РЕФЕРАТ

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

Об'єкт розробки: програмний додаток для інформаційної системи «WaterWay» на базі операційної системи Android.

Мета кваліфікаційної роботи: створення програмного забезпечення для автоматизації процесів бізнесу постачання води, яке дозволить здійснювати зручне замовлення води в електронній формі на смартфоні, а також обробляти отримане замовлення та спрощувати доставку товару замовнику.

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 додатка, що надає можливість електронного зберігання даних про користувачів та замовлення, ведення бази даних системи та доступу до даних користувачів та адміністраторів для ефективного надання сервісу замовлення та доставки товару.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, що оптимізують дії щодо виконання замовлень та доставки товарів, та скорочують час на їх доставку та на оформлення замовлення, підвищують ефективність діяльності компанії та спрощують життя людей.

Список ключових слів: МОБІЛЬНИЙ ДОДАТОК, ТОВАР, ІНТЕРФЕЙС, ПРОГРАМУВАННЯ, ОПЕРАЦІЙНА СИСТЕМА, КОРИСТУВАЧ, АДМІНІСТРАТОР, ПРОЕКТУВАННЯ.

ABSTRACT

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

Object of development: software application for the information system "WaterWay" based on the Android operating system.

The purpose of the qualification work: to create software for automation of water supply business processes, which will allow you to easily order water in electronic form on a smartphone, as well as process the received order and simplify delivery of goods to the customer.

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 an application that provides the ability to electronically store user and order data, maintain a system database and access user and administrator data to effectively provide the service of ordering and delivery of goods.

The relevance of this software product is determined by the high demand for such developments that optimize the execution of orders and delivery of goods, and reduce the time for their delivery and ordering, increase the efficiency of the company and simplify people's lives.

Keywords: MOBILE APP, PRODUCT, INTERFACE, PROGRAMMING, OPERATING SYSTEM, USER, ADMINISTRATOR, DESIGN.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – бази даних;

ЖЦПЗ – життєвий цикл програмного забезпечення;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СКБД – система керування базами даних.

ЗМІСТ

ВСТУП

Одним з перспективних видів бізнесу експерти називають ринок бутильованої води. Сім'ї з дітьми та працівники офісів все частіше замовляють доставку води за визначеною адресою. А все тому, що ця послуга досить недорога з точки зору щомісячних витрат. Крім того, багато клієнтів вирішують таким чином проблему очищення водопровідної води, якість якої залишає бажати кращого.

Споживання очищеної води щорічно збільшується на 7–10%. Це означає, що ринок знаходиться на постійному піднесенні, і в цьому полягає одна з головних переваг так званого «водяного» бізнесу. І якщо компаніям, які здійснюють очищення води, періодично необхідно вкладати кошти в оновлення обладнання, то постачальникам очищеної води це не загрожує – вся їх вигода полягає в експедиції води від виробника до споживача.

Підприємницька діяльність з постачання води працює, як з оптовими так і з роздрібними замовниками. В основному клієнти постійні. Отже, для здійснення замовлення поставки води замовник повинен кожен раз робити телефонний дзвінок. Даний процес потребує спрощення і модернізації.

Основними елементами доставки води є автомобіль і тара – бутлі. Водій, який доставляє клієнтам одиницю товару, зустрічається з проблемою пошуку маршруту при великій кількості замовлень в різні частини міста.

Вирішенням вище зазначених проблем може стати програмний продукт, який дозволить спростити процеси взаємодії клієнта з постачальником.

Метою кваліфікаційної роботи є створення програмного забезпечення для автоматизації процесів бізнесу постачання води, яке дозволить здійснювати зручне замовлення води в електронній формі на смартфоні, а також обробляти отримане замовлення та спрощувати доставку товару замовнику.

В ході роботи над роботою необхідно спроектувати і розробити інформаційну систему «WaterWay» для операційної системи Android, для автоматизації процесів бізнесу доставки води. Створений мобільний додаток

має зручний багатовіконний графічний інтерфейс та декілька рівнів доступу: «Клієнт», «Водій» та «Адміністратор».

В ході створення програмного додатку реалізовані наступні функції в залежності від рівня доступу користувача:

1. Користувач з рівнем доступу «Клієнт» має можливість створювати та повторювати замовлення з історії, редагувати особисті дані та користуватися технічною підтримкою.

2. Користувач з рівнем доступу «Водій» має можливість переглядати перелік замовлень, інформацію про клієнта та маршрут слідування до адреси замовника.

3. Для користувача з рівнем доступу «Адміністратор» додаток надає можливість переглядати інформацію про клієнтів та надавати їм технічну підтримку. Також можливо переглядати, редагувати та додавати дані про нових робітників до бази співробітників, додавати дані про нові товари, редагувати існуючі та змінювати інформацію про фірму.

Практичне значення полягає у створенні додатка, що надає можливість електронного зберігання даних про користувачів та замовлення, ведення бази даних системи та доступу до даних користувачів та адміністраторів для ефективного надання сервісу замовлення та доставки товару.

Унікальністю даного програмного продукту є реалізація технічної підтримки користувачів в реальному часі, за допомогою якої користувачі можуть швидко отримувати відповіді на непередбачувані проблеми та запитання.

Дана система може знайти широке застосування на підприємствах, які здійснюють доставку води та серед їх клієнтів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

В принципі роботи сервісу доставки є можливість замовлення товару на сайті та його доставці в потрібне місце призначення. Замовлення товару відбувається за принципом інтернет-магазину.

Для того, щоб проект торгівлі через Інтернет став успішний, крім навичок ведення бізнесу, наявності стартового капіталу, домовленостей з постачальниками, а також системи обробки замовлень, необхідно:

— написання продуктивної системи керування контентом, розрахованого на модульність і модернізацію;

— привабливий і зручний інтерфейс як для користувацької частини, так і частини адміністрування;

— оптимізація сайту у пошукових системах для збільшення кількості клієнтів, рекламні кампанії.

Типовий варіант інтернет-магазину складається із наступних функціональних частин: каталог товарів, пошукова система, віртуальна корзина користувача, реєстраційна форма, форма відправлення замовлення (рис. 1.1).

Рис. 1.1. Типовий алгоритм замовлення товару в інтернет-магазині

Каталог являє собою складну і багаторівневу структуру даних, яка повинна простим і зрозумілим способом виконувати упорядкування товарів. Простіше за все такий каталог представити у вигляді дерева об'єктів, верхній рівень якого складається зі списку розділів. Розділи можуть містити підрозділи або посилання на конкретний товар і т.д. Таке впорядкування просто необхідно для зручного і швидкого пошуку і замовлення товарів.

Пошукова система є обов'язковим елементом динамічного каталогу й реалізується на стороні сервера. Незважаючи на те, що каталог забезпечує упорядкування та угруповання даних, пошукова система дає користувачеві можливість швидкого пошуку інформації, що особливо важливо в тому випадку, коли каталог являє собою досить розгалужену структуру даних з великою кількістю розділів, підрозділів і товарів, користувач погано уявляє в якому розділі може перебувати цікавить його товар і чи є він в каталозі взагалі.

Пошукова система в деяких випадках дозволяє значно скоротити кількість переходів між сторінками каталогу для доступу до інформації, що цікавить.

Особливість реалізації пошуку в Інтернеті полягає в тому, що тут відбувається вибірка всіх записів, які задовольняють умовам запиту (даний механізм пошуку я називаю пошук з надлишком). У разі великої вибірки даних вивід результатів пошуку здійснюється посторінково для того, щоб відвідувачам не доводилося довго чекати завантаження всієї вибірки, яка може включати в себе сотні, тисячі і більше записів. Як правило, відвідувачі не переглядають всі сторінки вибірки, обмежуючись двома або трьома. Тому даний механізм пошуку в багатьох випадках працює вкрай повільно і неефективно. Однак він дозволяє здійснити вибірку однакових товарів від різних постачальників, порівняти їх параметри між собою і вибрати оптимальний варіант.

Користувацький кошик представляє собою деякий масив даних, який служить для зберігання замовленого користувачем товару.

Реєстраційна форма для введення персональних даних користувачів. Надалі ця інформація використовується для їх ідентифікації між сеансами роботи з інтернет-магазином. Дана інформація може зберігатися як на стороні сервера, так і на стороні клієнта.

Форма відправки замовлення служить для введення контактної інформації замовника і відправки її та замовлення на електронну скриньку організації.

Як додаткові елементи можна зустріти різні рейтинги з автоматичною вибіркою товарів для демонстрації в розділі "Спеціальна пропозиція" або "Десятка кращих", системи спостереження, які можуть враховувати переваги відвідувачів та надавати їм додатковий сервіс, особисті настройки, облік замовлень, різні інтерфейси, системи обробки кредитних карток, платежів і т.п.

Інтернет-магазин можна реалізувати як на стороні сервера, так і на стороні клієнта. У першому випадку використовуються серверні сценарії, побудовані на таких технологіях як PHP, Perl, ASP, JSP, ColdFusion і т.п. У

другому випадку - JavaScript (ActiveX, Java тощо, мають обмежене застосування). Вибір тієї чи іншої реалізації залежить від багатьох факторів, які випливають з визначених на етапі планування цілей та постановки завдання.

Технічна сторона питання залежить від передбачуваного обсягу бази даних і навантаження на неї, вимогам за швидкістю і функціональності програми, можливостей з розробки, адміністрування та інших потреб.

Інтернет магазин - довгостроковий проект, який повинен ґрунтуватися на тактичному й стратегічному плануванні діяльності організації-замовника, тобто необхідна постійна підтримка магазину та його подальший розвиток, що може зажадати значних фінансових витрат, оскільки для цієї роботи необхідно залучати декількох фахівців.

1.2. Призначення розробки та область застосування

В ході роботи над роботою необхідно розробити інформаційну систему для операційної системи Android, для автоматизації процесів бізнесу доставки води. Даний додаток надає можливість електронного зберігання даних про користувачів та замовлення, ведення бази даних системи та доступу до даних користувачів та адміністраторів для ефективного надання сервісу замовлення та доставки товару.

Унікальністю даного програмного продукту є реалізація технічної підтримки користувачів в реальному часі, за допомогою якої користувачі можуть швидко отримувати відповіді на непередбачувані проблеми та запитання.

Дана система може знайти широке застосування на підприємствах, які здійснюють доставку води та серед їх клієнтів.

Додаток призначений для користувачів, що ведуть активний образ життя та не мають часу, або бажання витрачати його на ходіння по магазинам та надають перевагу доставці необхідних товарів (бутильованої води) кур’єром в призначені місто та час.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного додатку для забезпечення мобільного сервісу доставки води на базі ОС Android» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою кваліфікаційної роботи є створення програмного забезпечення для автоматизації процесів бізнесу постачання води, яке дозволить здійснювати зручне замовлення води в електронній формі на смартфоні, а також обробляти отримане замовлення та спрощувати доставку товару замовнику.

Завданням даної роботи є розробка інформаційної системи "WaterWay" для автоматизації процесів доставки води від постачальника до клієнта завдяки мобільному девайсу на операційній системі Android.

В ході роботи необхідно виконати наступні етапи:

1. Спроектувати та розробити структуру інформаційної системи для автоматизації процесів бізнесу доставки води.
2. Розробити зручний графічний інтерфейс.
3. Виконати програмування системи, її тестування та налагодження.

Для виконання даної роботи необхідно розв’язати наступні завдання:

- дослідити середовище розробки;
- виділити основні складові створюваного додатку;
- дослідити найоптимальніші алгоритми для роботи з базами даних під ОС Android та скласти базу даних системи;
- дослідити створення інтерфейсу в програмах для ОС Android;
- визначити технології, методи та засоби взаємодії пристрою з інтернет-ресурсами.

В ході створення програмного додатку мають бути реалізовані наступні функції в залежності від рівня доступу користувача:

1. Користувач з рівнем доступу «Клієнт» має можливість створювати та повторювати замовлення з історії, редагувати особисті дані та користуватися технічною підтримкою.

2. Користувач з рівнем доступу «Водій» має можливість переглядати перелік замовлень, інформацію про клієнта та маршрут слідування до адреси замовника.

3. Для користувача з рівнем доступу «Адміністратор» додаток надає можливість переглядати інформацію про клієнтів та надавати їм технічну підтримку. Також можливо переглядати, редагувати та додавати дані про нових робітників до бази співробітників, додавати дані про нові товари, редагувати існуючі та змінювати інформацію про фірму.

В даному програмному продукті повинна бути реалізована технічна підтримка користувачів в реальному часі, за допомогою якої користувачі можуть швидко отримувати відповіді на непередбачувані проблеми та запитання.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програмний продукт «Water Way» повинен складатися з двох окремих додатків: для клієнта та для співробітників фірми.

Додаток для клієнта повинен виконувати наступні функції:

– реєстрація нових користувачів у системі;

– оновлення та редагування персональних даних клієнта;

– реєстрація клієнтом нових замовлень з вибором дати і кількості продукції;

– створення клієнтом та повторення збереженого замовлення;

– доступ до технічної підтримки;

– перегляд історії попередніх замовлень.

Додаток для співробітників фірми повинен виконувати наступні функції:

– перегляд списку даних про зареєстрованих клієнтів та співробітників;

– перегляд та додавання нових товарів та акцій;

– перегляд списку замовлень клієнтів;

– побудова оптимального маршруту для водія.

Інтерфейс додатку для клієнта повинен бути графічним та багатовіконним і мати наступні вікна:

– товари: містить відомості про наявні товари;

– замовлення: дає смогу додати в кошик обраний товар вказаної кількості;

– історія: містить історію замовлень користувача;

– кошик: дає змогу користувачеві оформити створене замовлення;

– підтримка: дає змогу отримувати технічну підтримку;

– налаштування: містить функціонал для редагування персональ них даних та налаштування додатку.

Інтерфейс додатку для постачальника повинен мати головне вікно і підпорядковані, які містять функціонал для перегляду бази клієнтів та замовлень, дозволяють додавати та редагувати інформацію про товари та створювати оптимальний маршрут для водія.

1.5.2. Вимоги до інформаційної безпеки

Для надійної роботи системи необхідно:

1. Використовувати ліцензійне програмне забезпечення на сервері.
2. Здійснювати захист від вірусів на сервері.
3. Здійснювати захист від несанкціонованого доступу.
4. Застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні.

5. Здійснювати контроль даних, що вводяться користувачами.

6. Автоматично завершувати сеанс роботи з користувачем у разі тривалої перерви його активності.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для використання розробленого додатку необхідно мати смартфони з наступними характеристиками:

– стандарти зв'язку: GSM: 850/ 900/1800/1900 МГц, HSPA/WCDMA: 900/2100 МГц;

– роздільна здатність дисплея: 1280x720;

– кількість кольорів: 16 мільйонів;

– тип дисплею: IPS;

– процесор: Qualcomm Snapdragon 400 MSM8228;

– кількість ядер: 2;

– частота процесора: 1.6 ГГц;

– внутрішня пам'ять: 8 Гб;

– оперативна пам'ять: 2 Гб;

– підтримка карт пам'яті: microSD;

– операційна система: Android 4.2 Jelly Bean;

– бездротові технології Wi–fi: IEEE 802.11 b/g/n[8].

Рекомендовані характеристики серверу:

– операційна система Windows Server 2016;

– вільне місце на жорсткому диску 35 Гб;

– процесор Intel Xeon E7–8867L ;

– процесор із тактовою частотою 3.2 GHz;

– оперативна пам'ять 12 GB;

– система керування базами даних MS SQL Server [3].

1.5.4. Вимоги до інформаційної та програмної сумісності

Інформаційна системи має бути розроблена на кросплатформовій мові програмування, завдяки чому її можна буде використовувати під різними платформами (Android , Windows, IOS тощо) та мати засоби для доступу до мережі Інтернет.

Програма повинна являти собою самостійний виконуваний модуль, бути структурована і за коментована.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Інформаційна система складається з двох окремих додатків: для клієнта та для співробітників фірми.

Додаток для клієнта виконує наступні функції:

– реєстрація нових користувачів у системі;

– оновлення та редагування персональних даних клієнта;

– реєстрація клієнтом нових замовлень з вибором дати і кількості продукції;

– створення клієнтом та повторення збереженого замовлення;

– доступ до технічної підтримки;

– перегляд історії попередніх замовлень.

Дані функції забезпечують наступні форми:

– товари: містить відомості про наявні товари;

– замовлення: дає смогу додати в кошик обраний товар вказаної кількості;

– історія: містить історію замовлень користувача;

– кошик: дає змогу користувачеві оформити створене замовлення;

– підтримка: дає змогу отримувати технічну підтримку;

– налаштування: містить функціонал для редагування персональ них даних та налаштування додатку.

Додаток для співробітників фірми виконує наступні функції:

– перегляд списку даних про зареєстрованих клієнтів та співробітників;

– перегляд та додавання нових товарів та акцій;

– перегляд списку замовлень клієнтів;

– побудова оптимального маршруту для водія.

Даний функціонал забезпечується формами перегляду бази клієнтів та

замовлень, додавання та редагування інформації про товари та створення оптимального маршруту для водія.

В даному програмному продукті реалізована технічна підтримка користувачів в реальному часі, за допомогою якої користувачі можуть швидко отримувати відповіді на непередбачувані проблеми та запитання.

2.2. Опис застосованих математичних методів

В даній інформаційній системі ні під час розробки, ні під час тестування та роботи додатка, не використовуються та не розглядаються ніякі математичні методи.

2.3. Опис використаної архітектури та шаблонів проектування

2.3.1. Опис архітектури системи

В даній програмі використана архітектура «клієнт-сервер».

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

— набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

— набір клієнтів, які використовують сервіси, що надаються серверами;

— мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть

не мати жодного уявлення про існування інших клієнтів (рис. 2.1.).

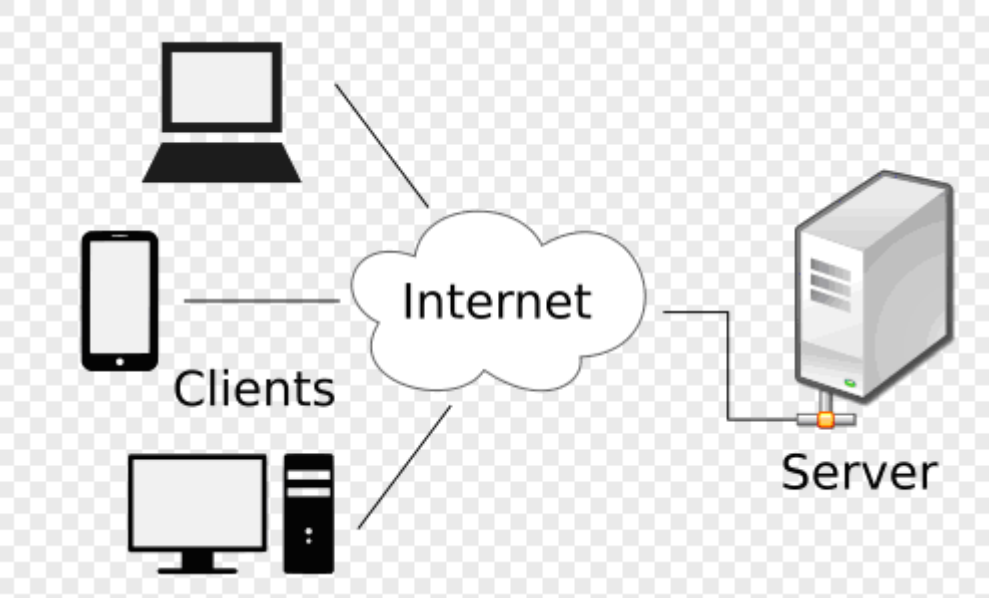

Рис. 2.1. Архітектура «клієнт-сервер»

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери - це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми - і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

— рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

— прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;

— рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів − клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

— модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

— модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку. Програми проміжного рівня можуть функціювати під управлінням спеціальних серверів застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення - звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає). Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює

обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

2.3.2. Опис технології ЖЦПЗ

Життєвий цикл програмного забезпечення - це безперервний процес, який починається з моменту прийняття рішення про необхідність створення ПЗ і закінчується в момент його повного вилучення з експлуатації.

Існує декілька підходів при визначенні фаз та робіт життєвого циклу програмного забезпечення (ЖЦПЗ), кроків процесу програмування, каскадна і спіральна моделі. Але всі вони містять загальні основні компоненти: постановка завдання, проектування рішення, реалізація, обслуговування.

В даній кваліфікаційній роботі використаний процес програмування по Райлі.

Процес програмування включає чотири кроки:

— постановка задачі, тобто отримання адекватного уявлення про те, яке завдання має виконати програма;

— проектування рішення вже поставленого завдання (загалом, таке рішення є менш формальним, ніж остаточна програма);

— кодування програми, тобто переклад спроектованого рішення в програму, яка може бути виконана на машині;

— супровід програми, тобто безперервний процес усунення в програмі

неполадок і додавання нових можливостей.

Програмування починається з того моменту, коли користувач, тобто той, хто потребує програми для вирішення завдання, викладає проблему системного аналітику. Користувач і системний аналітик спільно визначають постановку задачі. Остання потім передається алгоритмісту, який відповідає за проектування рішення. Рішення (або алгоритм) представляє послідовність операцій, виконання яких призводить до вирішення завдання. Оскільки алгоритм часто не пристосований до виконання на машині, його слід перевести в машинну програму. Ця операція виконується кодувальником. За наступні зміни в програмі несе відповідальність супроводжуючий програміст. І системний аналітик, і алгорітміст, та кодер, і супроводжуючий програміст - всі вони є програмістами.

Одним з найбільш важливих кроків програмування є постановка задачі. Вона виконує функції контракту між користувачем і програмістом (програмістами). Як і юридично погано складений контракт, погана постановка задачі марна. При гарній постановці завдання як користувач, так і програміст ясно і недвозначно представляють завдання, яку необхідно виконати, тобто в цьому випадку враховуються інтереси як користувача, так і програміста. Користувач може планувати використання ще нествореного програмного забезпечення, спираючись на знання того, що воно може. Гарна постановка завдання служить основою для формування її рішення.

Постановка завдання (специфікація програми), по суті, означає точне, повне і зрозуміле опис того, що відбувається при виконанні конкретної програми. Користувач зазвичай дивиться на комп'ютер, як на чорний ящик: для нього неважливо, як працює комп'ютер, а важливо, що може комп'ютер з того, що цікавить користувача. При цьому основна увага фокусується на взаємодії людини з машиною.

Д. Райлі пропонує для постановки задачі користуватися стандартною формою, яка забезпечує максимальну точність, повноту, ясність і включає:

— найменування завдання (схематичне визначення);

— загальний опис (короткий виклад завдання);

— введення;

— висновок;

— помилки (явно перераховані незвичайні варіанти введення, щоб показати користувачам і програмістам ті дії, які зробить машина в подібних ситуаціях);

— приклад (хороший приклад може передати сутність завдання, а також проілюструвати різні випадки).

Проектування рішення програмування є найбільш важким. На даній стадії постановка задачі має бути перетворена на алгоритм. Тому алгорітміст повинен володіти достатнім досвідом програмування і підходити до кожної нової задачі, спираючись на твердо встановлену методику проектування. Щоб уникнути помилок в програмах, алгорітмісти повинні використовувати ретельно розроблені процедури конструювання, засновані на правилах логічного висновку.

Завдання проектувальника - створення алгоритму, що виконує функції сполучної ланки між постановкою завдання і готова для виконання програмою. Перевірку створеного алгоритму, тобто наскільки останній відображає постановку завдання, здійснює системний аналітик. У силу цього і системний аналітик, і проектувальник повинні вміти читати і розуміти алгоритм. Кожен алгоритм записується на деякій псевдомова. Алгоритми, звані також псевдокод, не можуть бути виконані ні на якому комп'ютері.

Кодування алгоритму полягає в перекладі алгоритму в програму. Для створення повної, точної та зрозумілої програми необхідні відповідні методи запису програм. На відміну від природних, мови програмування створені спеціально для такого подання рішення завдання, яке може бути виконано комп'ютером.

Перш ніж завершити роботу, кодіровщік повинен переконатися, що програма відповідає псевдокод. Потім системний аналітик, алгорітміст і, що найголовніше, користувач повинні протестувати і підтвердити, що вона працює

правильно. Після цього можна вважати, що програма готова для передачі користувачеві в комплекті з усією необхідною документацією.

Однак на цьому програмування не закінчується, далі слідує крок супроводу. Справа в тому, що в програмі можуть бути помилки, зумовлені або неадекватною постановкою завдання, або тим, що проект не задовольняє постановці задачі або програма не відповідає проекту. Яка б не була причина, користувач має право вимагати коригування програми, оскільки він не уявляв, що програма буде працювати таким чином. Виправлення помилок є однією з головних завдань супроводу програм. Інший не менш важливим завданням супроводу програм є її модифікація, тобто додавання в програму нових можливостей або зміна існуючих. Користувач може змінити вимоги до роботи програми, що, у свою чергу, призведе до необхідності її переписати. Складність операцій із супроводу програми залежить від типу змін, які повинні бути зроблені: у гіршому випадку може знадобитися повна переробка програми від постановки до кодування. Зазвичай на супровід програми витрачається більше часу, ніж на її створення.

Останньою складовою процесу програмування є документування. Воно включає широкий спектр описів, що полегшують процес програмування і збагачують результуючу програму. Постійне документування має становити невід'ємну частину кожного кроку програмування. Постановка завдання, проектні документи, алгоритми і програми - все це документи. Внутрішня документація, зазначена безпосередньо в програму, полегшує читання коду.

Відповідно до моделі, представленої в цьому проекті, програмування можна розділити на чотири кроки: постановку задачі, проектування рішень, кодування програми, супровід програми. Додатково модель включає документування програми як дії, які необхідно виконувати протягом всього процесу програмування.

2.4. Опис використаних технологій та мов програмування

Програмний додаток реалізований за допомогою методів об'єктно-орієнтованої мови програмування C# та фреймворка Xamarin. Він може повноцінно працювати у середовищі операційних систем Android та IOS.

ОС iOS (відома як iPhone OS до червня 2010 року) – це власницька мобільна операційна система від Apple. Розроблена спочатку для iPhone, вона стала операційною системою також для iPod Touch, iPad і Apple TV. Apple не дозволяє роботу ОС на мобільних телефонах інших фірм. iOS є похідною від OS X, отже, є за своєю природою Unix – подібною операційною системою. Користувацький інтерфейс iOS заснований на концепції прямої маніпуляції з використанням жестів Multi – Touch. Елементи інтерфейсу управління складаються з повзунків, перемикачів і кнопок. Внутрішній акселерометр використовуються деякими програмами для реагування на струшування пристрою, яке є також загальною командою скасування, або обертати пристрій у трьох вимірах, що є загальною командою перемикання між книжковим та альбомним режимами [4].

Android – операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Під дану ОС розробляються додатки, які можна завантажити через Google Play або зі сторонніх ресурсів. На її основі пишуться інші операційні системи під кастомні і краудфандінгові девайси завдяки наявності відкритого коду для розробників.

Розглянемо головні особливості даної ОС:

– велика кількість безкоштовних, і тому цілком доступних додатків, в тому числі і антивіруса. Для порівняння, в тій же операційній системі, з яблуком на логотипі, практично всі програми доведеться купувати;

– android дозволяє використовувати на повну потужність високочастотні процесори. Крім того, як правило, пристрої, що працюють на

базі цієї системи, мають високу якість знімків;

– за допомогою Android дуже просто працювати з соціальними мережами, оскільки вона здатна підтримувати багато компонентів, які є дуже важливими для багатьох сайтів;

– крім того, з цією операційною системою досить просто працювати. Вона не вимагає настройки всіх служб окремо, для того, щоб все працювало так, як треба, можна всього лише створити один обліковий запис;

– інтерфейс системи доступний і зрозумілий. Правда, є люди, яким досить важко розібратися, але так трапляється досить рідко, до того ж, до кожного пристрою додається інструкція. А найголовнішою перевагою Android, є той факт, що на базі цієї системи здатні показувати хороший результат роботи, навіть пристрої, які можна назвати бюджетною моделлю, з дуже слабкими характеристиками процесора і оперативної пам'яті;

– система практично постійно перебуває в стадії оновлення. При цьому зростає її ступінь захисту, наприклад, Android 4.0, має чотири варіанти, як можна захистити свій телефон від зловмисників. Змінюється і її функціональна частина, додаючи нові опції, і удосконалюючи старі. Завдяки цьому, ОС Android завжди актуальна, і здатна задовольнити навіть найвищі вимоги [3].

Для розробки програми «Water Way» була використана мова програмування C# у середовищі візуального програмування Microsoft Visual Studio 2019.

C# – об’єктно–орієнтована мова програмування з безпечною системою типізації для платформи .NET. Дану мову використовують у багатьох напрямках, а саме: створення операційних систем, драйверів, різнопланового програмного забезпечення, а також додатків для вбудованих систем.

Мова програмування C# є об'єктно–орієнтованою мовою, відповідно вона підтримує основні концепції об’єктно–орієнтованого програмування, а саме: інкапсуляцію, успадкування і поліморфізм.

Бібліотека C# включає в себе набір засобів, які доступні для будь– якої реалізації мови, для того щоб забезпечити програмістам зручне користування

мовними засобами.

Доступ до всіх можливостей стандартної бібліотеки C# забезпечується за допомогою включення в програму (за допомогою директиви using) відповідних стандартних заголовків файлів. Засоби стандартної бібліотеки оголошуються, як вхідні в простір імен namespace.

Дана мова програмування досить тривалий час займає тверду позицію в десятці найпопулярніших та найкращих мов програмування.

Для використання абстрактних типів даних використовують класи. Клас – це деякий шаблон, на основі якого будуть створюватися його екземпляри – об’єкти. У C# класи оголошуються за допомогою ключового слова class.

Процес побудови програм в C# простіший та легший в порівнянні з мовами C або C++, але більш гнучкий, ніж в Java. Окремі файли заголовків не використовуються, тобто немає необхідності оголошувати методи і типи в певному порядку. Вихідний файл C# може визначити будь яке число класів, структур, інтерфейсів і подій [1].

Обрана мова програмування має величезний набір випадків використання:

– переважно використовується для створення корпоративного програмного забезпечення, фінансових проектів, наприклад для банків і бірж, зокрема мобільних додатків, хмарних сервісів;

– у порівнянні з Java легше взаємодіє, з кодом програм, написаних на інших мовах. І саме на C# часто пишуться розширення для інших мов програмування, які використовуються у якості прошарку між бібліотекою C#;

– широко використовується в розробці ігор на Unity – найпопулярніший ігровий двигун. Це означає, що сотні тисячі ігор, включаючи найпопулярніші створювалися за допомогою C#;

– взаємодіє з «Internet of Things» (IoT) – це концепція всеохоплюючого інтернету, підключення до інтернету холодильників, кондиціонерів, автомобілів і навіть кросівок з метою забезпечення своєму власникові більший комфот, а з іншого боку збільшення прибутку їх рітейлерам, розрахунку

кількості чого, скільки, коли потрібно мати в наявності на складах;

– наука та її прикладне застосування, наприклад проведення складних експериментальних розрахунків, криптографія, розпізнавання образів і тому подібне [2].

Що стосується порівняння мов програмування, слід відзначити, мова програмування C# високорівнева, це означає, що вона дещо схожа на англійську. Мова програмування C# має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk – у C#, опираючись на практику їхнього використання, навмисне виключили деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем у вище перелічених мовах програмування.

Для розробки додатку мовою програмування C# було використано середовище розробки Microsoft Visual Studio 2019 та фреймворк Xamarin.

Microsoft Visual Studio (MVS) – інтегроване середовище розробки програмного забезпечення від фірми Microsoft. Дане середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад віконні додатки Windows Forms, а також Web–додатки тощо.

Середовище Visual Studio дозволяє розробляти додатки, використовуючи різні мови програмування: Visual C#, Visual Basic, Visual F#, Visual C++, Python і т.д. Також існує можливість розробляти додатки не тільки під Windows, а і під інші популярні платформи: Android, iOS.

Visual Studio – це гарне середовище розробки додатків під ОС Windows. Microsoft Visual Studio надає користувачеві, при створенні проекту багато різних типів проектів, але велика кількість дає гарне уявлення про можливості даного інструменту. MVS – це висококласна Integrated Development Environment (IDE), якою користуються більшість професіоналів для розробки додатків під ОС Windows.

Для програмування під Windows, немає нічого кращого, ніж Visual Studio. Її редактор підсвічує синтаксис і виконує форматування коду, що в свою чергу збільшує читабельність коду. Більш того, редактор MVS автоматично завершує деякі структури коду.

Отже, MVS – це потужний інструмент розробки програмного забезпечення під ОС Windows. Він ідеально підходить для розробки великих проектів [1].

Framework Xamarin – це фреймворк для кросплатформеної розробки мобільних додатків (для платформ: iOS, Android, Windows Phone) з використанням мови C#. Призначений, щоб спростити програмісту процес написання коду на своїй улюбленій мові, із застосуванням всіх звичних мовних виразів, таких як Language- Integrated Query (LINQ), лямбда-виразів, узагальнених типів і асинхронних методів. При цьому маючи повний доступ до всіх можливостей SDK платформи і рідного механізму створення UI, отримуючи на виході додаток, який, строго кажучи, нічим не відрізняється від нативних і (принаймні по завіреннях) не поступається їм у продуктивності [3].

Фреймворк складається з декількох основних частин:

– Xamarin.IOS – бібліотека класів для C#, що надає розробнику доступ до iOS SDK;

– Xamarin.Android – бібліотека класів для C#, що надає розробнику доступ до Android SDK;

– компілятори для iOS і Android;

– IDE Xamarin Studio;

– плагін для Visual Studio.

Framework заснований на open – реалізації платформи .NET – Mono. Ця реалізація включає в себе власний компілятор C#, середу виконання, а так само основні .NET бібліотеки. Xamarin створювався, щоб дозволити запускати програми, написані на C#, на операційних системах, відмінних від Windows – Unix–системах, Mac OS і інших.

З точки зору виконання додатків між iOS і Android є одна ключова

відмінність – спосіб їх попередньої компіляції. Для виконання додатків в Android використовується віртуальна Java–машина Dalvik. Нативні додатки, які пишуться на Java, компілюються в проміжний байт-код, який інтерпретується з допомогою Dalvik в команди процесора в момент виконання. Це так звана Just-in-time компіляція (компіляція на льоту). В iOS використовується інша модель компіляції- Ahead-of-Time (компіляція перед виконанням). Xamarin враховує цю різницю, надаючи окремі компілятори для кожної з цих платформ. Способи компіляції додатків для Android та IOS зображено на рис. 2.2.

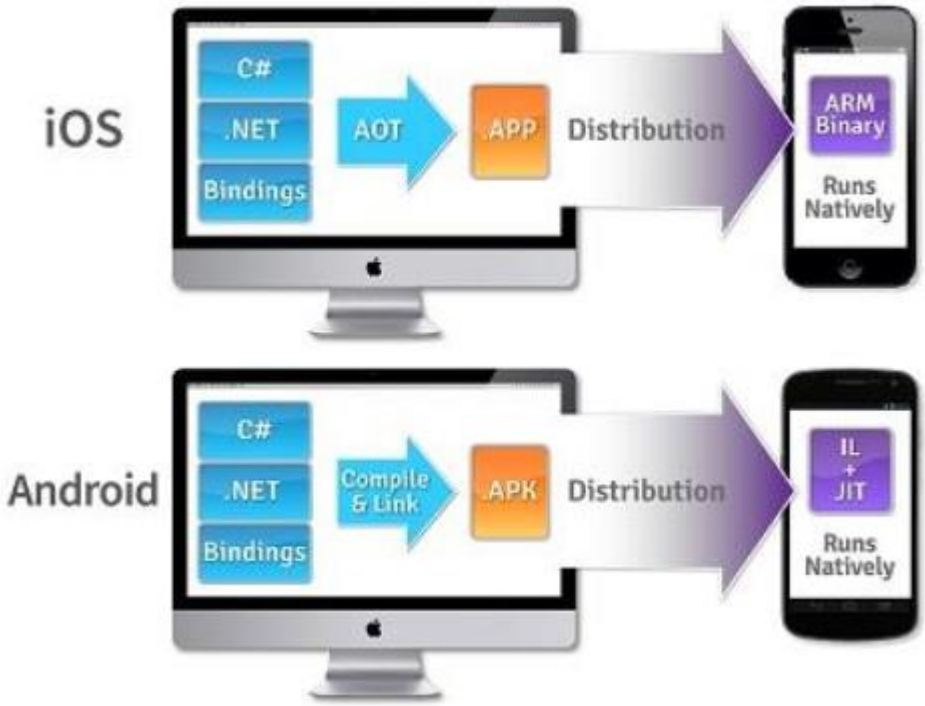

Рис. 2.2. Схема процесу компіляція додатків для Android та IOS

Для асинхронної розробки Xamarin надає можливість використовувати як класи з простору імен System.Threading.Thread і System.Threading.ThreadPool, так і повний спектр можливостей, що надаються Task Parallel Library. Використання останньої, однак, вважається кращим. Крім того, вийшла чергова Stable версія, в якій з'явилася підтримка .NET 4.7, зокрема, тепер можна використовувати ключові слова async/ await. Хоча ця можливість була доступна

і раніше, але для цього доводилося використовувати beta–канал оновлень.

В Xamarin.iOS є певні обмеження, які пов'язані в основному з тим, що в iOS, як було сказано вище, на відміну від .NET і Mono немає віртуальної машини. Тому виникають труднощі з підтримкою Generic. Причина зрозуміла – на компілятор лягає завдання проаналізувати код і визначити всі можливі конкретизації в тому чи іншому класі і методі. Звідси виникають такі обмеження:

– не рекомендується використовувати Virtual Generic методи, тому що компілятор може не врахувати всі можливі варіанти використання;

– не можна створювати Generic – спадкоємців від класу NSObject, який є базовим у ієрархії Objective – C. Досить серйозне обмеження, яке може певним чином зіпсувати стрункy і красиву архітектуру.

Xamarin, є засобом кросплатформеної розробки, тобто очікувано, що додаток, написаний один раз, може бути запущено на різних мобільних платформах.

Для реалізації цієї задачі необхідно для кожної платформи реалізувати власний шар UI. Тобто код, який відповідає за зовнішній вигляд програми, доведеться написати для кожної платформи окремо. Це ціна за можливість використання нативних механізмів роботи з UI. Якщо розбивати додаток на шари, то виходить така схема:

– Data Layer (DL) – сховище даних, наприклад, база SqlLite або xml–файли;

– Data Access Layer (DAL) – обгортка над сховищем для здійснення CRUD–операцій;

– Business Layer (BL) – шар, що містить бізнес–логіку додатка;

– Service Access Layer (SAL) – шар, який відповідає за взаємодію з віддаленими сервісами (Rest, Json, WCF);

– Application Layer (AL) – шар, що містить платформозавісімий код, який залежить від бібліотек Monotouch або monodroid.dll;

– User Interface Layer (UI) – шар призначеного для користувача

інтерфейсу.

Кросплатформеною є всі верстви, розташовані вище Application Layer. Частка переносимого коду досить сильно залежить від самого додатка, але вона навряд чи може перевищити 50–60%. Інженери Xamarin це розуміють, тому прагнуть до збільшення цієї частки. Як досягнення у вирішенні цієї проблеми можна розглядати бібліотеку Xamarin.Mobile. Вона надає єдиний для різних платформ API для роботи з камерою, контактами та геолокацією. Але використання цієї бібліотеки ніяк не обмежує програміста в застосуванні платформозалежного API, наприклад, за допомогою механізму делегатів [3].

У Xamarin існує власний магазин сторонніх компонентів Xamarin Components. Він інтегрується в IDE і дозволяє в кілька кліків підключати до проекту різні компоненти, написані як інженерами Xamarin, так і сторонніми розробниками. Кількість компонентів, до речі, росте як на дріжджах. Є як платні, так і безкоштовні (на даний момент їх більшість).

Всі компоненти можна розділити на дві частини. Одні надають додаткові елементи призначеного для користувача інтерфейсу, інші є бібліотеками класів. Наприклад варіант для Mono відомої бібліотеки для роботи з Json – Json.NET або ж бібліотека для взаємодії з Rest–сервісами – RestSharp.

Не всі компоненти Кросплатформені, багато - доступні тільки для конкретної платформи. Як згадувалося вище, Xamarin використовує механізм Біндінгу для зв'язування з нативними бібліотеками класів, що дозволяє перенести на C# будь–нативні бібліотеки класів. Крім того, для Xamarin.iOS, наприклад, існує спеціальна утиліта, яка вміє генерувати такі Біндінги автоматично. Власне це дозволяє інженерам Xamarin встигати за всіма нововведеннями iOS. Так, зокрема, в Xamarin.iOS практично відразу після виходу з'явилася можливість використовувати Dropbox API, а так само нові фічі iOS 7 [3].

Щоб здійснювати збереження і обробку даних в додатку використовується база даних (далі – БД). Вона є невід’ємною частиною додатку.

Сучасні бази даних зберігають великі об’єми інформації, тому обробляти її вручну, послідовно переглядаючи і редагуючи дані в таблицях, стає досить важко. Для підвищення ефективності користування базами даних застосовують запити, які дозволяють виконувати множинну обробку даних, тобто одночасно вводити, редагувати та видаляти велику кількість записів, а також вибирати дані з таблиць.

Бази даних становлять невід’ємну складову діяльності сучасних підприємств та організацій. Невпинне зростання обсягів інформації, що зберігається в базах даних, та розширення кола користувачів спонукають до використання систем керування базами даних (далі – СКБД).

Для створення і реалізації бази даних було обрано СКБД PhpMyAdmin – це безкоштовний програмний інструмент, написаний на PHP, який призначений для адміністрування сервера баз даних MySQL або MariaDB. Можна використовувати phpMyAdmin для виконання більшості завдань адміністрування, включаючи створення бази даних, запуск запитів і додавання облікових записів користувачів.

В даний час використовуючи PhpMyAdmin можливо:

– створювати, переглядати, редагувати та видаляти бази даних, таблиці, перегляди, стовпці та індекси;

– відобразити кілька наборів результатів за допомогою збережених процедур або запитів;

– створювати, копіювати, видаляти, перейменовувати і змінювати бази даних, таблиці, стовпці та індекси;

– обслуговувати сервер, бази даних і таблиці, з пропозиціями про конфігурацію сервера;

– виконувати, редагувати і робити закладки будь-якого SQL– оператора, навіть пакетних запитів;

– завантажувати текстові файли в таблиці;

– створювати і читати дампи таблиць;

– здійснювати адміністрування декількох серверів;

– додавати, редагувати та видаляти облікові записи та права користувачів MySQL;

– перевірити цілісність даних в таблицях MyISAM;

– за допомогою запиту за зразком (QBE), створювати складні запити автоматичного підключення необхідних таблиць;

– створювати графічний макет бази даних;

– здійснювати глобальний пошук в базі даних чи її підмножинах;

– здійснювати перетворення збережених даних в будь-який формат, використовуючи набір стандартних функцій, відображення BLOB – data у вигляді зображення або посилання на завантаження;

– відстеження змін в базах даних, таблицях і поданнях;

– підтримка таблиць InnoDB та зовнішніх ключів;

– створити, редагувати, виклик, експорт та видалення збережених процедур і функцій;

– створити, редагувати, експортувати і видалення подій і тригерів.

Основною використовуваною мовою запитів при роботі з базою даних є мова SQL. Вона є реалізацією стандарту ANSI / ISO по структуруванню мови запитів (SQL) з розширеннями.

SQL (англ. Structured Query Language) – мова структурованих запитів – це універсальна мова для створення, модифікації та керування інформацією, яка входить до складу реляційних баз даних.

На сьогоднішній день SQL – це єдиний механізм, який здатний зв’язати прикладне програмне забезпечення та базу даних.

Сама мова володіє кількома видами запитів. Варто відзначити, що будь–який запит SQL має під собою на увазі звернення до бази даних.

У зв’язку з цим прийнято виділяти такі види запитів:

– створення, зміна в базі даних нових або вже існуючих об’єктів;

– отримання даних;

– додавання нових даних в таблицю;

– видалення даних;

– звернення до системи керування базами даних.

Розглянемо основні переваги мови SQL:

– незалежність від існуючої в даній системі СКБД, а саме тексти SQL є універсальними для багатьох СКБД;

– наявність стандартів SQL сприяє «стабілізації» мови;

– декларативність, при роботі з даними, програміст вибирає тільки ту інформацію, яка повинна бути змінена або модифікована.

Перейдемо до недоліків мови SQL:

– складність SQL;

– деяка невідповідність стандартів;

– невідповідність реляційної моделі даних.

Для реалізації роботи додатку з базою даних було використано технологію ADO.NET. Вона забезпечує можливість підключення та управління БД із програмного додатку.

ADO.NET – об'єктно-орієнтована технологія доступу до даних. Надає можливість взаємодії з об'єктами, як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL.

Платформа ADO.NET дозволяє розробникам створювати застосування для доступу до даних, працюючі з концептуальною моделлю застосування, а не безпосередньо з реляційною схемою зберігання. Її метою є зменшення об'єму коду і зусиль по обслуговуванню застосувань, орієнтованих на обробку даних.

Застосування ADO.NET дають наступні переваги:

– застосування можуть працювати з концептуальною моделлю в термінах наочної області, зокрема з успадкованими типами;

– застосування звільняються від жорстких залежностей від конкретного ядра СКБД або схеми зберігання;

– зіставлення між концептуальною моделлю і схемою, специфічною для конкретного сховища, можуть мінятися без зміни коду;

– розробники мають можливість працювати з узгодженою моделлю

об'єктів застосування, яка може бути зіставлена з різними схемами зберігання, які, можливо, реалізовані в різних системах управління даними;

– декілька концептуальних моделей можуть бути зіставлені з єдиною схемою зберігання;

– підтримка інтегрованих в мову запитів (LINQ) забезпечує під час компіляції перевірку синтаксису запиту щодо концептуальної моделі.

Є безліч популярних СКБД, які можуть бути серверними або файловими. Серверні СКБД працюють стабільніше і краще підтримують одночасну роботу декількох користувачів, а файлові СКБД легше впроваджувати і управляти ними після інсталяції застосування клієнтів.

Для кожного конкретного джерела даних потрібен спеціальний постачальник даних .NET. Ця відмінність дещо розмита у разі OleDb і ODBC, оскільки вони за своєю суттю створені для роботи з будь–якою базою даних, сумісною з OleDb або ODBC, але навіть їх спеціальні реалізації знаходяться в окремому постачальнику даних, створеному спеціально для них.

Застосування починається із створення підключення до бази даних. ADO.NET дозволяє організувати з’єднання з нею за допомогою об’єкта підключення, який входить до складу відповідного постачальника даних. Об’єкти підключення (SqlConnection, OleDbConnection та ін.) мають безліч методів, які використовуються для ефективної роботи з’єднання [5]. Клас SqIConnection включає дві події: InfoMessage і StateChange. InfoMessage – наступає, коли з’єднання одержує інформаційне повідомлення від джерела даних. StateChange – наступає при зміні властивості State з’єднання.

Для відстежування таких повідомлень можна скористатися подією InfoMessage класу SqIConnection. Можна також примусити об’єкт SqIConnection реагувати на помилки в запитах (наприклад, на спробу запиту неіснуючої таблиці) за допомогою події InfoMessage, не генеруючи виняткової ситуації. Для цього потрібно привласнити властивості FireInfoMessageEventOnUserErrors об’єкта SqIConnection значення True.

Подія StateChange наступає після зміни значення властивості State об’єкта

Connection. Воно буде особливо корисним під час відображення поточного стану з'єднання (наприклад, в інформаційній панелі застосування).

Для того, щоб відкрити підключення необхідно вказати, яка інформація необхідна для виконання цього завдання, наприклад, ім'я сервера, ідентифікатор користувача, пароль і т. д. Оскільки кожному цільовому джерелу підключення може знадобитися особливий набір інформації, що дозволяє ADO.NET підключиться до джерела даних, обирають гнучкий механізм вказівки всіх параметрів через рядок підключення ConnectionString. Рядок підключення – це набір розділених крапкою з комою пар атрибут значень, що визначають, як слід встановлювати підключення до джерела даних. Нижче приведений перелік атрибутів рядків підключення для встановлення з'єднань:

– Provider – ім'я провайдера (тільки для OLE DB);

– Driver – ODBC–драйвер (тільки для ODBC);

– Connection Timeout – час у секундах для спроб установки з'єднання (за замовчуванням 15 сек.);

– InitialCatalog – ім'я бази даних в SQLServer;

– Data Source – ім'я сервера або шлях до бази даних Access;

– Password – пароль;

– User ID – логін;

– Integrated Security – True, якщо вибирається Windows Authentication, False – якщо SQL Authentication.

При програмуванні застосувань потрібно вибрати найкраще місце для зберігання рядків підключення, з тим щоб досягти більшої зручності обслуговування, полегшити внесення змін у майбутньому і позбавитися необхідності перекомпілювати застосування при його модифікації. Зберігати рядки підключення можна різними способами: жорстко програмувати в коді застосування, у конфігураційному файлі застосування, у реєстрі Windows, представляти за допомогою UDL– файла (файл універсальної вказівки даних – universal data link).

Ефективність використання з'єднань істотно залежить від управління

процесом підключення до баз даних. При цьому можна використовувати поточний стан з’єднання, який зберігається у властивості State. Ця властивість приймає наступні значення типа ConnestionState:

– Broken – з’єднання з джерелом даних розірване, подібне може трапитися тільки після того, як з’єднання було встановлено, у цьому випадку з’єднання може бути або закрито, або повторно відкрито;

– Closed – з’єднання закрите;

– Connecting – триває процес підключення (зарезервовано);

– Executing – з’єднання знаходиться у процесі виконання команди (зарезервовано);

– Fetching – об’єкт з’єднання зайнятий вибіркою даних;

– Open – з’єднання відкрито.

Наприклад, наступний код закриває з’єднання якщо воно відкрито:

```
if (connect.State == ConnectionState.Open)
connect.Close() .
```

Відповідно до функціоналу програмного додатку «Water Way», програма формує список клієнтських замовлень. Таким чином сформовані дані можуть бути переглянуті за допомогою програми Microsoft OneNote. В Microsoft OneNote можливо обрати необхідне розташування створеного файлу та вивести його на друк.

Microsoft Office OneNote – це застосунок для створення нотаток і організації особистої інформації від корпорації Microsoft, що є частиною пакету Microsoft Office.

Найзручнішим є використання програми на планшетному комп'ютері, де присутня можливість рукописного введення тексту і додавання нотаток. Проте, часто Microsoft OneNote використовується і на звичайних комп'ютерах з операційною системою Microsoft Windows. Також можна встановити на пристрої з: Android, Mac, Windows Phone тощо; доступна також вебверсія.

Робочий простір OneNote є порожнім аркушем, в будь-якому місці якого можна робити текстові і рукописні примітки чи малювати, а також записувати

голосові коментарі. Також інформацію в OneNote можна перетягати мишкою з вікна браузера. Тут відсутня іконка «Зберегти», бо введені дані зберігаються автоматично. Зроблені записи можна надсилати електронною поштою як в форматі HTML, так і у вигляді вкладеного файлу OneNote [6].

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Розробка логічної структури проекту

Програмний продукт складається з двох додатків – для клієнта та для співробітників (водій/адміністратор). Блок-схема алгоритму визначення рівня доступу користувача відображена на рис. 2.3.

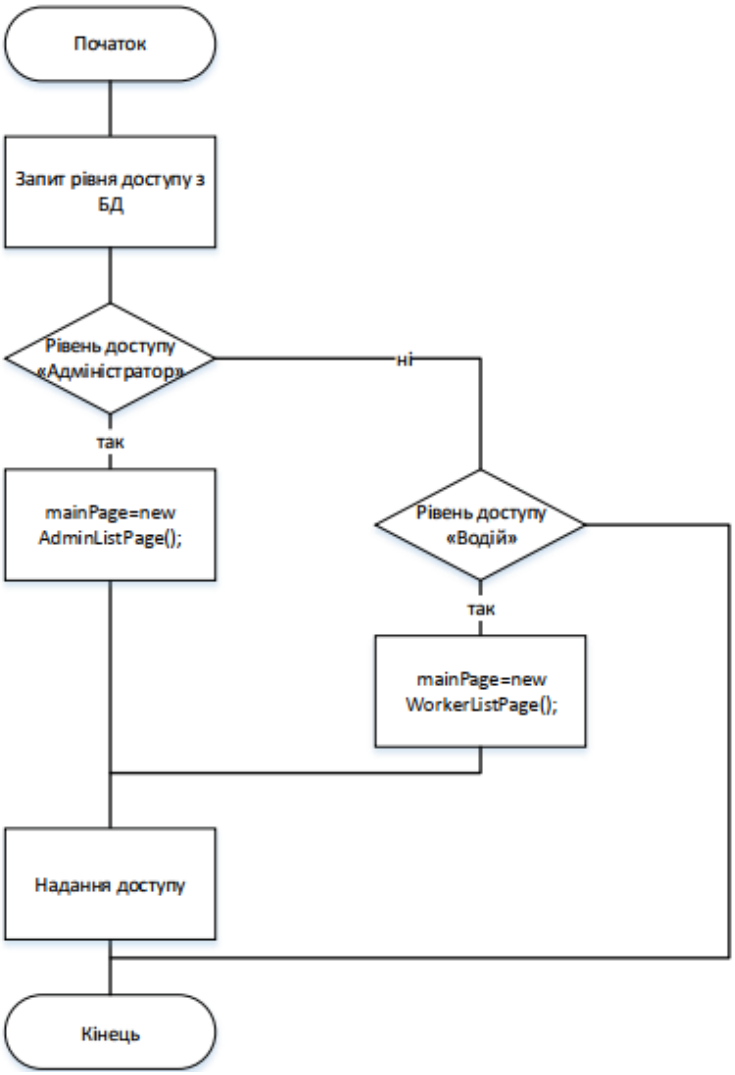

Рис. 2.3. Блок-схема процесу визначення рівня доступу користувача

Діаграма прецедентів (Use-case діаграма) візуально відображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання); описує функціональні аспекти системи (бізнес логіку) [7].

Розглянемо Use-case діаграму дій клієнта, водія та адміністратора представлену на рис. 2.4.

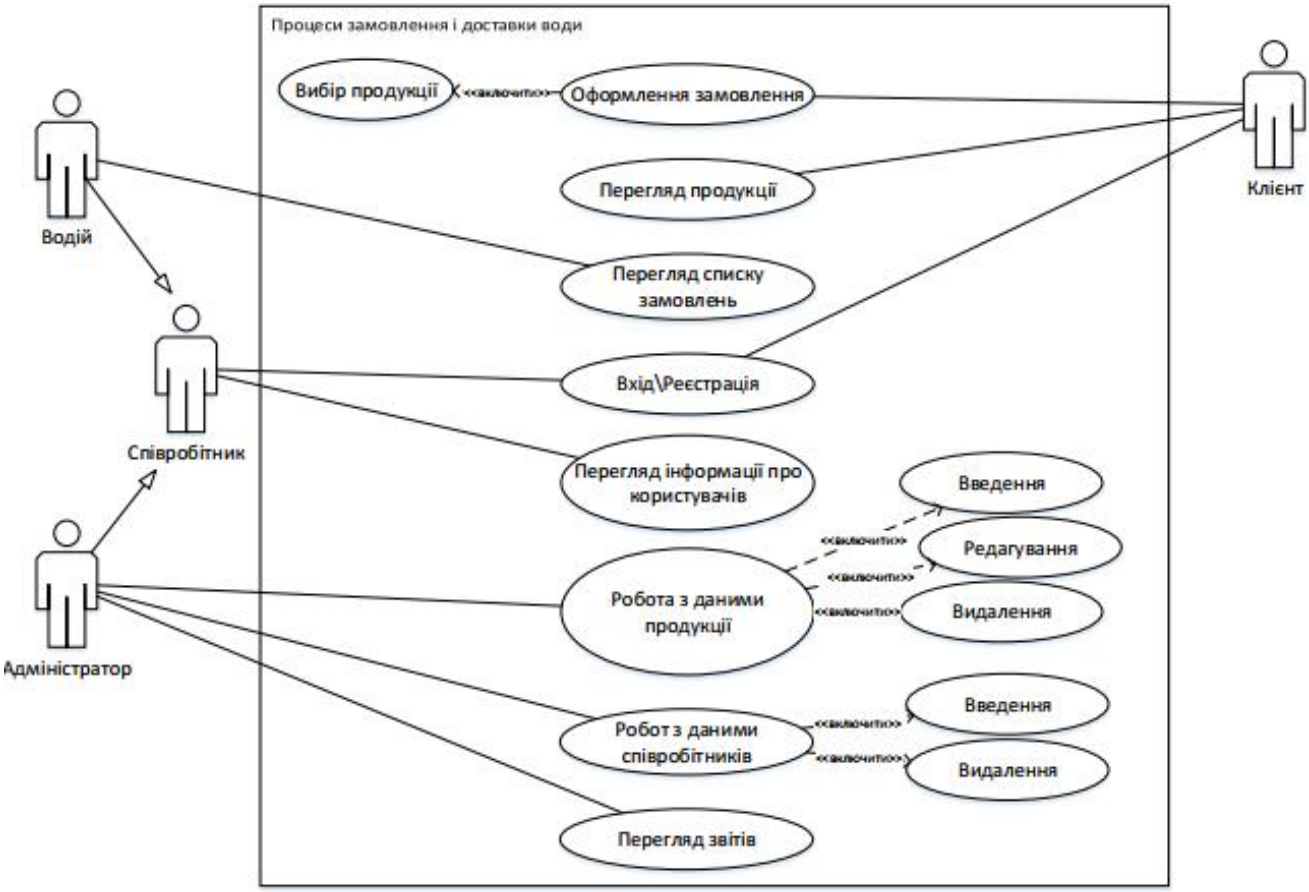

Рис. 2.4. Use–case діаграма взаємодії клієнта та співробітників

Побудована діаграма прецедентів описує наступні функції:

– клієнт:оформлення замовлення, перегляд продукції, вхід/реєстрація;

– водій: перегляд списку замовлень, вхід/реєстрація та перегляд інформації про замовників;

– адміністратор: вхід/реєстрація, перегляд інформації про користувачів,

робота з даними продукції, даними співробітників, та перегляд звітів.

Головна ціль додатку – створення платформи для взаємодії користувача з продавцем та автоматизація процесу замовлення та доставки води. Цей процес зображено на діаграмі послідовностей (SEQUENCE DIAGRAM) (рис. 2.5).

Діаграма послідовності (англ. sequence diagram) – різновид діаграми в UML. Діаграма послідовності – відображає взаємодії об'єктів впорядкованих за часом. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, які пов'язані з повідомленнями [8].

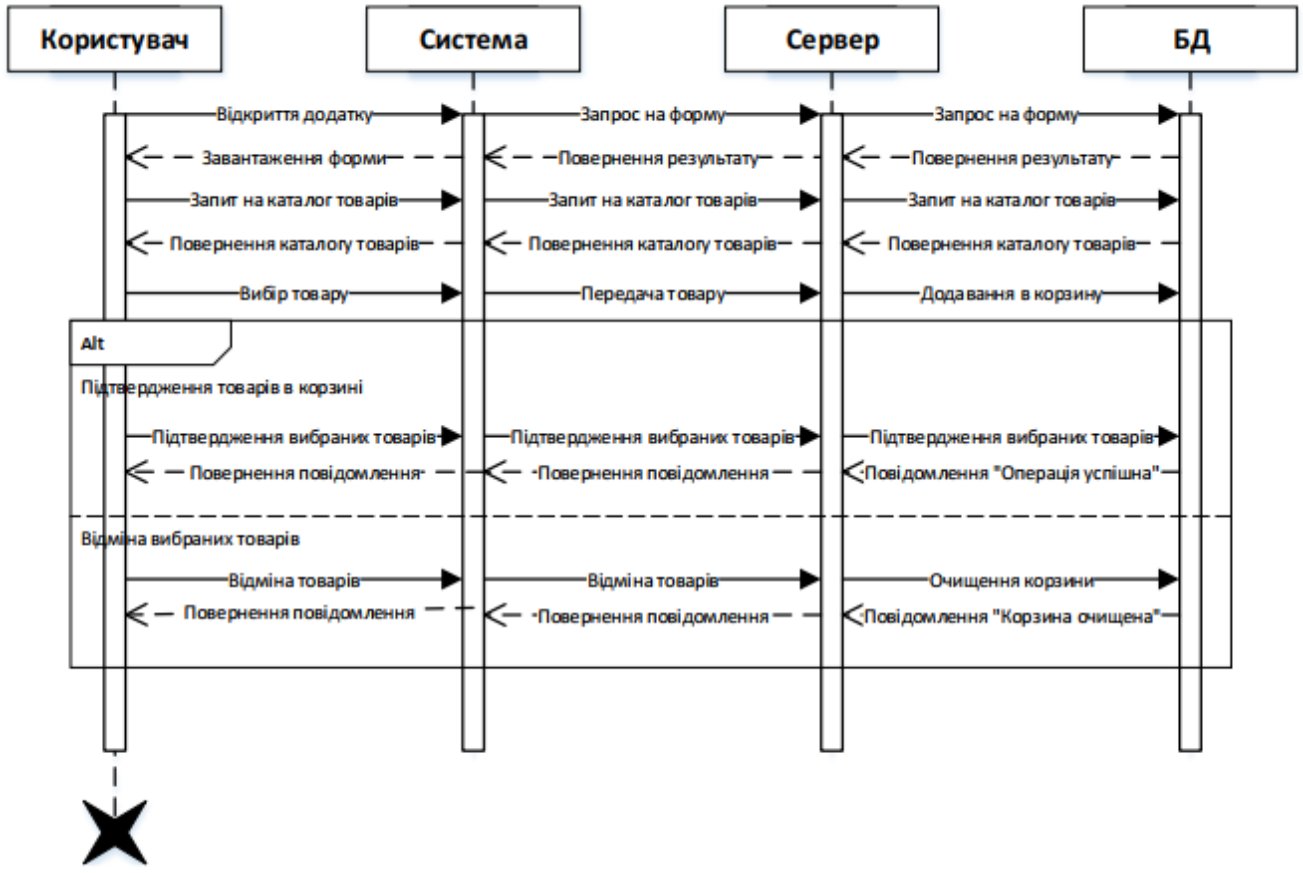

Рис. 2.5. Sequence diagram процесу створення замовлення продукції

2.5.2. Проектування бази даних проекту

Зберігання та обробка інформації – це невід'ємний процес роботи кожного додатку та системи. Для розроблюваного додатку було обрано систему керування базами даних phpMyAdmin. База даних розташовується в Інтернеті на хостингу www.beget.com, до якої мають доступ два додатки – для клієнта та для співробітників.

Спроектуємо базу даних, яка буде зберігати та обробляти дані про користувачів системи, про товари та акції.

Виділимо базові сутності предметної області з атрибутами:

– «Користувачі». Атрибути: ім'я, прізвище, логін, пароль, адреса, нотації;

– «Товари». Атрибути: назва, логотип, опис, вартість, категорія;

– «Акції». Атрибути: дата початку, дата закінчення, товар.

«Замовлення» будемо розглядати як зв'язок між клієнтами та товарами.

Побудуємо ER-діаграму бази даних системи (рис. 2.6).

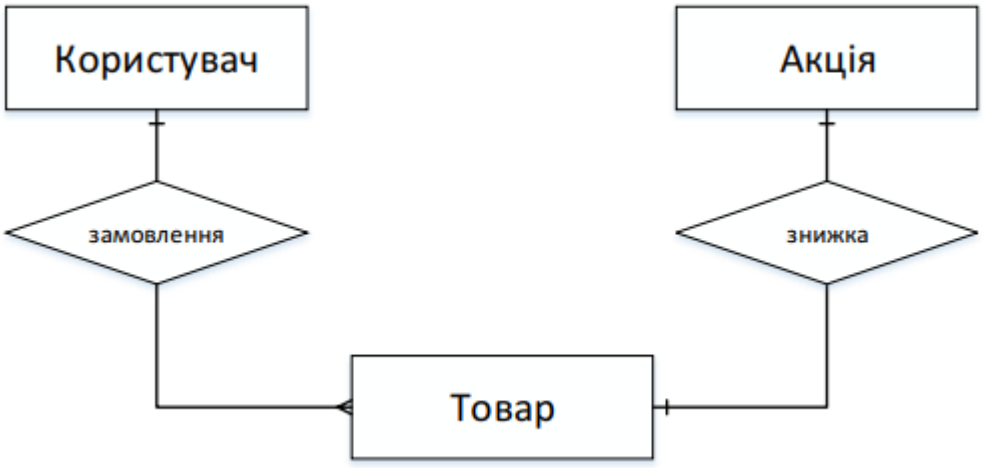

Рис. 2.6. ER-діаграма бази даних

Система складається для обслуговування наступного кола користувачів:

– адміністрація (дирекція);

– водії;

– клієнти.

Виділимо наступні функціональні можливості бази даних:

– ведення бази даних (запис, читання, модифікація, видалення);

– забезпечення логічної несуперечності;

– забезпечення захисту даних від несанкційного або випадкового доступу;

– забезпечення можливістю сформувати запит на мові маніпулювання даними.

База даних складається на основі схеми БД. Для переведення ER-діаграми в схему БД приведемо уточнену ER-діаграму, яка містить атрибути сутностей (рис. 2.7).

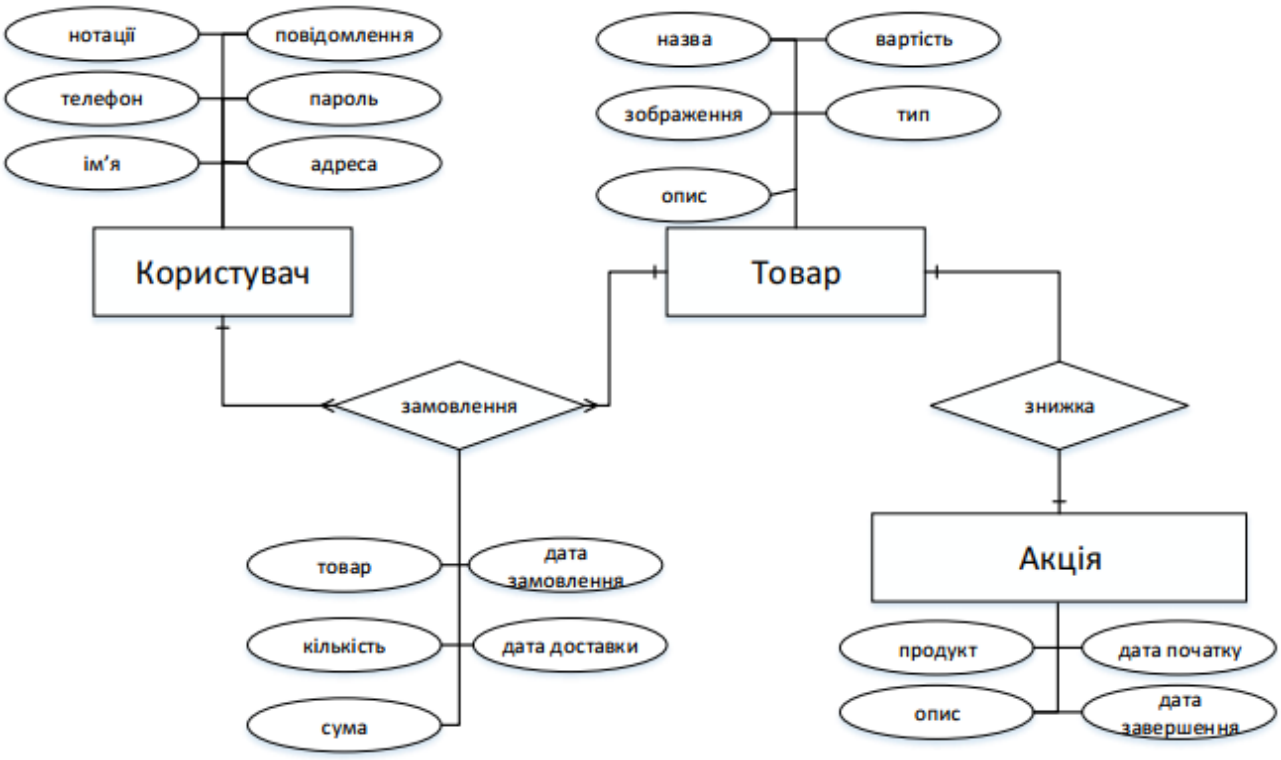

Рис. 2.7. Уточнена ER-діаграма бази даних

На даній схемі між таблицями «Користувач» та «Замовлення», «Замовлення» та «Продукт» – зв'зок 1:m (один до багатьох), так як

користувач може створювати безліч замовлень, і один товар може бути замовлений кілька разів. Таблиці «Продукт» та «Акція» також мають зв'язок 1:m так як товар має одну знижку, але акція може бути розподілена на декілька товарів. Кожне реляційне відношення відповідає одній сутності і в нього заносяться всі атрибути і сутності. Для кожного відношення необхідно визначити первинний ключ і зовнішні ключі (якщо вони є). У випадку, якщо базове відношення немає потенційних ключів, то вводиться сурогатний первинний ключ, який не має смислового навантаження і слугує лише для ідентифікації записів. Для відношень необхідно ввести сурогатні ключі id, для їх ідентифікації, так як таблиці не мають потенційних ключів.

На етапі складання реляційних відносин побудуємо таблицю для кожної сутності.

Таблиця – це об'єкт, призначений для збереження даних у вигляді записів (рядків) та полів (стовпців). Кожна таблиця використовується для збереження відомостей по одному конкретному питанню [9].

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

– C – символьне значення до 255 символів;
– D – дата;
– B – масив двійкових даних;
– BL – булевий тип;
– M – грошовий тип даних;
– 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. 2.1 – 2.4).

Таблиця 2.1

Схема відносин «Акції» (discount)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код продукту	idProduct	N	FK
Опис	description	C(500)	
Дата початку	dateFrom	D	
Дата закінчення	datetTo	D	

Таблиця 2.2

Схема відносин «Товари» (product)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Назва	title	C(50)	
Опис	description	C(500)	
Зображення	imageSource	B	
Вартість	cost	N	

Таблиця 2.3

Схема відносин «Замовлення» (order)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код користувача	idUser	N	FK
Код продукту	idProduct	N	FK
Кількість	amount	N	
Вартість	cost	M	
Дата замовлення	date	D	
Дата доставки	dateOfOrder	D	

Таблиця 2.4

Схема відносин «Користувачі» (user)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Ім’я	name	C(50)	
Телефон	phone	C(10)	
Пароль	password	C(50)	
Адреса	adress	N	
Нотації	notation	C(50)	
Повідомлення	notification	C(50)	

Нормалізація схеми бази даних – покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Процедура нормалізації БД полягає в усуванні надмірності даних та виявленні функціональних залежностей. Усунення надмірності даних гарантує компактність наборів даних за рахунок уникнення їх зайвого дублювання та унеможливлює виникнення аномалій вставки, вилучення й оновлення кортежів після фізичної реалізації БД

Нормальна форма – властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки.

Існують такі рівні нормалізації: перша нормальна форма (1НФ), 2НФ, 3НФ, нормальна форма Бойса-Кодда (БКНФ), 4НФ, 5НФ. Але дотепер жодна з реляційних СКБД не надає належної підтримки усім п'яти нормальним формам. Суть справи полягає в тому, що в повністю нормалізованій БД для виконання запиту треба з'єднати настільки багато таблиць, що продуктивність такої системи не зможе задовольнити користувачів. Тому на практиці використовують лише перші три рівня нормалізації – 1НФ, 2НФ, 3НФ.

Таким чином, схема реляційної бази даних переходить у першу, другу,

третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми n та всіх попередніх нормальних форм, тоді вважається, що це відношення знаходиться у нормальній формі рівня n [9].

Для приведення таблиць до першої нормальної форми (1НФ) необхідно скласти прямокутні таблиці і розбити складні атрибути на прості, а багатозадачні атрибути винести в окремі відношення.

Розділимо атрибут адреса на атбрибути: місто, вулиця, номер дому, номер квартири, поверх, код під'їзду. Так як адреса багатоскладна, то створимо сутність «Адреса».

Відношення знаходиться в другій нормальній формі (2НФ), якщо воно знаходиться в 1НФ та кожен не ключовий атрибут функціонально повністю залежить від первинного ключа.

В нашому випадку не ключові атрибути відношень функціонально повністю залежать від первинних ключів.

Щоб привести відношення до третьої нормальної форми (3НФ) необхідно привести його до 2НФ, і винести всі не ключові поля, вміст яких може відноситися до декількох записів сутності в окремі таблиці.

Сутність «Користувач» має два атрибути «нотації» та «повідомлення», які можуть мати декілька записів. Створимо дві нові сутності «Нотації» та «Повідомлення».

Для забезпечення безпеки доступу до акаунта адміністратора створимо окрему сутність для робітників та редакторів фірми «Робітники», яка буде містити авторизаційні дані та особисту інформацію.

Відношення даної бази даних не порушують четверту нормальну форму, так як не містять нетривіальних багатозначних залежностей.

Після проведення перетворень схема бази даних виглядає наступним чином (рис. 2.8):

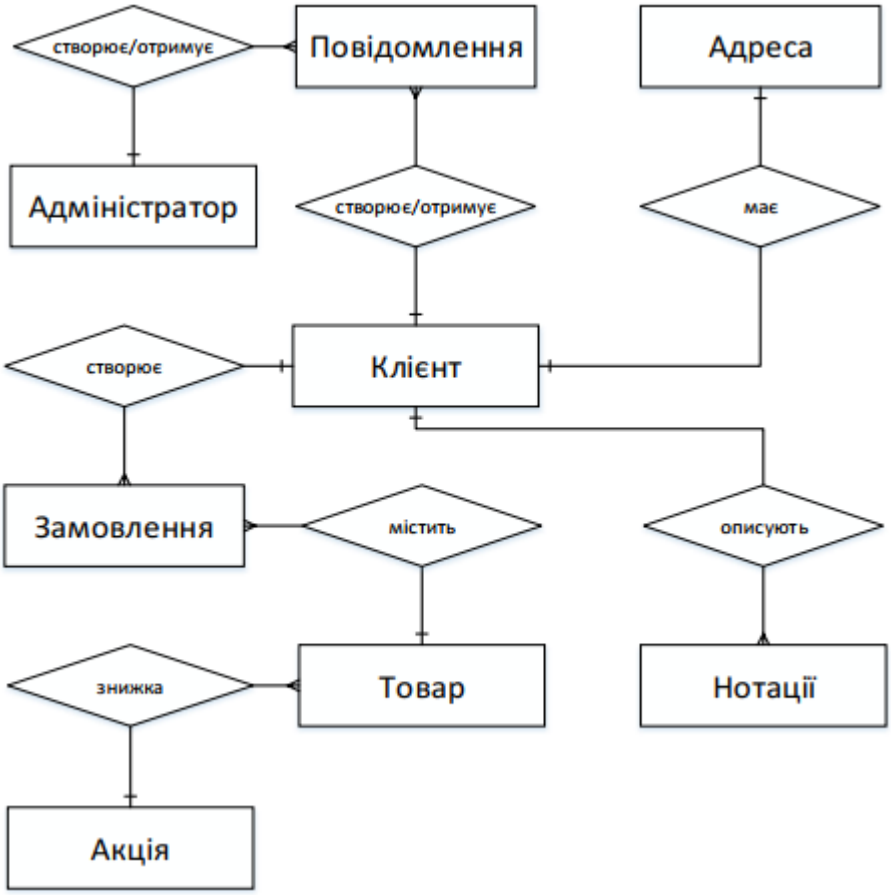

Рис. 2.8. Остаточна схема РБД для бази даних

Остаточні схеми відносин бази даних з зазначенням ключів та інших обмежень цілісності приведені нижче у таблицях 2.4 – 2.11.

Таблиця 2.4

Схема відносин «Замовлення» (order)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код користувача	idUser	N	FK
Код продукту	idProduct	N	FK
Кількість	amount	N	
Вартість	cost	M	
Дата замовлення	date	D	
Дата доставки	dateOfOrder	D	

Таблиця 2.5

Схема відносин «Акції» (discount)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код продукту	idProduct	N	FK
Опис	description	C(500)	
Дата початку	dateFrom	D	
Дата закінчення	datetTo	D	

Таблиця 2.6

Схема відносин «Користувачі» (user)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Ім’я	name	C(50)	
Телефон	phone	C(10)	
Пароль	password	C(50)	
Код адреси	idAdress	N	FK
Нотації	notation	C(50)	
Повідомлення	notification	C(50)	

Таблиця 2.7

Схема відносин «Адреса» (adress)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код користувача	idClient	N	FK
Місто	city	C(50)	
Вулиця	street	C(50)	
Номер дому	houseNumber	C(50)	
Номер квартири	flatNumber	N	
Поверх	floor	N	
Код під’їзду	code	N	

Таблиця 2.8

Схема відносин «Товари» (product)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Назва	title	C(50)	
Опис	description	C(500)	
Зображення	imageSource	B	NULL
Вартість	cost	N	

Таблиця 2.9

Схема відносин «Нотації» (notation)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код клієнта	idClient	N	FK
Нотація	notation	C(50)	
Дата	date	D	

Таблиця 2.10

Схема відносин «Робітник» (worker)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Ім’я	name	C(50)	
Телефон	phone	C(10)	
Пароль	password	C(50)	
Посада	role	C(50)1	

Таблиця 2.11

Схема відносин «Повідомлення» (notification)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код	id	N	PK
Код клієнта	idClient	N	FK
Відправник	Sender	C(50)	
Текст	text	C(50)	
Дата	date	D	
Прочитано	seen	BL	

Опишемо для кожної групи користувачів права доступу до кожної таблиці і до кожного поля (атрибута):

– адміністратор БД: має доступ до всіх даних (по запису), може змінювати структуру бази даних і зв’язки між відношеннями. Встановлює права доступу для всіх інших груп;

– водії маю доступ в режимі читання до відношень «Замовлення» та «Користувачі», а в режимі читання і запису до відношення «Нотації»;

– клієнти: мають доступ в режимі читання до відношень «Товари», «Замовлення», «Акції».

2.5.3. Розробка компонентів додатку

Для розробки програмного продукту було обране середовище програмування Microsoft Visual Studio, фреймворк Xamarin та мова програмування C#. Перелік файлів програмного продукту зображено на рис. 2.9.

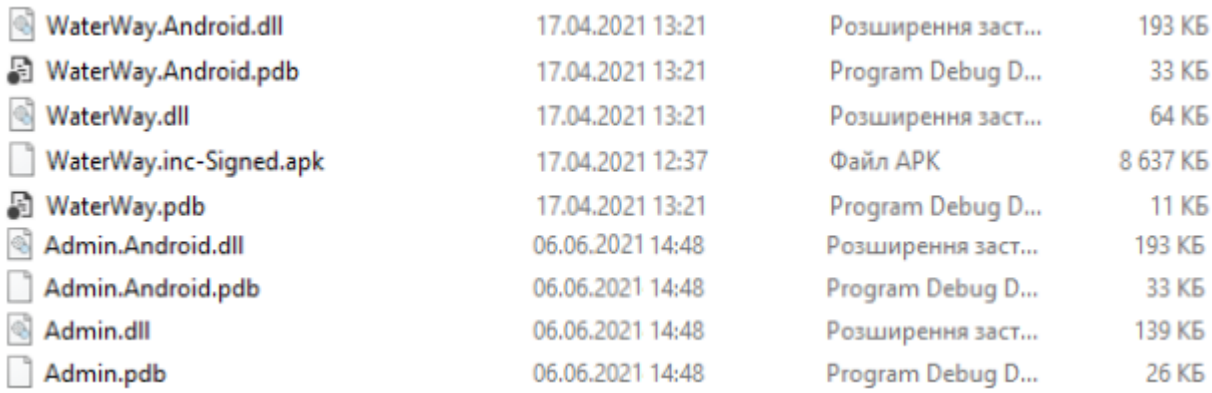

Рис. 2.9. Перелік файлів програмного продукту WaterWay

Система складається зі сторінок, які описуються двома файлами – .xaml та .xaml.cs. Файли з розширенням .xaml описують компоненти сторінки, їх розміщення та властивості. У файлі .xaml.cs створюється логіка програми, описуються події на компоненти. Також використовуються класи та інтерфейси, які зберігаються у файлах з розширенням .cs.

Для навігації в програмі використовуються вбудовані інструменти переходу між формами (Page) «TabbedPage» та «NavigationPage».

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. 2.10) [3].

Інструмент NavigationPage допоміг втілити шаблонну навігацію за принципом Model–View–View–Model (MVVM) (рис. 2.11).

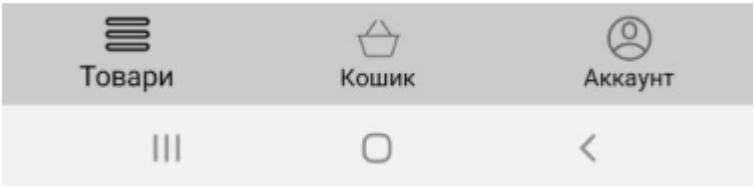

Рис. 2.10. Інструмент TabbedPage

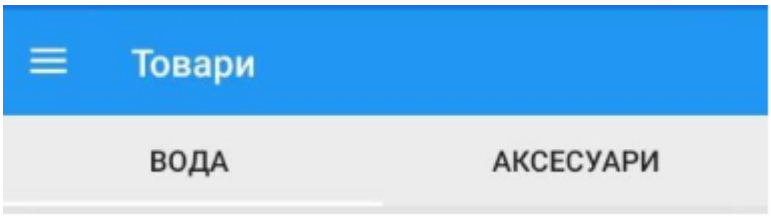

Рис. 2.11. Інструмент NavigationPage

Програмний додаток для користувача багатовіконний. Для кожного вікна використовується окрема форма. Було використано наступні компоненти, для будування форм додатку:

– Label – текстова мітка;
– Entry – поле для вводу;
– Image – зображення;
– Edit – багаторядкове поле для вводу;
– Button – кнопка;
– ListView – список;
– ScrollView – контейнер прокрутки;
– ToolBar – панель додаткових елементів;
– Grid – сітка;
– DatePicker – вибір дати;
– StackLayout – контейнер компоновки.

При запуску додатку для клієнта запускається форма авторизації «Login». Блок-схема процесу авторизації користувача в системі наведена на рис. 212. Ескіз форми зображено на рис. 2.13. Події компонентів даної форми відображені в таблиці 2.12.

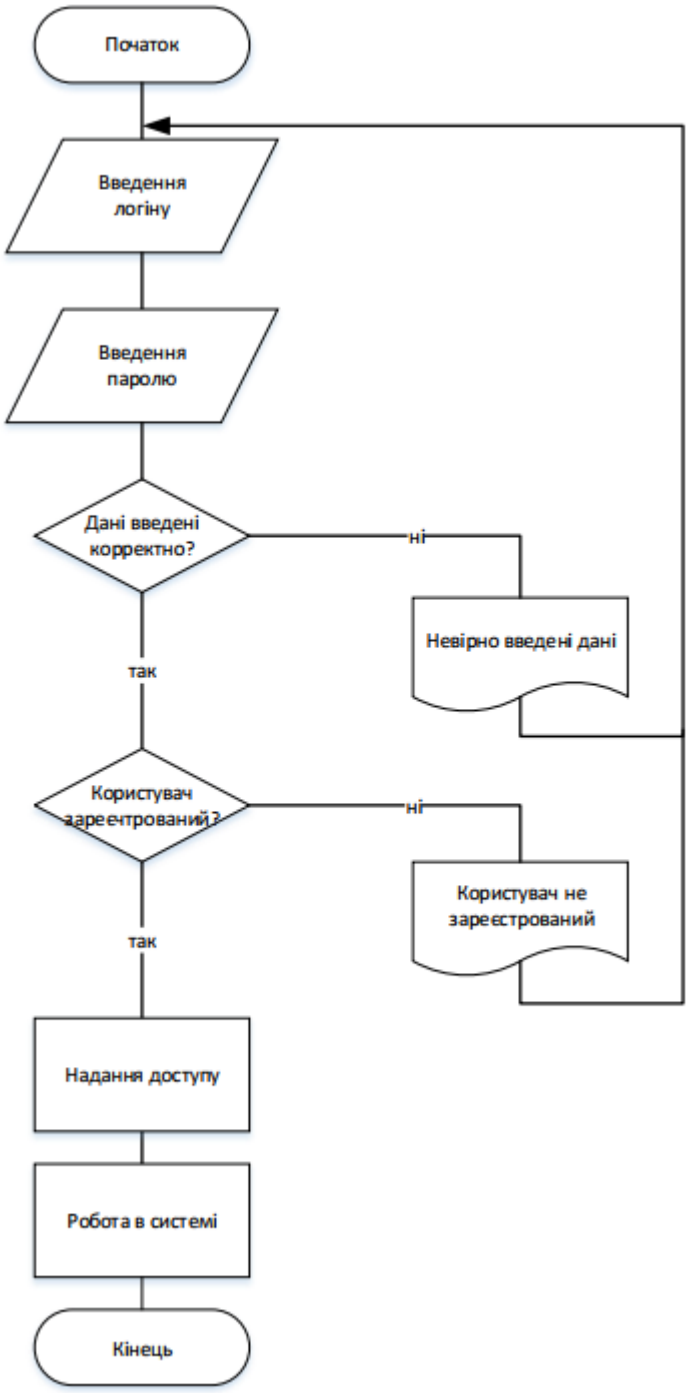

Рис. 2.12. Блок-схема процесу авторизації користувача в системі

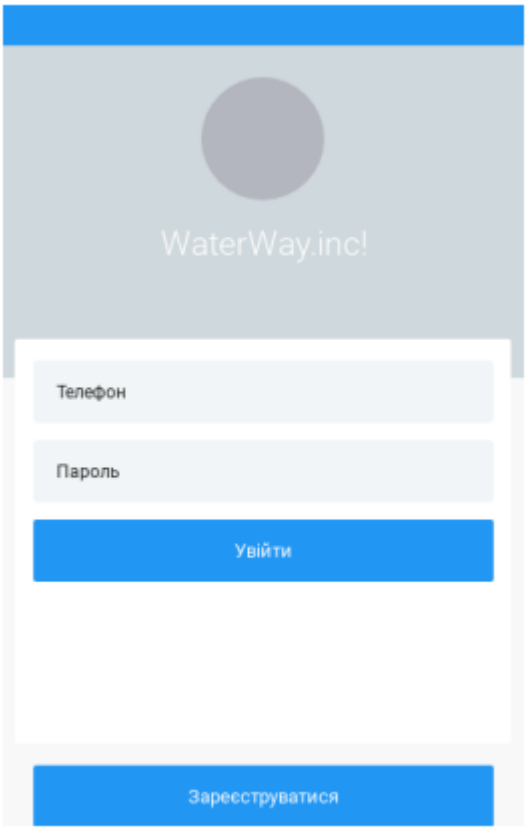

Рис. 2.13. Ескіз форми авторизації «Login»

Таблиця 2.12

Події форми «Login»

Подія	Опис
Login()	Конструктор класу
ButtonVisiblePassword_Clicked	Видімість поля для паролю
ButtonLogin_Clicked	Авторизація в системі
ButtonRegistration_Clicked	Перехід на форму реєстрації

Для реєстрації клієнта в системі створена форма «Registration». На дану форму користувач може перейти з форми «Login» Події компонентів описано в таблиці 2.13. Ескіз форми зображено на рисунку 2.14.

Таблиця 2.13

Події форми «Registration»

Подія	Опис
Registration()	Конструктор класу
ButtonVisiblePassword_Clicked	Видімість поля для паролю
ButtonRegistration_Clicked	Реєстрація в системі
ButtonLogin_Clicked	Перехід на форму авторизації
check	Перевірка введення даних
RegistrationSecondButton_Clicked	Реєстрація адреси
RegistrationSecondPicker_SelectedIndexChanged	Вибір типу сектора

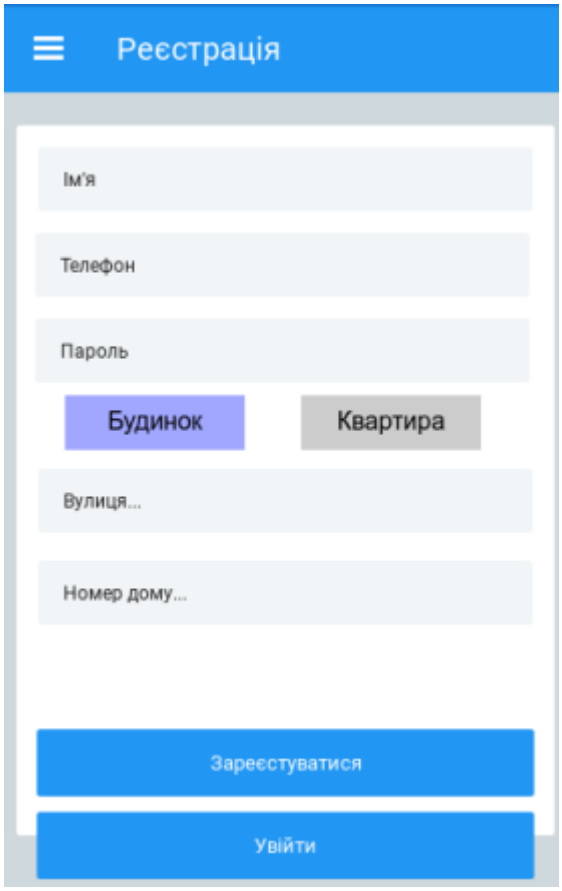

Рис. 2.14. Ескіз форми «Registration»

Після реєстрація або авторизації в системі клієнт потрапляє на форму «ProductPage» (рис. 2.15). Події компонентів описано в таблиці 2.14.

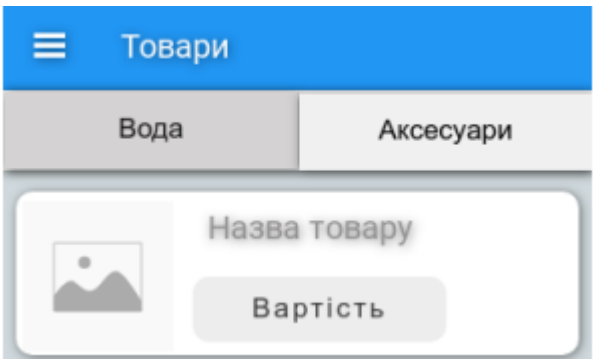

Рис. 2.15. Ескіз форми «ProductPage»

Таблиця 2.14

Події форми «ProductPage»

Подія	Опис
ProductPage()	Конструктор класу
List<Product> productList	Список основних товарів
List<Product> accessoriesList	Список додаткових товарів
connectToDB	З’єднання з БД
ProductAccessoriesListView_ItemTapped	Вибір основного товару
ProductWaterListView_ItemTapped	Вибір додаткового товару
SetProductPage()	Завантаження даних форми
listViewFirst_ItemSelected	Товари головної групи
listViewSecond_ItemSelected	Товари додаткової групи
LaunchSelectProduct	Перехід на форму опису товару

Після вибору товару клієнт потрапляє на форму опису товару (рис. 2.16). Події компонентів описано в таблиці 2.15.

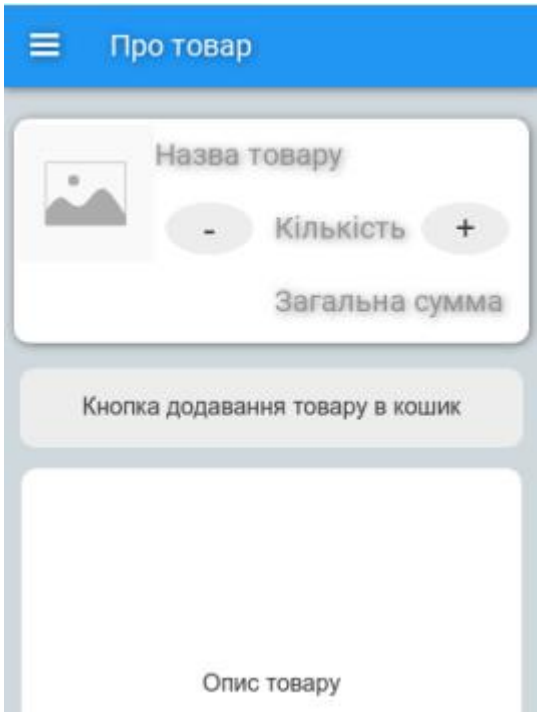

Рис. 2.16. Ескіз форми «AboutProductPage»

Таблиця 2.15

Події форми «AboutProductPage»

Подія	Опис
AboutProductPage ()	Конструктор класу
LaunchAboutProductPageMinusButton_Clicked	Зменшення кількості товарів
LaunchAboutProductPagePlusButton_Clicked	Збільшення кількості товарів
LaunchAboutProductPageAddInBusketButton_ Clicked	Додавання товару в кошик
LaunchAboutProductPageCall	Виклик телефонного меню

Блок-схема процесу замовлення відображена на рис. 2.17.Форма опису товару дозволяє додати продукт у кошик (рис. 2.18). Події компонентів форми «BasketPage» описано в таблиці 2.16.

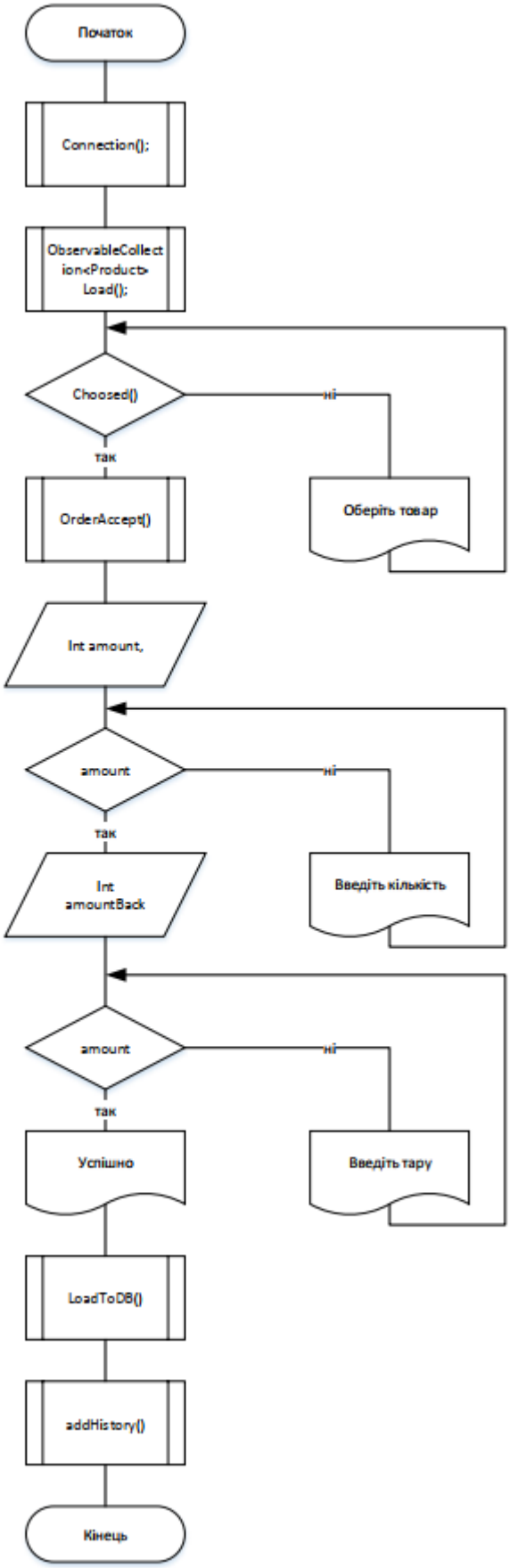

Рис. 2.17. Блок-схема процесу створення замовлення клієнтом

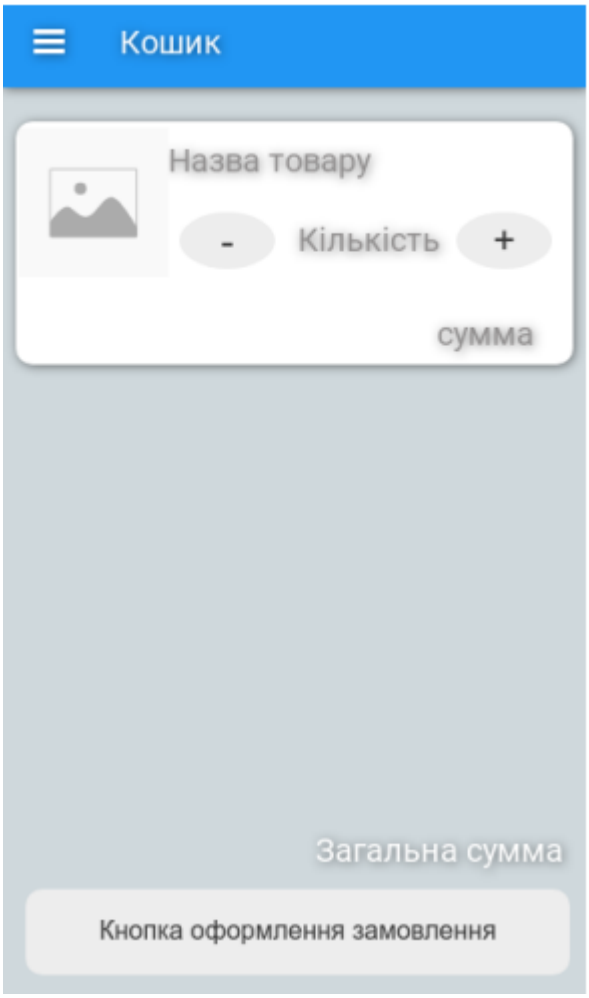

Рис. 2.18. Ескіз форми «BasketPage»

Таблиця 2.16

Події форми «BasketPage»

Подія	Опис
BasketPage ()	Конструктор класу
BasketPageClear()	Очищення колекції
listView_ItemSelected	Вибір елементу колекції
BasketPageAddInBusketButton_ Clicked()	Форма підтвердження товару

Оформлене замовлення записується в історію. Переглянути її можна на формі «HistoryPage» (рис. 2.19). Події компонентів описано в таблиці 2.17.

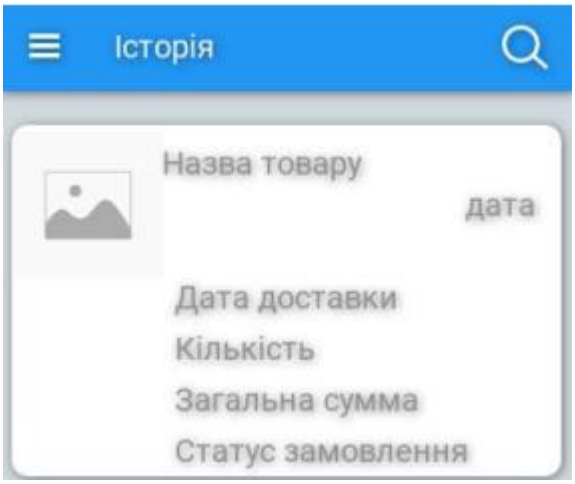

Рис. 2.19. Ескіз форми «HistoryPage»

Таблиця 2.17

Події форми «HistoryPage»

Подія	Опис
HistoryPage ()	Конструктор класу
HistoryList_CollectionChanged	Зміна колекції
HistoryButtonClear_Clicked	Очищення історії
HistorySearchEntry_TextChanged	Пошук по колекції
HistorySearchButton_Clicked	Виклик поля для пошуку

Налаштування аккаунту користувача розташовуються на формі «AccountPage» (рис. 2.20). Події компонентів форми описано в таблиці 2.18.

Таблиця 2.18

Події форми «AccountPage»

Подія	Опис
AccountPage ()	Конструктор класу
AccountPageExit()	Вихід з аккаутну
AccountPageChangeButton()	Вибір типу налаштувань
AccountPageSave()	Збереження змін

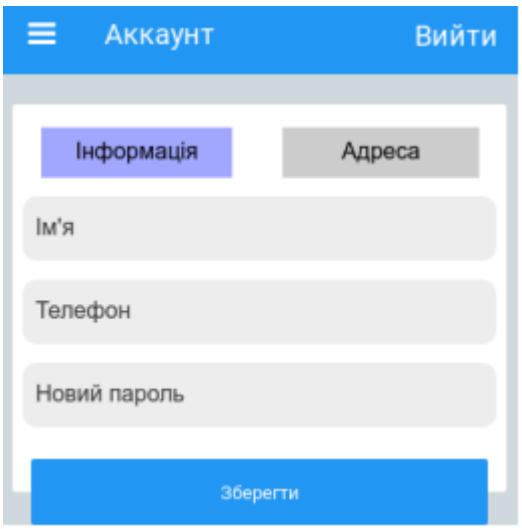

Рис. 2.20. Ескіз форми «AccountPage»

Підтримка користувача реалізована на формі «SupportPage» (рис. 2.21). Події компонентів форми описано в таблиці 2.19.

Таблиця 2.19

Події форми «SupportPage»

Подія	Опис
SupportPage ()	Конструктор класу
SupportPageClear_Clicked()	Очищення колекції
SupportPageSent_Clicked()	Вібправлення повідомлення
SetSupportPage()	Завантаження даних форми

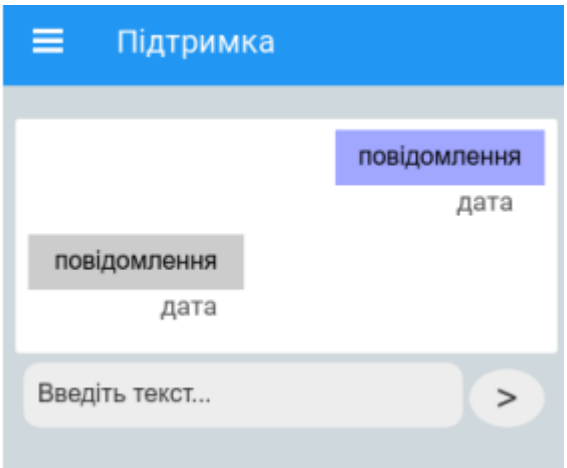

Рис. 2.21. Ескіз форми «SupportPage»

Програмний продукт для співробітників має два рівня доступу та різний функціонал для водія і адміністратора. Після авторизації користувача з рівнем доступу водій відкривається форма зі списком замовлень «ListPage» (рис. 2.22). Події компонентів форми описано в таблиці 2.20.

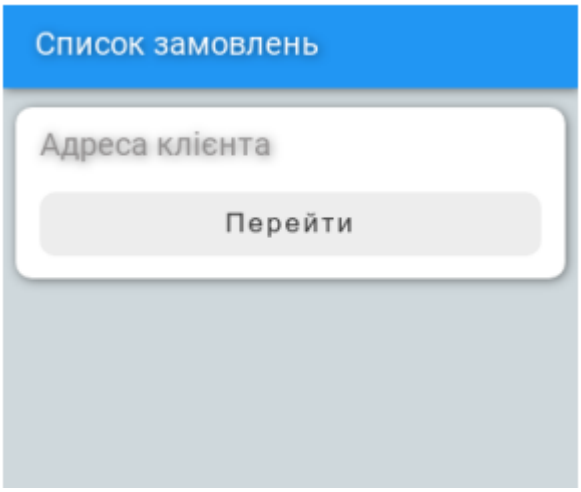

Рис. 2.22. Ескіз форми «ListPage»

Таблиця 2.20

Події форми «ListPage»

Подія	Опис
ListPage ()	Конструктор класу
ListView_ItemSelected()	Вибір елемента колекції
Button_Clicked()	Форма інформація про замовлення
ListView_ItemTapped()	Виклик елемента списку
Connection()	Підключення до БД
SetListPage()	Завантаження даних форми

Після вибору замовлення водій потрапляє на форму інформації про клієнта та замовлення, на якій відображаються дані про клієнта та список замовлень з детальною інформацією (рис. 2.23). Події компонентів форми «Order Page» описано в таблиці 2.21.

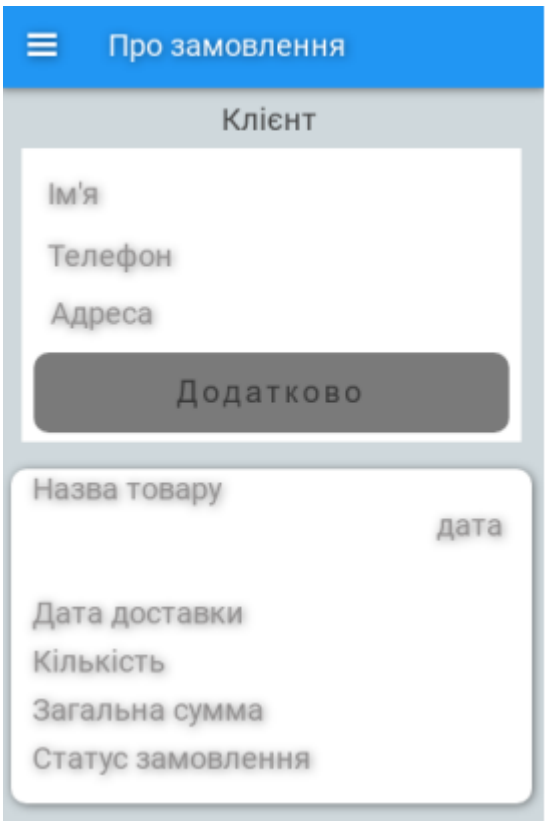

Рис. 2.23. Ескіз форми «OrderPage»

Таблиця 2.21

Події форми «OrderPage»

Подія	Опис
OrderPage ()	Конструктор класу
OrderPageCallButton()	Дзвінок клієнту
OrderPageAdditionButton()	Форма нотації про клієнта
OrderPageRouteButton()	Прокладання маршруту
OrderPageDoneButton()	Виконання замовлення
SetOrderPage(int idClient,bool isDone)	Завантаження даних форми

Пройшовши авторизацію з рівнем доступу адміністратор, відкривається головна форма з даними про товари «ProductListPage» (рис. 2.24). Події компонентів форми описано в таблиці 2.22.

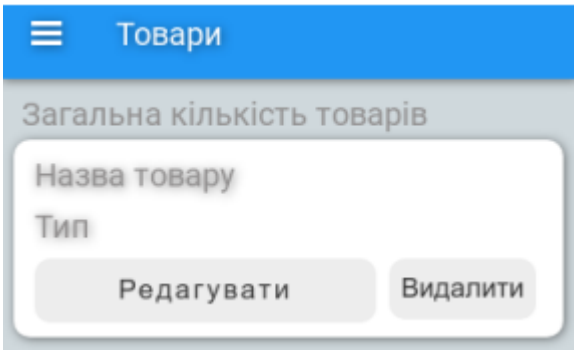

Рис. 2.24. Ескіз форми «ProductListPage»

Таблиця 2.22

Події форми «ProductListPage»

Подія	Опис
ProductListPage ()	Конструктор класу
ProductListPageAddButton()	Додавання товару
SetProductListPage()	Завантаження даних форми
EditButton_Clicked()	Редагування товару
DeleteButton_Clicked()	Видалення товару
Connection()	Підключення до БД

Дані про акції та знижки відображаються на формі «DiscountPage» (рис. 2.25). Події компонентів форми описано в таблиці 2.23.

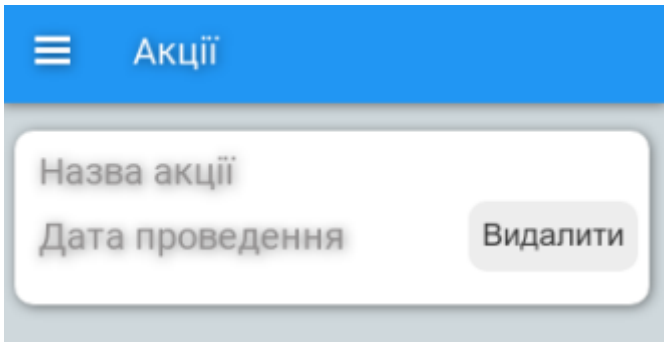

Рис. 2.25. Ескіз форми «DiscountPage»

Таблиця 2.23

Події форми «DiscountPage»

Подія	Опис
DiscountPage ()	Конструктор класу
DiscountPageAddButton()	Додавання акції
SetDiscountPage()	Завантаження даних форми
DeleteButton_Clicked()	Видалення акції

Доступ до бази клієнтів відбувається на формі «WorkerPage» (рис. 2.26). Події компонентів форми описано в таблиці 2.24.

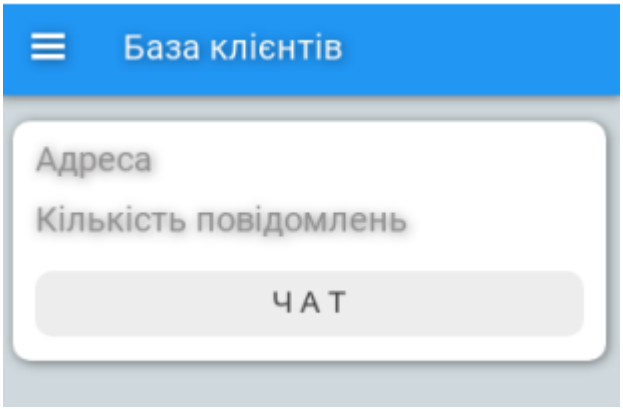

Рис. 2.26. Ескіз форми «ClientPage»

Таблиця 2.24

Події форми «ClientPage»

Подія	Опис
ClientPage ()	Конструктор класу
ClientPageClear_Clicked()	Очищення колекції
SetClientInfoPage()	Завантаження даних форми
ClientPageSent_Clicked()	Відправлення повідомлення

Дані про співробітників відображаються на формі «WorkerPage» (рис. 2.27). Події компонентів форми описано в таблиці 2.25.

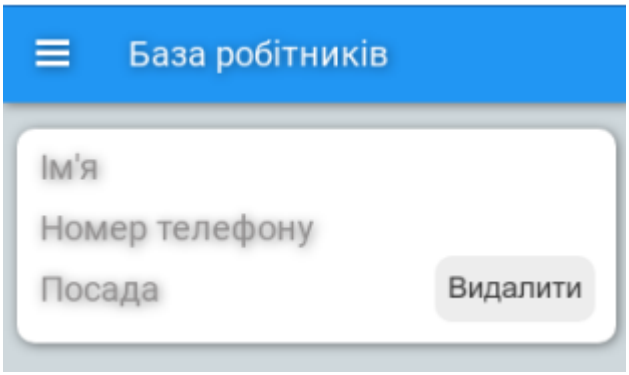

Рис. 2.27. Ескіз форми «WorkerPage»

Таблиця 2.25

Події форми «WorkerPage»

Подія	Опис
WorkerPage ()	Конструктор класу
WorkerPageAddButton()	Додавання робітника
Set WorkerPage ()	Завантаження даних форми

2.6. Обґрунтування та організація вхідних та вихідних даних програми

У якості вхідних даних програми виступають:

– дані про клієнта (ім’я, номер телефону, адреса);

– дані про працівника (ім'я, посада, номер телефону, адреса);

– дані про замовлення (дата доставки, дата замовлення, кількість, ціна).

Вихідними даними будуть виступати:

– сформовані перелік замовлень;

– перелік клієнтів та працівників;

– сформований найкоротший маршрут до адреси замовника.

Щоб здійснювати збереження і обробку даних в додатку використовується база даних

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки програмного забезпечення використовувався ПК з наступною конфігурацією:

- процесор: Intel® Core™2 Duo CRU @ 3.06GHz;
- розрядність: 64-bit, 32-bit;
- RAM: 6.00 GB.

Для тестування та демонстрації роботи даної системи використовувався такий мобільний пристрій: смартфон Huawei Y3 2017 (CRO-U00) DualSim з наступними характеристиками: екран (5", TFT, 854x480)/ MediaTek MT6580M (1.3 ГГц)/ / RAM 1 ГБ/ 8 ГБ вбудованої пам'яті + підтримка microSD/SDHC (до 128 ГБ)/ 3G/ GPS/ підтримка 2-х SIM-карток (Micro-SIM)/ Android 6.0 (Marshmallow).

2.7.2. Використані програмні засоби

Для розробки програмного продукту було обране середовище програмування Microsoft Visual Studio, фреймворк Xamarin та мова програмування C#. Для створення і реалізації бази даних було обрано СКБД PhpMyAdmin. Для реалізації роботи додатку з базою даних було використано технологію ADO.NET. Вона забезпечує можливість підключення та управління БД із програмного додатку. Сформовані дані можуть бути переглянуті за допомогою програми Microsoft OneNote.

2.7.3. Виклик та завантаження програми

Готовий мобільний додаток займає близько 35 Мб пам’яті мобільного девайсу. Розроблений програмний продукт складається з двох додатків: для клієнта і для співробітників (адміністратора та водія).

Для установки програми необхідно скопіювати в пам'ять пристрою (або пам'ять SD-карти) файл WaterWay.apk. Після цього, за допомогою файлового менеджера, необхідно запустити даний файл – почнеться процес установки. Коли установка завершиться, ярлик програми з'явиться в головному меню пристрою.

Перед запуском програми необхідно включити перевірити можливість доступу в Інтернет для завантаження мапи місцевості.

2.7.4. Опис інтерфейсу користувача

Розглянемо інтерфейс додатку для клієнта. При відкритті програми завантажується вікно авторизації, у якому розташовуються поля для вводу логіна та пароля (рис. 2.28).

Рис. 2.28. Вікно авторизації клієнта в системі

Якщо клієнт ще не має акаунта, то є можливість створити аккаунт у вікні реєстрації (рис. 2.29). Під час процесу реєстрації, клієнт повинен ввести своє ім’я, номер телефону, пароль та адресу. В залежності від місця проживання: квартира або будинок – є можливість обрати та зареєструвати відповідний тип адреси.

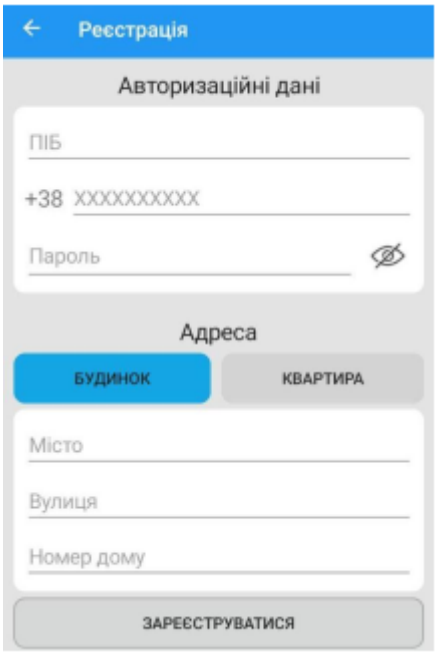

Рис. 2.29. Вікно реєстрації клієнта в системі

Після авторизації завантажується вікно з переліком доступних товарів (рис. 2.30.

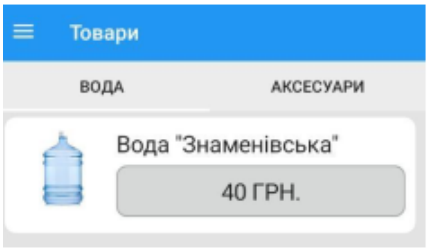

Рис. 2.30. Вікно для відображення переліку товарів

Щоб переглянути інформацію про товар та додати його до кошика, необхідно натиснути на кнопку з вартістю і відобразиться необхідне вікно (рис.

2.31.

Рис. 2.31. Вікно з інформацією про товар

Після вибору кількості товару та натиснувши на кнопку «В кошик», товар буде відображатися у вікні «Кошик» (рис. 2.32).

Рис. 2.32. Вікно «Кошик»

Для оформлення замовлення необхідно натиснути кнопку «Оформити замовлення» – завантажиться вікно з вибором дати доставки води та перевіркою адреси доставки (рис. 2.33).

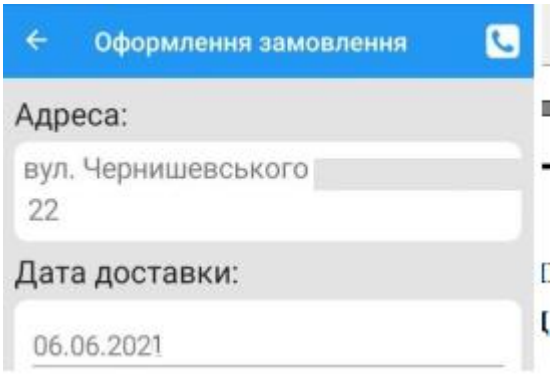

Рис. 2.33. Вікно для підтвердження замовлення клієнта

При виникненні додаткових запитань про замовлення або товар, є можливість зателефонувати адміністратору, натиснувши на кнопку телефону у верхньому правому кутку (рис. 2.34).

Рис. 2.34. Вікно для здійснення дзвінка до технічної підтримки

Після створення замовлення, воно зберігається в історію і відображається у вікні «Історія» (рис. 2.35). У даному вікні можна переглянути статус та повторити заказ.

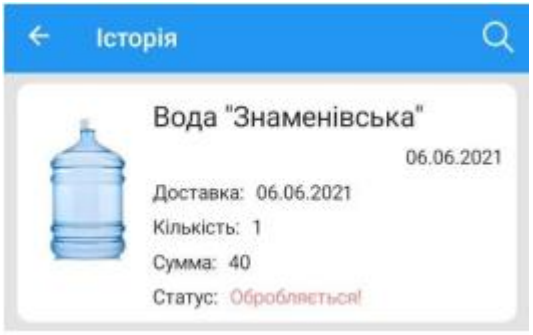

Рис. 2.35. Вікно відображення історії замовлень клієнта

Клієнт може переглянути та змінити персональну та авторизаційну інформацію про акаунт у вікні «Налаштування» (рис. 2.36).

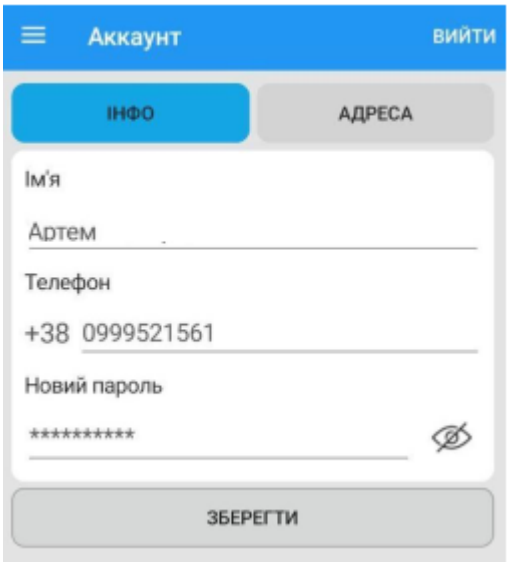

Рис. 2.36. Вікно «Налаштування»

При виникненні додаткових питань або непередбачуваних обставин, користувач з рівнем доступу «Клієнт» може написати до технічної підтримки, перейшовши у вікно «Підтримка» (рис. 2.37).

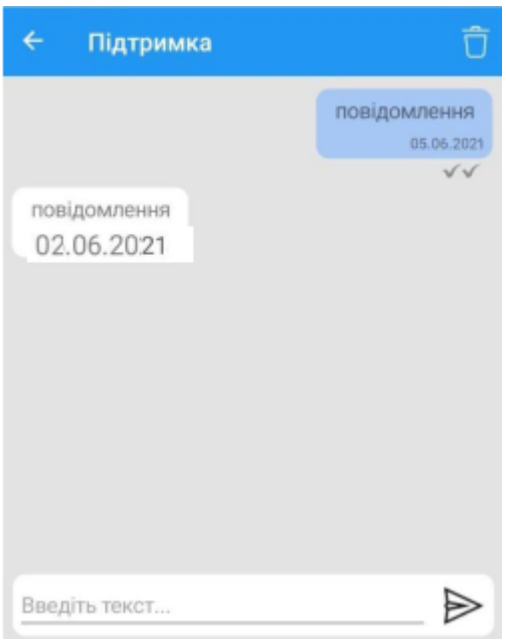

Рис. 2.37. Вікно «Підтримка» у додатку для клієнта

Розглянемо функціонал додатку для водія. Після створення замовлення клієнтом, воно відображається у головному вікні додатку для водія (рис. 2.38).

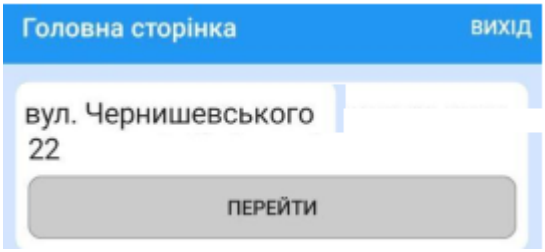

Рис. 2.38. Вікно відображення замовлень у додатку для водія

Для виконання замовлення, водій повинен обрати запис із переліку та перейти у вікно детальної інформації (рис. 2.39).

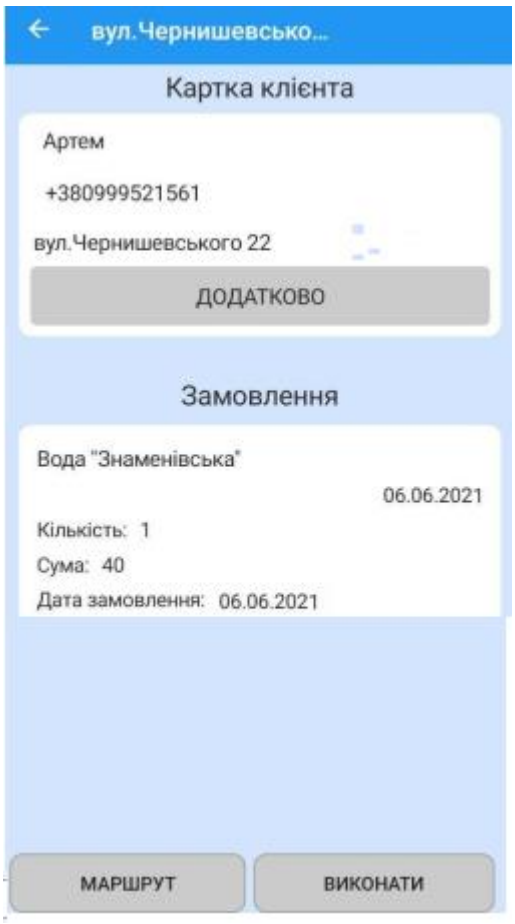

Рис. 2.39. Вікно з детальною інформацією про замовлення у додатку для водія

Натиснувши на номер телефону відкриється вікно для створення дзвінків. Кнопка «Додатково» завантажує детальну інформацію про користувача та можливість додавати нотації (рис. 2.40).

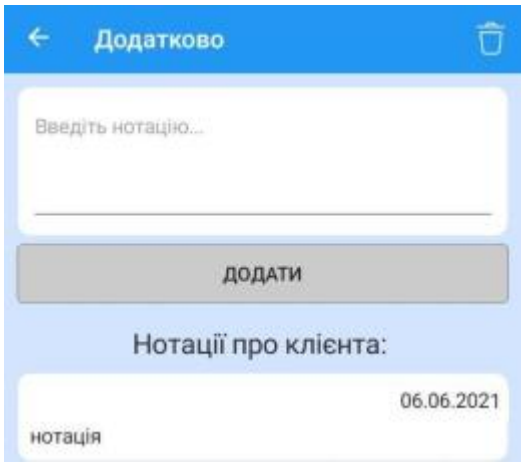

Рис. 2.40. Вікно з детальною інформацією про клієнта у додатку для водія

Для прокладання маршруту до заданої адреси замовником, необхідно у вікні опису замовлення натиснути кнопку «Маршрут», і додаток завантажить карту з пошуком найкоротшого шляху (рис. 2.41).

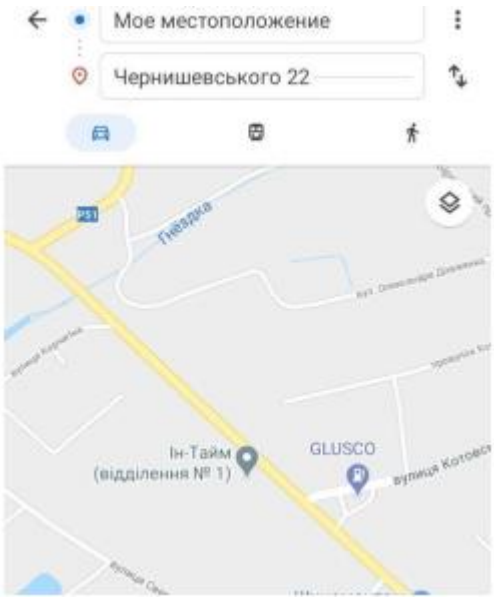

Рис. 2.41. Вікно пошуку найкоротшого маршруту у додатку для водія

Розглянемо функціонал додатку для адміністратора. Пройшовши авторизацію з рівнем доступу адміністратор, користувач потрапляє у вікно редагування та додавання товарів до бази даних (рис. 2.42).

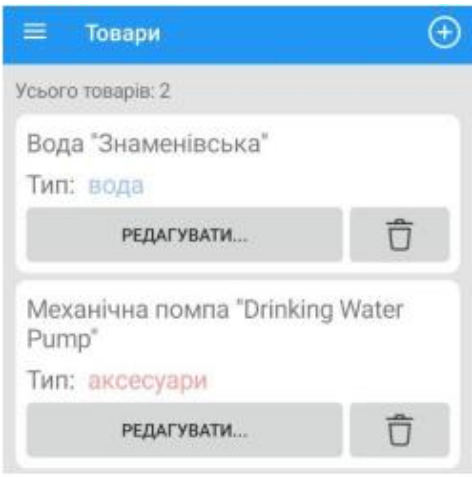

Рис. 2.42. Вікно редагування та додавання нових товарів у додатку для адміністратора

Інформація про робітників фірми відображається у вікні «База робітників» (рис. 2.43). Також є можливість додавати або видаляти дані про робітника з системи.

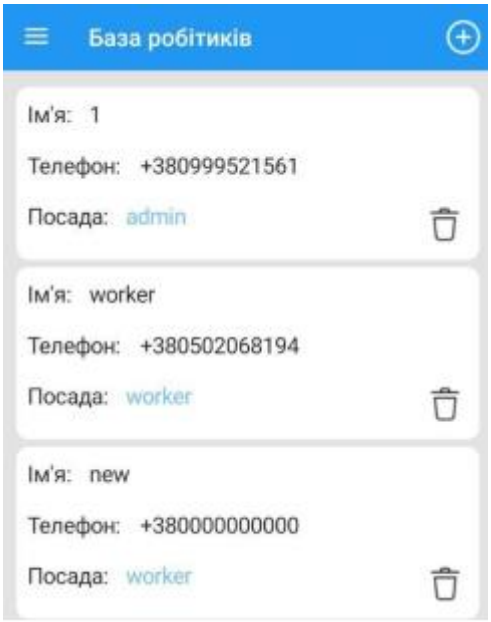

Рис. 2.43. Вікно «База робітників» у додатку для адміністратора

Змінити інформацію про фірму адміністратор зможе відкривши вікно «Про нас» (рис. 2.44).

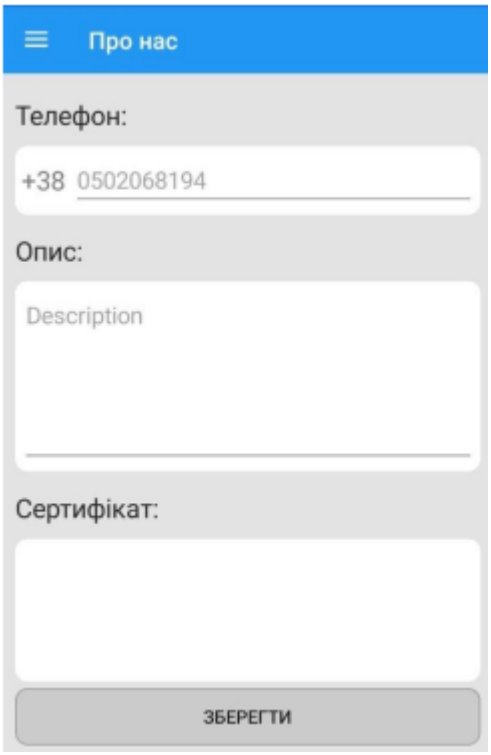

Рис. 2.44. Вікно редагування інформації про фірму у додатку для адміністратора

Перелік клієнтів, інформація про них, та можливість надавати технічну підтримку доступно у вікні «База клієнтів» (рис. 2.45).

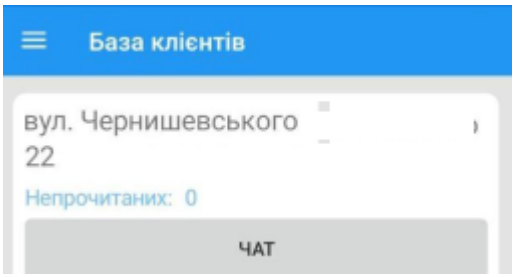

Рис. 2.45. Вікно «База клієнтів» у додатку для адміністратора

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 1000;
- коефіцієнт складності програми – 2;
- коефіцієнт корекції програми в ході її розробки – 0,08;
- годинна заробітна плата програміста, грн / год – 40.

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:

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

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50),

ти - витрати праці на дослідження алгоритму рішення задачі,

та - витрати праці на розробку блок-схеми алгоритму,

тп - витрати праці на програмування по готовій блок-схемі,

тотл - витрати праці на налагодження програми на ЕОМ,

тд - витрати праці на підготовку документації.

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

- умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів,

С - коефіцієнт складності програми,

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1000 \cdot 2 \cdot (1 + 0{,}08) = 2808 \text{ людино-годин.}$$

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

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \qquad (3.3)$$

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

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2808 \cdot 1{,}2}{80 \cdot 0{,}8} = 52 \text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{2808}{20 \cdot 0{,}8} = 175 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{2808}{25 \cdot 0{,}8} = 140 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{2808}{4 \cdot 0{,}8} = 877 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

$$t_{отл} = 877 * 1{,}2 = 1053$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.8}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.9}$$

$$t_{др} = \frac{2808}{15 \cdot 0{,}8} = 234 \text{ людино-годин,}$$

tдо – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.10)$$

$$tdo = 0{,}75 \cdot 234 = 175$$

$$t_{д} = 234 + 175 = 409 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 52 + 175 + 140 + 1053 + 409 = 1861 \text{ людино-годин.}$$

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:

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

Заробітна плата виконавців визначається за формулою:

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

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1861 \cdot 30 = 43755 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

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

де tотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 5 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 810 \times 5 = 4050 \text{ грн.}$$

$$Кпо = 43755 + 4050 = 47805 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.13)$$

де Bk - число виконавців,

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

$$T = \frac{2160}{1 \cdot 176} = 8{,}3 \text{ міс.}$$

Висновок: були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 8,3 місяця, трудомісткість розробки ПЗ – 1861 людино-годин, а витрати на її створення програмного забезпечення – 47805 грн.

ВИСНОВКИ

Метою кваліфікаційної роботи є створення програмного забезпечення для автоматизації процесів бізнесу постачання води, яке дозволить здійснювати зручне замовлення води в електронній формі на смартфоні, а також обробляти отримане замовлення та спрощувати доставку товару замовнику.

Інформаційна система складається з двох окремих додатків: для клієнта та для співробітників фірми.

Додаток для клієнта виконує наступні функції:

– реєстрація нових користувачів у системі;

– оновлення та редагування персональних даних клієнта;

– реєстрація клієнтом нових замовлень з вибором дати і кількості продукції;

– створення клієнтом та повторення збереженого замовлення;

– доступ до технічної підтримки;

– перегляд історії попередніх замовлень.

Дані функції забезпечують наступні форми:

– товари: містить відомості про наявні товари;

– замовлення: дає смогу додати в кошик обраний товар вказаної кількості;

– історія: містить історію замовлень користувача;

– кошик: дає змогу користувачеві оформити створене замовлення;

– підтримка: дає змогу отримувати технічну підтримку;

– налаштування: містить функціонал для редагування персональ них даних та налаштування додатку.

Додаток для співробітників фірми виконує наступні функції:

– перегляд списку даних про зареєстрованих клієнтів та співробітників;

– перегляд та додавання нових товарів та акцій;

– перегляд списку замовлень клієнтів;

– побудова оптимального маршруту для водія.

Даний функціонал забезпечується формами перегляду бази клієнтів та замовлень, додавання та редагування інформації про товари та створення оптимального маршруту для водія.

В даному програмному продукті реалізована технічна підтримка користувачів в реальному часі, за допомогою якої користувачі можуть швидко отримувати відповіді на непередбачувані проблеми та запитання.

Для розробки програмного продукту було обране середовище програмування Microsoft Visual Studio, фреймворк Xamarin та мова програмування C#. Для створення і реалізації бази даних було обрано СКБД PhpMyAdmin. Для реалізації роботи додатку з базою даних було використано технологію ADO.NET. Вона забезпечує можливість підключення та управління БД із програмного додатку. Сформовані дані можуть бути переглянуті за допомогою програми Microsoft OneNote.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 8,3 місяця, трудомісткість розробки ПЗ – 1861 людино-годин, а витрати на її створення програмного забезпечення - 47805 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мова програмування C#. URL: https://programm.top/uk/c–sharp/tutorial/introduction/. дата звернення: 24.05.2021.
2. Інтернет Речей або Internet of Things URL: https://deps.ua/ua/katalog/iot.html. дата звернення: 24.05.2021.
3. Xamarin.Forms TabbedPage URL: https://docs.microsoft.com/ru–ru/xamarin/xamarinforms/ app–fundamentals/navigation/tabbed–page. дата звернення: 24.05.2021.
4. СКБД phpMyAdmin URL: https://docs.phpmyadmin.net /uk/latest/intro.html. дата звернення: 24.05.2021.
5. Що таке SQL? URL: https://a–yak.com/shho–take–sql/.
6. Microsoft Office URL:https://uk.wikipedia.org/wiki/Microsoft_Office.
7. USE–case діаграма URL:studopedia.ru/19_284009_diagrama–variantivvikoristannya– USE–case–diagram.html. дата звернення: 24.05.2021.
8. Діаграма послідовності – Sequence Diagram URL: http://www.dut.edu.ua/ua/news–1–626– 7897–zastosuvannya–uml–chastina–2–diagrama–poslidovnosti-sequen-ce –diagram.
9. Поняття бази даних URL:https://studfile.net/preview/5454020/page:19/.
10. 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).
11. 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: 24.05.2021.
12. 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: 15.01.2018.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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).

19. Стиллмен Э., Грин Дж. – «C# для чайников» – Л.: Питер, 2012. – 680 с.

20. Царев, В.А. Проектирование, анализ и программная реализация структур данных и алгоритмов: Учебное пособие / В.А. Царев, А.Ф. Дробанов. – Череповец., 2007. – 140 с.

ДОДАТОