виконаних чи не виконаних задач та групові кімнати. Статистика розробленого програмного додатку показує успішність людини (користувача) і її витрачені зусилля на досягнення конкретних цілей.

В розробленому додатку реалізовано:

– особистий кабінет користувача де відображаються отримані повідомлення та запрошення інших користувачів програми;

– створення здач де користувач може ставити заплановані задачі;

– формування статистики, яка дозволяє користувачеві контролювати себе і стежити за своїм просуванням до цілі;

– спільні завдання і цілі для команди користувачів у групових кімнатах, які можна ставити для групового виконання.

Під час виконання завдання кваліфікаційної роботи було розроблено мобільний додаток «The Motivation». Додаток був розроблений в середовищі програмування MS Visual Studio 2019 на платформі Xamarin, за допомогою мови програмування C#. Мобільний додаток розроблений під операційну систему Android 7.0 та вище. Для розробки мобільного програмного додатку були додатково використані програмні технології для роботи з базою даних SQLlite та віддалений сервер MySQL. Зв'язок з компонентами операційної системи Android був досягнутий за допомогою технології Mono, що вбудована в платформу Xamarin.

Практичне значення полягає у створенні програмного додатка, що надає можливість користувачеві ставити (створювати) свої задачі (цілі) і визначати термін їх виконання. Така взаємодія гаджету з власником смартфону дозволить користувачу контролювати саморозвиток і постійно вдосконалюватись.

Подібні розробки для смарт гаджетів, за допомогою яких користувач (людина) може спостерігати і контролювати стан свого життя, встановлювати різноманітні цілі та задачі та одержувати оцінку в результаті їх виконання, чи не виконання можуть знайти широкий попит серед активних, сучасних користувачів.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 4,6 місяця, трудомісткість розробки ПЗ – 821 людино-годин, а витрати на її створення програмного забезпечення − 35947 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пахомов Б.И. Самоучитель C/C++ и Borland C++ Builder 2006. – Санкт-Петербург: БХВ-Петербург, 2008. – 670 с.

2. Архангельський Я.А. Программирование в C++ Builder. – Москва: Бином, 2010. – 1304 с.

3. Стаття з інтернет-ресурсу URL: https://habr.com/ru/. Xamarin.Forms. дата звернення: 15.04.2021.

4. Стаття з інтернет-ресурсу URL: http://v-androide.com. Android vs IOS. дата звернення: 15.04.2021.

5. Стаття з інтернет-ресурсу URL: https://metanit.com/sharp/ SQLite.php/.SQLite использование. дата звернення: 15.04.2021.

6. Культин Н.Б. C++ Builder в задачах и примерах. – Санкт- Петербург: БХВ-Петербург, 2005. – 336 с.

7. Стаття з інтернет-ресурсу URL: uk.wikipedia. org/wiki. Xamarim.Android. дата звернення: 15.04.2021.

8. Стаття з інтернет-ресурсу URL: https://servak.com.ua/servers/. MVVM. дата звернення: 15.04.2021.

9. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

10. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.04.2021.

11. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство

СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.

12. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

13. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

14. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

15. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

16. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

17. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

18. Харди, Б. Программирование под Android. Для профессионалов : [пер. с англ.] / Брайан Харди, Билл Филлипс. – СПб. : Питер, 2014. – 592 с. – (Для профессионалов).

19. Цехнер, М. Программирование игр под Android / Марио Цехнер. – СПб. : Питер, 2013. – 688 с.

20. Benjamin Pierce. Types and Programming Languages. — MIT Press, 2002. – 221с.

ДОДАТОК А

КОД ПРОГРАМИ

Клас Model.cs

```
using App1.Class;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Text;
using Xamarin.Forms;
using MySql.Data.MySqlClient;
using System.Data;
using System.Windows.Input;
using App1.MainFile;
using App1.MainFile.Add_Rooms;
namespace App1.ViewModel
{
public class Models : INotifyPropertyChanged
{
int ID;
private string ConnectionString;
int idcat = -1;
Models models;
public Models(int id)
{
models = this;
ConnectionString = @"server = nj2****2.mysql.ukraine.com.ua; Port = 3306; database
=nj297****d; user id = ny****2***ad; password = *****vNjM; charset = utf8"; ID = id;
Categors = new ObservableCollection<Categor>(); Targets = new
ObservableCollection<Targets>(); Task = new ObservableCollection<Task>(); Rooms = new
ObservableCollection<Rooms>();
Member_Magazine = new ObservableCollection<Member_Magazine>(); Rooms_US = new
ObservableCollection<Rooms>();
Task_Room = new ObservableCollection<Task_Room>();
Tasks_Room_Complated = new ObservableCollection<Tasks_Room_Complated>();
Tasks_Room_Complated_not = new ObservableCollection<Tasks_Room_Complated>();
Task_Room_ = new ObservableCollection<Task_Room>();
Task_Room_NOT_ = new ObservableCollection<Task_Room>();
Rating_User = new ObservableCollection<Tasks_Room_Complated>(); Invitation_ = new
ObservableCollection<Invitation>(); Select_Categor();
Select_Tagrets(); Select_Task(); Select_Rooms(); Select_Number_of_users();
Select_Rooms_US(); Select_Invitation();
///Категории
Add_Categor = new Command(addcategor); Dell = new Command(Dell_cat);
Sort = new Command(sort_cat);
Sort_Cat = new Command(sort_name_cat);
Sort_Point = new Command(sort_point_cat);
```

```
Add_Visible = new Command(AddVisible);
///Цели
Add_Visible_Target = new Command(Add_Tagrets_Visible);
Add_Target = new Command(AddTerget);
Dell_Target = new Command(DellTarget);
Sort_Target = new Command(sort_tag);
Sort_Tag_Name = new Command(sort_targ_name);
Sort_Tag_Cat = new Command(sort_targ_cat);
Sort_Tag_Point = new Command(sort_trag_point);
///Задания
Add_Visible_Task = new Command(Add_Task_Visible);
Add_Tasks = new Command(AddTasks);
Add_Completed = new Command(AddCompleted);
Not_execute = new Command(Notexecute);
Dell_Task = new Command(DellTask);
Sort_Task = new Command(SortTask);
Sort_Task_Name = new Command(sort_task_name);
Sort_Task_Tag = new Command(sort_task_tag);
Sort_Task_Date = new Command(sort_task_date);
///Комнаты
Add_Rooms = new Command(AddRooms);
Visible_password_add_room = new Command(Visible_password_add_room_); Add_Room
= new Command(AddRoom);
Open_List_of_room_user = new Command(Open_List_of_room_user_);
Open_List_of_room_user_alien = new Command(Open_List_of_room_user_alien_);
Input_Room = new Command(InputRoom);
Open_Room = new Command(OpenRoom);
Open_My_Task = new Command(Open_My_Task_);
Open_Add_Task_Room = new Command(Open_Add_Task_Room_);
Add_Task_Room = new Command(Add_Task_Room_);
Open_List_Task_Alien = new Command(Open_List_Task_Alien_);
Complate_Task = new Command(Complate_Task_);
Open_MyTask_Archive = new Command(Open_MyTask_Archive_);
Open_Archibve_Complated = new Command(Open_Archibve_Complated_);
Open_Archibve_NOT_Complated = new Command(Open_Archibve_NOT_Complated_);
Open_Rating_User = new Command(Open_Rating_User_);
Open_Invitation = new Command(Open_Invitation_); Send_invitation = new
Command(Send_invitation_);
}
public Models() { }
public event PropertyChangedEventHandler PropertyChanged;
public void OnPropertyChanged([CallerMemberName] string name = "")
{
if (PropertyChanged != null)
PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
}
public string name { get; set; }
public string point { get; set; }
public string IdCategor { get; set; }
public string Name
{
get { return name; }
```

```
set
{
name = value;
OnPropertyChanged();
}
}
public string Point
{
get { return point; }
set
{
point = value;
OnPropertyChanged();
}
}
public string CategorID
{
get { return IdCategor; }
set
{
IdCategor = value;
OnPropertyChanged();
}
}
public Command Add_Categor { get; set; } //Добавить категорію
 public Command Dell { get; set; } //Удалить категорию
public Command Sort { get; set; } //Сортировка отображение
 public Command Sort_Cat { get; set; } //Сортировка по названию
public Command Sort_Point { get; set; } //Сортировка по балам
public Command Add_Visible { get; set; } //Добавление отображение
ObservableCollection<Categor> categors;
public ObservableCollection<Categor> Categors
{
get
{
return categors;
}
set
{
if (categors != value)
{
categors = value;
OnPropertyChanged();
}
}
}
///       <summary>
///       Отображение
///       </summary>
void Select_Categor()
{
using (var connection = new MySqlConnection(ConnectionString))
```

```
{
using (var command = new MySqlCommand("Select_Categories", connection))
{
Categors.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Categor(rdr[0].ToString(), rdr[1].ToString(),
rdr[2].ToString());
Categors.Add(item);
}
rdr.Close();
connection.Close();
}
}
}
/// <summary>
/// Поиск
/// </summary>
private string Search__;
public string SearchedText
{
get
{ return Search__; }
set
{
Search__ = value;
SearchCommandExecute();
OnPropertyChanged();
}
}
private void SearchCommandExecute()
{
if (SearchedText.Length != 0)
{
using (MySqlConnection connection = new MySqlConnection(ConnectionString))
{
using (MySqlCommand command = new MySqlCommand("Search_Categories",
connection))
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("ID", ID); command.Parameters.AddWithValue("Text",
SearchedText); Categors = new ObservableCollection<Categor>(); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
{
var item = new Categor(rdr[0].ToString(), rdr[1].ToString(),
rdr[2].ToString());
Categors.Add(item);
```

```
}
rdr.Close();
connection.Close();
}
}
}
else Select_Categor();
}
///     <summary>
///     Передача ид с listview
///     </summary>
public Categor c_Select_categor;
public Categor SelectedItem
{
get
{     return c_Select_categor;     }     set
{
if (c_Select_categor != value)
{
c_Select_categor = value;
idcat = int.Parse(c_Select_categor.IdCategor);
OnPropertyChanged();
}
}
}
///     <summary>
///     Удалить
///     </summary>
async void Dell_cat()
{
if (idcat >= 0)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Dell_Categories", connection))
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdCategories", idcat); command.ExecuteNonQuery();
}
Frame_Categot = false;
idcat = -1;
Select_Categor();
connection.Close();
}
}
else await Application.Current.MainPage.DisplayAlert("Уведомления", "Выберите эле-
мент!", "Ок");
}
///     <summary>
///     Добавить
///     </summary> void addcategor()
```

```
        {
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command = new MySqlCommand("Insert_Categories",
connection)) {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("Name_categories", Name);
command.Parameters.AddWithValue("Points", Point); int result = command.ExecuteNonQuery();
        Name = "";
        Point = "";
        connection.Close();
        Frame_Categot = false;
        }
        }
        Select_Categor();
        }
        ///     <summary>
        ///     Сортировка
        ///     </summary>
        public bool flag_sort = false;
        bool Sort_IsVisible = false;
        bool Search_IsVisible = true;
        public bool SortIsVisible
        {
        get { return Sort_IsVisible; }
        set
        {
        Sort_IsVisible = value;
        OnPropertyChanged();
        }
        }
        public bool SearchIsVisible
        {
        get { return Search_IsVisible; }
        set
        {
        Search_IsVisible = value;
        OnPropertyChanged();
        }
        }
        public void sort_cat()
        {
        if (flag_sort == false)
        {
        SortIsVisible = true;
        SearchIsVisible = false;
        flag_sort = true;
        Select_Categor();
        }
        else if (flag_sort == true)
```

```
{
SortIsVisible = false;
SearchIsVisible = true;
flag_sort = false;
Select_Categor();
}
}
bool flag_ = false;
bool flag_1 = false;
public void sort_name_cat()
{
if (flag_ == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Name_Cat_DESC", connection))
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
Categors = new ObservableCollection<Categor>();
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Categor(rdr[0].ToString(), rdr[1].ToString(),
rdr[2].ToString());
Categors.Add(item);
}
rdr.Close();
connection.Close();
}
}
flag_ = true;
}
else if (flag_ == true)
{
Select_Categor();
flag_ = false;
}
}
public void sort_point_cat()
{
if (flag_1 == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Point_DESC", connection))
{
Categors.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
```

```
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Categor(rdr[0].ToString(), rdr[1].ToString(),
rdr[2].ToString());
Categors.Add(item);
}
rdr.Close();
connection.Close();
}
}
flag_1 = true;
}
else if (flag_1 == true)
{
Select_Categor();
flag_1 = false;
}
}
bool frame = false;
public bool Frame_Categot
{
get { return frame; }
set
{ frame = value;OnPropertyChanged();
}
}
bool flag_frame = false;
void AddVisible()
{
if (flag_frame == false)
{
Frame_Categot = true;
flag_frame = true;
}
else if (flag_frame == true)
{
Frame_Categot = false;
flag_frame = false;
}
}
/// <summary>
/// // ЦЕЛИ
/// </summary>
public string IdTarget { get; set; }
public string IdCategories { get; set; }
public string Name_Target { get; set; }
public string Point_Target { get; set; }
public string Complate_Point { get; set; }
public Command Add_Visible_Target { get; set; } //отображение добавления
public Command Add_Target { get; set; } // добавлене цели
```

```
public Command Dell_Target { get; set; } // удаление цели
public Command Sort_Target { get; set; } //Отобразить сортировку
public Command Sort_Tag_Name { get; set; }
public Command Sort_Tag_Cat { get; set; }
public Command Sort_Tag_Point { get; set; }
ObservableCollection<Targets> targets;
public ObservableCollection<Targets> Targets
{
get	{	return targets;
}set
{
if (targets != value)
{
targets = value; OnPropertyChanged();
}
}
}
public string IDTarget
{
get { return IdTarget; }
set
{
IdTarget = value;
OnPropertyChanged();
}
}
///	<summary>
///	Отобрвжение Целей
///	</summary>
void Select_Tagrets()
{
try
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Select_Terget", connection))
{
Targets.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Targets(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString());
Targets.Add(item);
}
rdr.Close();
connection.Close();
}
}
```

```
}
catch (Exception ex) { ex.ToString(); }
}
///        <summary>
///        Отображение добавления
///        </summary>
bool frame_tagret = false;
public bool Frame_Target
{
get { return frame_tagret; }
set
{
frame_tagret = value;
OnPropertyChanged();
}
}
bool flag_frame_target = false;
void Add_Tagrets_Visible()
{
if (flag_frame_target == false)
{
Frame_Target = true;
flag_frame_target = true;
}
else if (flag_frame_target == true)
{
Frame_Target = false;
flag_frame_target = false;
}
}
///        <summary>
///        Получить ид с Picker
///        </summary>
public Categor index_target;
public Categor IndexChanged_Targrt
{
get
{        return index_target;    }
set
{
if (index_target != value)
{
index_target = value;
//        id_insex_target = int.Parse(index_target.IdCategor); OnPropertyChanged();
}
}
}
///        <summary>
///        Добавить цель
///        </summary>
///
async void AddTerget()
```

```
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Insert_Target", connection))
{
connection.Open();
try
{

if (int.Parse(POINT_Target) > 0 && int.Parse(POINT_Target) <= 100 &&
POINT_Target.Length > 0)
{
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdCategories", In-dexChanged_Targrt.IdCategor);
command.Parameters.AddWithValue("Name_Target", NAME_Target);
command.Parameters.AddWithValue("Point_Target", POINT_Target);
command.ExecuteNonQuery();
NAME_Target = "";
POINT_Target = "";
Frame_Target = false;
connection.Close();
}
else await Application.Current.MainPage.DisplayAlert("Уведомления", "Количество
баллов должно быть от 1 до 100!", "Ок");
}
catch { await Application.Current.MainPage.DisplayAlert("Уведомления",
"Неправильный ввод данных. Все поля должны бать заполнены", "Ок"); }
}
}
Select_Categor();
Select_Tagrets();
//      Select_Task();
}
///     <summary>
///     Получить ид для удаления
///     </summary>
int idtag = -1;
public Targets c_Select_target;
public Targets SelectedItem_Target
{
get
{
return c_Select_target;
}
set
{
if (c_Select_target != value)
{
c_Select_target = value;
OnPropertyChanged(c_Select_target.IdTarget);
idtag = int.Parse(c_Select_target.IdTarget);
```

```
        }
        }
        }
        ///     <summary>
        ///     Удалить цель
        ///     </summary>
        async void DellTarget()
        {
        Frame_Target = false;
        if (idtag >= 0)
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Dell_Target", connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdTarget", idtag); command.ExecuteNonQuery();
        }
        Select_Tagrets();
        connection.Close();
        idtag = -1;
        Frame_Target = false;
        }
        }
        else await Application.Current.MainPage.DisplayAlert("Уведомления", "Выберите эле-
мент!", "Ок");
        }
        ///     <summary>
        ///     Поиск по целям
        ///     </summary>
        private string Search_tag;
        public string Search_Targert
        {
        get
        {
        return Search_tag;
        }
        set
        {
        Search_tag = value;
        OnPropertyChanged();
        SearchCommandExecute_Tagret();
        }
        }
        void SearchCommandExecute_Tagret()
        {
        if (Search_Targert.Length != 0)
        {
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command = new MySqlCommand("Search_Target", connection))
```

```
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("ID", ID); command.Parameters.AddWithValue("Text",
Search_Targert); Targets = new ObservableCollection<Targets>(); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
{
var item = new Targets(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString());
Targets.Add(item);
}
rdr.Close();
connection.Close();
}
}
}
else Select_Tagrets();
}
///     <summary>
///     Сортировка
/// </summary>
public bool flag_sort_tag = false;
bool Sort_IsVisible_tag = false;
bool Search_IsVisible_tag = true;
public bool SortIsVisible_Target
{
get { return Sort_IsVisible_tag; }
set
{
Sort_IsVisible_tag = value;
OnPropertyChanged();
}
}
public bool SearchIsVisible_Target
{
get { return Search_IsVisible_tag; }
set
{
Search_IsVisible_tag = value; OnPropertyChanged();
}
}
void sort_tag()
{
if (flag_sort_tag == false)
{
SortIsVisible_Target = true;
SearchIsVisible_Target = false;
flag_sort_tag = true;
Select_Tagrets();
}
else if (flag_sort_tag == true)
{
```

```
SortIsVisible_Target = false;
SearchIsVisible_Target = true;
flag_sort_tag = false;
Select_Tagrets();
}
}
bool flag_name = false;
bool flag_cat = false;
bool flag_point = false;
void sort_targ_name()
{
if (flag_name == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Target_Name", connection))
{
Targets.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Targets(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString());
Targets.Add(item);
}
rdr.Close();
connection.Close();
}
}
flag_name = true;
}
else if (flag_name == true)
{
Select_Tagrets();
flag_name = false;
}
}
void sort_targ_cat()
{
if (flag_cat == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Target_Cat", connection))
{
Targets.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID); MySqlDataReader rdr =
```

```
command.ExecuteReader(); while (rdr.Read())
        {
        var item = new Targets(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString());
        Targets.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        flag_cat = true;
        }
        else if (flag_cat == true)
        {
        Select_Tagrets();
        flag_cat = false;
        }
        }
        void sort_trag_point()
        {
        if (flag_point == false)
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Sort_Target_Point", connection))
        {
        Targets.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
        //        Targets = new ObservableCollection<Targets>(); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
        {
        var item = new Targets(rdr[0].ToString(), rdr[1].ToString(),
        rdr[2].ToString(), rdr[3].ToString(), rdr[4].ToString()); Targets.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        flag_point = true;
        }
        else if (flag_point == true)
        {
        Select_Tagrets();
        flag_point = false;
        }
        }
        ////////////////////////////////////////Задания
        public string IdTask { get; set; } public string IdTarget_ { get; set; }
        public string Name_Task { get; set; }
        public string Date { get; set; }
```

```
public string Time { get; set; }
ObservableCollection<Task> task;
public ObservableCollection<Task> Task
{
get
{       return task;
}
set
{
if (task != value)
{
task = value;
OnPropertyChanged();
}
}
}
public string IDTask
{
get { return IdTask; }
set     {
IdTask = value;
OnPropertyChanged();
}
}
public string IDTargets
{
get { return IdTarget_; }
set     {
IdTarget_ = value;
OnPropertyChanged();
}
}
public string NAME_Task
{
get { return Name_Task; }
set     {
Name_Task = value;
OnPropertyChanged();
}
}
public string DATE
{
get { return Date; }
set     {
Date = value;
OnPropertyChanged();
}
}
public string TIME
{
get { return Time; }
set     {
```

```
Time = value;
OnPropertyChanged();
}
}
public Command Add_Visible_Task { get; set; }
public Command Add_Tasks { get; set; }
public Command Add_Completed { get; set; }
public Command Dell_Task { get; set; }
public Command Sort_Task { get; set; }
public Command Sort_Task_Name { get; set; }
public Command Sort_Task_Tag { get; set; }
public Command Sort_Task_Date { get; set; }
public Command Search_Task { get; set; }
public Command Not_execute { get; set; }
///     <summary>
///     Отображение
///     </summary>
///
int idTag = -1; //int idTask = -1; void Select_Task()
{
idTag = -1;
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Select_Tasks", connection))
{
Task.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Class.Task(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), DateTime.Parse(rdr[4].ToString()).ToShortTimeString());
Task.Add(item);
}
rdr.Close();
connection.Close();
}
}
}
///     <summary>
///     Отображение добавления
///     </summary>
bool frame_task = false;
public bool Frame_Task
{
get { return frame_task; }
set
{
frame_task = value;
OnPropertyChanged();
```

```
}
}
bool flag_frame_task = false;
void Add_Task_Visible()
{
if (flag_frame_task == false)
{
Frame_Task = true;
flag_frame_task = true;
}
else if (flag_frame_task == true)
{
Frame_Task = false;
flag_frame_task = false;
}
//        Select_Tagrets();
}
///        <summary>
///        Выбор цели с пикера
///        </summary>
public Targets index_task;
public Targets IndexChanged_Task
{
get
{
return index_task;
}
set
{
if (index_task != value)
{
index_task = value;
OnPropertyChanged();
}
}
}
///        <summary>
///        Добавить цель
///        </summary>
DateTime date_task = DateTime.Today;
public DateTime TaskDate
{
get
{
return date_task;
}
set
{
date_task = value; OnPropertyChanged();
}
}
TimeSpan time_task = new TimeSpan();
```

```
public TimeSpan TaskTime
{
get
{
return time_task;
}
set
{
time_task = value;
OnPropertyChanged();          }
}
string title = "Цель";
public string Title_Picker
{
get { return title; }
set { title = value; OnPropertyChanged(); }
}
async void AddTasks()
{
using (var connection = new MySqlConnection(ConnectionString))
{
try
{
if (NAME_Task.Length != 0)
{
using (var command = new MySqlCommand("Insert_Tasks", connection))
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdTarget",IndexChanged_Task.IdTarget);
command.Parameters.AddWithValue("Name_Task", NAME_Task);
command.Parameters.AddWithValue("Date", TaskDate);
command.Parameters.AddWithValue("Time", TaskTime); command.ExecuteNonQuery();
Frame_Task = false;
NAME_Task = "";
Title_Picker = "Цель";
connection.Close();
}
}
else await Application.Current.MainPage.DisplayAlert("Уведомления", "Все поля
должны быть заполенны", "Ок");
}
catch { await Application.Current.MainPage.DisplayAlert("Уведомления", "Все поля
должны быть заполенны", "Ок"); }
}
Select_Task();
Select_Tagrets();
}
int idtask = -1;
///       <summary>
///       Передача ид с listview
```

```
        ///     </summary>
        public Task Item_Task;
        public Task SelectedItem_Task
        {
        get
        {       return Item_Task;       }
        set
        {
        if (Item_Task != value)
        {
        Item_Task = value;
        idtask = int.Parse(Item_Task.IdTask);
        OnPropertyChanged();
        }
        }
        }
        ///     <summary>
        ///     Удалить задание
        ///     </summary>
        async void DellTask()
        {
        //Frame_.IsVisible = false;
        //flag = false;
        //Name_tasks.Text = "";
        //Picker.Title = "Цель";
        //Date.Date = DateTime.Today;
        if (idtask >= 0)
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Dell_Tasks", connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdTasks", idtask); command.ExecuteNonQuery();
        }
        Select_Task();
        connection.Close();
        idtask = -1;
        }
        }
        else await Application.Current.MainPage.DisplayAlert("Уведомления", "Выберите эле-
мент!", "Ок");
        }
        ///     <summary>
        ///     Выполнить задание
        ///     </summary>
        ///
        void AddCompleted()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
```

```
using (var command = new MySqlCommand("Update_Status_1", connection))

{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdTasks", idtask); command.ExecuteNonQuery();
connection.Close();
}
using (var command1 = new MySqlCommand("Update_Complate_Point", connection))
{
connection.Open();
command1.CommandType = CommandType.StoredProcedure;
command1.Parameters.AddWithValue("IdTasks", idtask); command1.ExecuteNonQuery();
connection.Clone();
}
}
idtask = -1;
Select_Task();
Select_Tagrets();
}
void Notexecute()
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Update_Status_0", connection))
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdTasks", idtask); command.ExecuteNonQuery();
connection.Close();
}
}
idtask = -1;
Select_Task();
Select_Tagrets();
}
///     <summary>
///     Сортировка
///     </summary>
public bool flag_sort_task = false;
bool Sort_IsVisible_task = false;
bool Search_IsVisible_task = true;
public bool SortIsVisible_Task
{
get { return Sort_IsVisible_task; }
set
{     Sort_IsVisible_task = value;  OnPropertyChanged();
}
}
public bool SearchIsVisible_Task
{
get { return Search_IsVisible_task; }
```

```
set
{   Search_IsVisible_task = value;
OnPropertyChanged();
}
}
void SortTask()
{
if (flag_sort_task == false)
{
SortIsVisible_Task = true;
SearchIsVisible_Task = false;
flag_sort_task = true;
Select_Task();
}
else if (flag_sort_task == true)
{
SortIsVisible_Task = false;
SearchIsVisible_Task = true;
flag_sort_task = false;
Select_Task();
}
}
bool flag_name_task = false;
bool flag_task_tag = false;
bool flag_task_date = false;
void sort_task_name()
{
if (flag_name_task == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Task_Name", connection))
{
Task.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Class.Task(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), DateTime.Parse(rdr[4].ToString()).ToShortTimeString());
Task.Add(item);
}
rdr.Close();
connection.Close();
}
}
flag_name_task = true;
}
else if (flag_name_task == true)
{
```

```
Select_Task();
flag_name_task = false;
}
}
void sort_task_tag()
{
if (flag_task_tag == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Task_Targ", connection))
{
Task.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Class.Task(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), DateTime.Parse(rdr[4].ToString()).ToShortTimeString());
Task.Add(item);
}
rdr.Close();
connection.Close();
}
}
flag_task_tag = true;
}
else if (flag_task_tag == true)
{
Select_Task();
flag_task_tag = false;
}
}
void sort_task_date()
{
if (flag_task_date == false)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Sort_Task_Date", connection))
{
Task.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Class.Task(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), DateTime.Parse(rdr[4].ToString()).ToShortTimeString());
```

```
Task.Add(item);
}
rdr.Close();
connection.Close();
}
}
flag_task_date = true;
}
else if (flag_task_date == true)
{
Select_Task();
flag_task_date = false;
}
}
private string Search_task;
public string SearchTask
{
get
{       return Search_task;    }
set
{
Search_task = value;
OnPropertyChanged();
SearchCommandExecute_Task();
}
}
void SearchCommandExecute_Task()
{
if (SearchTask.Length != 0)
{
using (MySqlConnection connection = new MySqlConnection(ConnectionString))
{
using (MySqlCommand command = new MySqlCommand("Search_Task", connection))
{
Task.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("ID", ID);
command.Parameters.AddWithValue("Text", SearchTask);
Task = new ObservableCollection<Task>();
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Class.Task(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), DateTime.Parse(rdr[4].ToString()).ToShortTimeString());
Task.Add(item);
}
rdr.Close();
connection.Close();
}
}
}
```

```
else Select_Task();
}
//////////////////////////////////////////////////Комнаты
 public string IdRoom { get; set; } public string Name_Room { get; set; }
public string Password { get; set; }
public string Num_User { get; set; }
ObsevableCollection<Rooms> rooms;
public ObservableCollection<Rooms> Rooms
{
get
{        return rooms;  }
set
{
if (rooms != value)
{
rooms = value;
OnPropertyChanged();
}
}
}
public string IDRoom
{
get { return IdRoom; }
set      {         IdRoom = value;
OnPropertyChanged();
}
}
public string NAME_Room
{
get { return Name_Room; }
set      { Name_Room = value; OnPropertyChanged();
}
}
public string PASSWORD
{
get { return Password; }
set      {
Password = value;
OnPropertyChanged();
}
}
public string NUM_User
{
get { return Num_User; }
set      {
Num_User = value;
OnPropertyChanged();
}
}
////////////////////// Количесто пользователей в комнате
public string idnum { get; set; }
public string name_us { get; set; }
```

```
        ObservableCollection<Number_of_users> number_of_users; public
ObservableCollection<Number_of_users> Number_of_users {
        get
        {
        return number_of_users;
        }
        set
        {
        if (number_of_users != value)
        {
        number_of_users = value;
        OnPropertyChanged();
        }
        }
        }
        public string IDnum
        {
        get { return idnum; }
        set
        {
        idnum = value;
        OnPropertyChanged();
        }
        }
        public string Name_us
        {
        get { return name_us; }
        set
        {
        name_us = value;
        OnPropertyChanged();
        }
        void Select_Number_of_users()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Select_Number_of_users", connection)) {
        connection.Open();
        Number_of_users = new ObservableCollection<Number_of_users>();
command.CommandType = CommandType.StoredProcedure; MySqlDataReader rdr =
command.ExecuteReader();
        while (rdr.Read())
        {
        var item = new Number_of_users(rdr[0].ToString(), rdr[1].ToString());
Number_of_users.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        }
        ////////////////////////////////////////////////////////////
```

```
public Command Add_Rooms { get; set; }
public Command Visible_password_add_room { get; set; } public Command Add_Room { get; set; }
public Command Open_List_of_room_user { get; set; } public Command Open_List_of_room_user_alien { get; set; } public Command Input_Room { get; set; }
public Command Open_Room { get; set; } public Command Open_My_Task { get; set; } public Command Open_Add_Task_Room { get; set; } public Command Send_invitation { get; set; } void AddRooms()
{
App.Current.MainPage.Navigation.PushAsync(new Add_Room(ID, models));
}
public ICommand OpenMyRoom => new Command(Open_MyRoom);
public ICommand OpenAlienRoom => new Command(Open_AlienRoom); public ICommand OpenTask_Room => new Command(OpenTask_Room_);
public ICommand OpenTask_Room_Alien => new Command(OpenTask_Room_Alien_);
///  <summary>
///  Вывод комнат
///  </summary> void Select_Rooms()
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Select_Rooms", connection))
{
Rooms.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure; command.Parameters.AddWithValue("IdUser", ID); MySqlDataReader rdr = command.ExecuteReader(); while (rdr.Read())
{
var item = new Rooms(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(), rdr[3].ToString(), rdr[4].ToString());
Rooms.Add(item);
}
rdr.Close();
connection.Close();
}
}
}
int index_room = -1;
int max_user_room = -1;
void Open_MyRoom(object obj)
{
if (obj is Rooms item)
{
App.Current.MainPage.Navigation.PushAsync(new
Open_My_Room(int.Parse(item.IdRoom), item.Num_User, item.Max_User, item.Name_Room, ID, models));
index_room = int.Parse(item.IdRoom);
max_user_room = int.Parse(item.Num_User);
}
}
///  <summary>
```

```
///        Отображать пароль  при созданнии комнат
///        </summary>
///
bool Visible_password_add_room_flag = true;
public bool Visible_password_add_room_bool
{
get { return Visible_password_add_room_flag; }
set
{
Visible_password_add_room_flag = value; OnPropertyChanged();
}
}
void Visible_password_add_room_()
{
if (Visible_password_add_room_flag == false)
{
Visible_password_add_room_bool = true;
Visible_password_add_room_flag = true;
}
else if (Visible_password_add_room_flag == true)
{
Visible_password_add_room_bool = false;
Visible_password_add_room_flag = false;
}
}
///        <summary>
///        Получить ид с пикера
///        </summary>
public Number_of_users index_number;
public Number_of_users IndexChanged_Number_of_users
{
get
{        return index_number; }
set
{
if (index_number != value)
{
index_number = value;
OnPropertyChanged();
}
}
}
///        <summary>
///        Создать комнату
///        </summary> void AddRoom()
{
try
{
if (PASSWORD.Length >= 6)
{
using (MySqlConnection connection = new MySqlConnection(ConnectionString))
```

```
        {
        using (MySqlCommand command = new MySqlCommand("Insert_Room", connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("Name_Room", NAME_Room);
command.Parameters.AddWithValue("Password", PASSWORD);
        command.Parameters.AddWithValue("Max_user", IndexChanged_Num-
        ber_of_users.name);
        int result = command.ExecuteNonQuery();
        connection.Close();
        }
        }
        Select_Rooms();
        Application.Current.MainPage.Navigation.PushAsync(new          Rooms_page(ID,
  models));
        }
        else Application.Current.MainPage.DisplayAlert("Уведомления", "Длинна пароля дожна
быть больше 6 символов!", "Ок");
        } catch { Application.Current.MainPage.DisplayAlert("Уведомления", "Все поля должны
быть заполнены!", "Ок"); }
        }
        ///     <summary>
        ///     Открыть список участников комнаты
        ///     </summary>
        void Open_List_of_room_user_()
        {
        if (index_room > -1)
        {
        App.Current.MainPage.Navigation.PushAsync(new List_of_room_user(ID, models, in-
        dex_room));
        Select_Member_Magazine(index_room);
        }
        }
        ///     <summary>
        ///     Вывод списка участников
        ///     </summary>
        ///
        public string IdMagazine { get; set; } public string Surname { get; set; } public string
Name_us_room { get; set; } public string IdUser { get; set; }
ObservableCollection<Member_Magazine> member_magazine; public
ObservableCollection<Member_Magazine> Member_Magazine {
        get
        {       return member_magazine;     }
        set
        {
        if (member_magazine != value)
        {
        member_magazine = value;
        OnPropertyChanged();
        }
```

```
}
}
public string IDMagazine
{
get { return IdMagazine; }
set
{IdMagazine = value;
OnPropertyChanged();
}
}
public string SURNAME
{
get { return Surname; }
set { Surname = value;
OnPropertyChanged();
}
}
public string NAME_us_room
{
get { return Name_us_room; }
set {
OnPropertyChanged();
Name_us_room = value;
}
}
public string IDUser
{
get { return IdUser; }
set { IdUser = value;
OnPropertyChanged();
}
}
void Select_Member_Magazine(int index)
{
if (index_room > -1)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Select_Member_Magazine", connection))
{
Member_Magazine.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdRoom", index_room); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
{
var item = new Member_Magazine(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString());
Member_Magazine.Add(item);
}
rdr.Close();
connection.Close();}}}
```

```
        ///     <summary>
        ///     Комнаты в которых участвует пользователь
        ///     </summary>
        ObservableCollection<Rooms> rooms_us;
        public ObservableCollection<Rooms> Rooms_US
        {
        get
        {       return rooms_us;        }
        set
        {       if (rooms_us != value)          {       rooms_us = value;
        OnPropertyChanged();
        }
        }
        }
        ///     <summary>
        ///     Вывод комнат в которых участвует пользователь
        ///     </summary>
        void Select_Rooms_US()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Select_Rooms_US", connection))
        {
        Rooms_US.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
        command.Parameters.AddWithValue("IdUser", ID);
        MySqlDataReader rdr = command.ExecuteReader();
        while (rdr.Read())
        {
        var item = new Rooms(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString());
        Rooms_US.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        }
        ///     <summary>
        ///     Открыть окно входа в комнату
        ///     </summary>
        void InputRoom()
        {
        App.Current.MainPage.Navigation.PushAsync(new Input_Room(ID, models));
        }
        ///     <summary>
        ///     Вход в комнату
        ///
        ///     </summary> void OpenRoom()
        {
        //try
```

```
        //{
        if (PASSWORD.Length >= 6)
        {
        int x = 0;
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command = new MySqlCommand("Chek_Input_Room",
connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("Login", int.Parse(IDRoom));
command.Parameters.AddWithValue("Password", PASSWORD);
command.Parameters.AddWithValue("IdUser", ID); MySqlDataReader reader =
command.ExecuteReader(); while (reader.Read()) x++; reader.Close();
        connection.Close();
        }
        if (x > 0)
        {
        using (MySqlCommand command_ = new
MySqlCommand("Insert_Member_Magazine", connection))
        {
        connection.Open();
        command_.CommandType = CommandType.StoredProcedure;
        command_.Parameters.AddWithValue("IdUser", ID);
        command_.Parameters.AddWithValue("IdRoom", int.Parse(IDRoom));
        int result_ = command_.ExecuteNonQuery();
        connection.Close();
        }
        using (MySqlCommand command__ = new MySqlCommand("Update_Num_User_Room",
        connection))
        {
        connection.Open();
        command__.CommandType = CommandType.StoredProcedure;
        command__.Parameters.AddWithValue("IdUser", ID);
        command__.Parameters.AddWithValue("IdRoom", int.Parse(IDRoom));
        int result_ = command__.ExecuteNonQuery();
        connection.Close();
        }
        Select_Rooms_US();
        Application.Current.MainPage.Navigation.PushAsync(new Rooms_page(ID,
models));
        }
        else { Application.Current.MainPage.DisplayAlert("Ошибка", "Неправильный ввод
логина или пароля!", "Ок"); }
        }
        }
        else Application.Current.MainPage.DisplayAlert("Уведомления", "Длинна пароля дожна
быть больше 6 символов!", "Ок");
        }
        ///     <summary>
        ///     Войти в комнату, как участник
```

```
/// </summary>
void Open_AlienRoom(object obj)
{
if (obj is Rooms item)
{
App.Current.MainPage.Navigation.PushAsync(new Open_Al-
ien_Room(int.Parse(item.IdRoom), item.Num_User, item.Max_User, item.Name_Room, ID,
models));
index_room = int.Parse(item.IdRoom);
}
}
void Open_List_of_room_user_alien_()
{
if (index_room > -1)
{
App.Current.MainPage.Navigation.PushAsync(new List_of_room_user_alien(ID, mod-els,
index_room));
Select_Member_Magazine(index_room);
}
}
///     <summary>
///     Открыть список заданий для админа
///     </summary>
void Open_My_Task_()
{
if (index_room > -1)
{
Select_Task_Room();
App.Current.MainPage.Navigation.PushAsync(new Open_List_Task_My_Room(ID, mod-
els, index_room));
}
}
////////// Задания комнаты
public string IdTaskRoom { get; set; }
public string IdRoom_task { get; set; }
public string Title_Task { get; set; }
public string Text_Room { get; set; }
public string Data { get; set; }
public string Point_Target_task { get; set; }
public string Point_task { get; set; }
public string Complate_Point_task { get; set; }
ObservableCollection<Task_Room> task_room; public
ObservableCollection<Task_Room> Task_Room {
get
{ return task_room; } set
{
if (task_room != value)
{
task_room = value;
OnPropertyChanged();
}
}
```

```
}
public string IDTaskRoom
{
get { return IdTaskRoom; }
set
}
{ IdTaskRoom = value; OnPropertyChanged();
}
public string IDRoom_task
{
get { return IdRoom_task; }
set
{ IdRoom_task = value; OnPropertyChanged();
}
}
public DateTime DATA
{
get { return date_task_Room; }
set      {
date_task_Room = value;
OnPropertyChanged();
}
}
public string POINT_Target_task
{
get { return Point_Target_task; }
set      {
Point_Target_task = value;
OnPropertyChanged();
}
}
public string POINT
{
get { return Point; }
set      {         Point = value
OnPropertyChanged();
}}public string COMPLATE_POINT_Task
{
get { return Complate_Point_task; }
set      { Complate_Point_task = value; OnPropertyChanged();
}
}
public string TITLE_Task
{
get { return Title_Task; }
set      {         Title_Task = value;
         OnPropertyChanged();
}        }         ublic Command Add_Task_Room { get; set; }
public Command       Open_List_Task_Alien { get;          set; }
public Command       Complate_Task { get; set; }
public Command       Open_MyTask_Archive { get; set; }
public Command       Open_Archibve_Complated { get; set; }
```

```
        public Command      Open_Archibve_NOT_Complated   { get; set; }
        public Command      Open_Rating_User { get; set; }
        public Command      Open_Invitation { get; set;    }
        ///     <summary>
        ///     Вывод заданий
        ///     </summary>
        void Select_Task_Room()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Select_Task_Room", connection))
        {
        Task_Room.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
        command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdRoom", index_room); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
        {
        var item = new Task_Room(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString(), rdr[5].ToString(), rdr[6].ToString(), rdr[7].ToString(),
rdr[8].ToString());
        Task_Room.Add(item);
        }
        rdr.Close()     connection.Close();
        ///     <summary>
        ///     Открыть список задний
        ///     </summary>
        int index_task_;
        void OpenTask_Room_(object obj)
        {
        if (obj is Task_Room item)
        //      float progres =float.Parse (item.Complate_Point);
        {
        App.Current.MainPage.Navigation.PushAsync(new
        Open_Task(int.Parse(item.IdTaskRoom),    item.Title_Task,       item.Text_Room,
  item.Point_Target,
        item.Point, item.Complate_Point_Task_Room, item.Data, item.Num_Completed, ID,
models));
        //      index_room = int.Parse(item.IdRoom); index_task_ = int.Parse(item.IdTaskRoom);
        }
        }
        void Open_Add_Task_Room_()
        {
        if (index_room > -1)
        {
        App.Current.MainPage.Navigation.PushAsync(new  Add_Task_My_Room(ID,  models,
index_room));
        }       }
        ///     <summary>
        ///     Создать задание
        ///     </summary>
```

```
        void Add_Task_Room_()
        {
        try
        {
        if (DATA > DateTime.Today)
        {
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command = new MySqlCommand("Insert_Task_Room", con-
        nection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdRoom", index_room);
command.Parameters.AddWithValue("Title_Task", TITLE_Task);
command.Parameters.AddWithValue("Text_Room", TEXT_Room);
command.Parameters.AddWithValue("Data", DATA);
command.Parameters.AddWithValue("Point_Target", max_user_room - 1); int result =
command.ExecuteNonQuery(); connection.Close();
        } } Select_Task_Room(); Application.Current.MainPage.Navigation.PushAsync(new
        Open_List_Task_My_Room(ID, models, index_room));
        }
        else  Application.Current.MainPage.DisplayAlert("Уведомления",  "Дата  выполнения
должна быть больше текущей!", "Ок");
        }
        catch { Application.Current.MainPage.DisplayAlert("Уведомления", "Все поля должны
быть заполнены!", "Ок"); }
        }
        ///       <summary>
        ///       Отобразить список заданий для выполнения
        ///       </summary>
        void Open_List_Task_Alien_()
        {
        if (index_room > -1)
        {
        Select_Task_Room_Alien();
        App.Current.MainPage.Navigation.PushAsync(new  Open_List_Task_Alien(ID,  models,
        index_room));
        }       }
        ///       <summary>
        ///       Отобразить список заданий для выполнения
        ///       </summary>
        void Select_Task_Room_Alien()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Select_Task_Room_Alien", connection)) {
        Task_Room.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
```

```
command.Parameters.AddWithValue("IdRoom", index_room); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
        {
        var item = new Task_Room(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString(), rdr[5].ToString(), rdr[6].ToString(), rdr[7].ToString(),
rdr[8].ToString());
        Task_Room.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        }
        int task_;
        ///     <summary>
        ///     Открыть задание для участника
        ///     </summary>
        ///
        void OpenTask_Room_Alien_(object obj)
        {
        if (obj is Task_Room item)
        {
        App.Current.MainPage.Navigation.PushAsync(new Open_Task_Al-
ien(int.Parse(item.IdTaskRoom), item.Title_Task, item.Text_Room, item.Point_Target, item.Point,
item.Complate_Point_Task_Room, item.Data, item.Num_Completed, ID, models));
        task_ = int.Parse(item.IdTaskRoom);
        }
        }
        ///     <summary>
        ///     Выполнить задание
        /// </summary>
        void Complate_Task_()
        {
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command = new
MySqlCommand("Insert_Task_Room_Complate", connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdTasks", task_);
command.Parameters.AddWithValue("Date", DateTime.Now.Date);
command.Parameters.AddWithValue("IdRoom", index_room); int result =
command.ExecuteNonQuery(); connection.Close();
        }
        using (MySqlCommand command = new MySqlCommand("Update_Status_Task",
connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdTask", task_); int result = command.ExecuteNonQuery();
```

```
connection.Close();
        }
        }
        Select_Task_Room_Alien();
        Application.Current.MainPage.Navigation.PushAsync(new Open_List_Task_Alien(ID,
mod-els, index_room));
        }
        ObservableCollection<Tasks_Room_Complated> tasks_room_complated_; public
ObservableCollection<Tasks_Room_Complated> Tasks_Room_Complated {
        get
        {
        set
        return tasks_room_complated_;       }
        {       if (tasks_room_complated_ != value)
        tasks_room_complated_ = value;
        {
        OnPropertyChanged();
        }
        }
        }
        ObservableCollection<Tasks_Room_Complated> tasks_room_complated_not; public
ObservableCollection<Tasks_Room_Complated> Tasks_Room_Complated_not {
        get
        { return tasks_room_complated_not; } set { if (tasks_room_complated_not != value)
        {
        tasks_room_complated_not = value;
        OnPropertyChanged();
        }
        }
        }
        void Select_Task_Archive_Not()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Select_Task_Archive_Not", connection))
        {
        Tasks_Room_Complated_not.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdRoom", index_room);
command.Parameters.AddWithValue("IdTask", index_task_); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
        {
        var     item    =       new     Tasks_Room_Complated(rdr[0].ToString(),
        rdr[1].ToString());
        Tasks_Room_Complated_not.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        }
```

```
        void Select_Task_Archive()
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (var command = new MySqlCommand("Select_Task_Archive", connection))
        {
        Tasks_Room_Complated.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdTask", index_task_); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
        {
        var item = new Tasks_Room_Complated(rdr[0].ToString(), rdr[1].ToString(),
rdr[2].ToString());
        Tasks_Room_Complated.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        }
        ///     <summary>
        ///     Открыть архив по заданиям
        ///     </summary>
        void Open_MyTask_Archive_()
        {
        Select_Task_Archive();
        Select_Task_Archive_Not();
        Application.Current.MainPage.Navigation.PushAsync(new
Open_MyTask_Archive(models));
        }
        ///     <summary>
        ///     Открыть архив выполненых заданий
        ///     </summary>
        void Open_Archibve_Complated_()
        {
        Select_Archive_Complate();
        Application.Current.MainPage.Navigation.PushAsync(new
  Open_Archive_Complate(models));
        }
        ObservableCollection<Task_Room> task_room_; public
ObservableCollection<Task_Room> Task_Room_
        {
        get
        {
        return task_room_;
        }
        set     {
        if (task_room_ != value)
        {
        task_room_ = value;
        OnPropertyChanged();
```

```
}
}
}
///     <summary>
///     Архив выполненых задач
///     </summary>
void Select_Archive_Complate()
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Select_Archive_Complate", connection))
{
Task_Room_.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdRoom", index_room); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
{
var item = new Task_Room(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString(), rdr[5].ToString(), rdr[6].ToString(), rdr[7].ToString(),
rdr[8].ToString());
Task_Room_.Add(item);
}
rdr.Close();
connection.Close();
}
}
}
ObsevableCollection<Task_Room> task_room_not_; public
ObservableCollection<Task_Room> Task_Room_NOT_ {
get
{
return task_room_not_;
}
set
{
if (task_room_not_ != value)
{
task_room_not_ = value;
OnPropertyChanged();
}
}
}
///     <summary>
///     Архив не выполненых задач
///     </summary>
void Select_Archive_NOT_Complate()
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command =     new MySqlCommand("Select_Archive_NOT_Complate",
```

```
connection))
        {
        Task_Room_NOT_.Clear();
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdRoom", index_room); MySqlDataReader rdr =
command.ExecuteReader(); while (rdr.Read())
        {
        var item = new Task_Room(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString(), rdr[4].ToString(), rdr[5].ToString(), rdr[6].ToString(), rdr[7].ToString(),
rdr[8].ToString());
        Task_Room_NOT_.Add(item);
        }
        rdr.Close();
        connection.Close();
        }
        }
        }
        ///     <summary>
        ///     Архив выполненых задач
        ///     </summary>
        void Open_Archibve_NOT_Complated_()
        {
        Select_Archive_NOT_Complate();
        Application.Current.MainPage.Navigation.PushAsync(new Open_Ar-
chive_NOT_Complate(models));
        }
        ///     <summary>
        ///     Открыть рейтинг участников комнты
        ///     </summary>
        void Open_Rating_User_()
        {
        Select_Rating();
        Application.Current.MainPage.Navigation.PushAsync(new Rating_User(models));
        }
        ObservableCollection<Tasks_Room_Complated> rating_us;
        public ObservableCollection<Tasks_Room_Complated> Rating_User
        {
        get
        {     return rating_us;     }     set     {
        if (rating_us != value)
        {
        rating_us = value;
        OnPropertyChanged();
        }
        }
        }
        ///     <summary>
        ///     Отправить приглашение
        ///     </summary>
        void Send_invitation_()
```

```
        {
        if (LOGIN.Length >= 1)
        {
        int x = 0;
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        int id_out;
        using (MySqlCommand command = new
  MySqlCommand("Check_Login_Invitation",connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("Login", LOGIN); MySqlDataReader reader =
command.ExecuteReader(); while (reader.Read()) x++; reader.Close(); connection.Close();
        }
        if (x > 0)
        {
        using (MySqlCommand command_ = new MySqlCommand("Out_Id_User", connection))
        {
        connection.Open();
        command_.CommandType = CommandType.StoredProcedure;
command_.Parameters.AddWithValue("Login", LOGIN); MySqlParameter id = new
MySqlParameter
        {
        DbType = DbType.Int32,
        ParameterName = "IdUser",
        Direction = ParameterDirection.Output
        };
        command_.Parameters.Add(id);
        int result_procedure = command_.ExecuteNonQuery();
        string Id = command_.Parameters["IdUser"].Value.ToString();
        connection.Close();
        id_out = int.Parse(Id);
        }
        using (MySqlCommand command__ = new MySqlCommand("Insert_Invitation",
        connection))
        {
        connection.Open();
        command__.CommandType = CommandType.StoredProcedure;
command__.Parameters.AddWithValue("IdUser_in", ID);
command__.Parameters.AddWithValue("IdUser_out", id_out);
command__.Parameters.AddWithValue("IdRoom", index_room); int result_ =
command__.ExecuteNonQuery(); connection.Close();
        }
        }
        else { Application.Current.MainPage.DisplayAlert("Ошибка", "Пользователь с такими
данными не существует", "Ок"); }
        }
        }
        else Application.Current.MainPage.DisplayAlert("Уведомления", "Введите логин или
почту пользователя", "Ок");
        }
```

```
ObservableCollection<Invitation> invitation_; public ObservableCollection<Invitation>
Invitation_ {
get
{
return invitation_;
}
set
{
if (invitation_ != value)
{
invitation_ = value;
OnPropertyChanged();
}
}
}
///  <summary>
///  Вывод списка приглашений
///  </summary>
void Select_Invitation()
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Select_Invitation", connection))
{
Invitation_.Clear();
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
MySqlDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
var item = new Invitation(rdr[0].ToString(), rdr[1].ToString(), rdr[2].ToString(),
rdr[3].ToString());
Invitation_.Add(item);
}
rdr.Close();
connection.Close();}}}
public ICommand Input_Invitation => new Command(Input_Invitation_); public ICommand
Close_Invitation => new Command(Close_Invitation_); void Input_Invitation_(object obj) {
if (obj is Invitation item)
{
using (var connection = new MySqlConnection(ConnectionString))
{
using (var command = new MySqlCommand("Insert_Member_Magazine", connection))
{
connection.Open();
command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("IdUser", ID);
command.Parameters.AddWithValue("IdRoom", int.Parse(item.IdRoom)); MySqlDataReader rdr =
command.ExecuteReader(); connection.Close();
}
using (MySqlCommand command__ = new MySqlCommand("Update_Num_User_Room",
```

```
connection))
        {
        connection.Open();
        command__.CommandType = CommandType.StoredProcedure;
        command__.Parameters.AddWithValue("IdUser", ID);
        command__.Parameters.AddWithValue("IdRoom", int.Parse(item.IdRoom));
        int result_ = command__.ExecuteNonQuery();
        connection.Close();
        }
        using (MySqlCommand command__ = new MySqlCommand("Update_Status_Invitation",
        connection))
        {
        connection.Open();
        command__.CommandType = CommandType.StoredProcedure;
        command__.Parameters.AddWithValue("IdInvitation", item.IdInvitation); int result_ =
command__.ExecuteNonQuery(); connection.Close();
        }
        }
        Select_Invitation();
        }
        }
        void Close_Invitation_(object obj)
        {
        if (obj is Invitation item)
        {
        using (var connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command__ = new MySqlCommand("Update_Status_Invitation",
        connection))
        {
        connection.Open();
        command__.CommandType = CommandType.StoredProcedure;
        command__.Parameters.AddWithValue("IdInvitation", item.IdInvitation); int result_ =
command__.ExecuteNonQuery();
        connection.Close();
        }
        }
        Select_Invitation();}}}}
```

Клас Entrance.cs

```
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;
        using MySql.Data.MySqlClient;
        using App1.MainFile;
        using App1.Registration_entry.form;
        using Xamarin.Forms;
        using Xamarin.Forms.Xaml;
        using System.Data;
        namespace App1
```

```
        {
        [XamlCompilation(XamlCompilationOptions.Compile)] public partial class Entrance :
ContentPage {
        string ConnectionString;
        public Entrance()
        {
        InitializeComponent();
        }
        async void Login_password(object sender, System.EventArgs e)
        {
        if (Ent1.Text != "Логин или электроннная почта" && Ent4.Text != "Пароль
        ")
        {
        int x = 0;
        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
        using (MySqlCommand command = new MySqlCommand("Check_User", connection))
        {
        connection.Open();
        command.CommandType = CommandType.StoredProcedure;
command.Parameters.AddWithValue("Login", Ent1.Text);
command.Parameters.AddWithValue("Password", Ent4.Text); MySqlDataReader reader =
command.ExecuteReader(); while (reader.Read()) x++; reader.Close(); connection.Close();
        if (x != 0)
        {
        using (MySqlCommand command1 = new MySqlCommand("Out_ID", connection))
        {
        connection.Open();
        command1.CommandType = CommandType.StoredProcedure;
command1.Parameters.AddWithValue("Login", Ent1.Text);
command1.Parameters.AddWithValue("Password", Ent4.Text);
        MySqlParameter id = new MySqlParameter
        {
        DbType = DbType.Int32,
        ParameterName = "IdUser",
        Direction = ParameterDirection.Output
        };
        command1.Parameters.Add(id);
        int result_procedure = command1.ExecuteNonQuery();
        string Id = command1.Parameters["IdUser"].Value.ToString(); connection.Close();
        Navigation.InsertPageBefore(new Home_page(int.Parse(Id)),this);
        await Navigation.PopAsync();}
        }
        else await DisplayAlert("Ошибка", "Неверный логин или пароль","OK");}}}}
        void Visible_password(object sender, System.EventArgs e)
        {
        if (Ent4.IsPassword == true)
        {
        Ent4.IsPassword = false;
        }
        else Ent4.IsPassword = true;
        }
```

```
void Remember(object sender, System.EventArgs e)
{
}
private async void Registration (object sender, System.EventArgs e)
{
await Navigation.PushModalAsync(new Registration());
}
}
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи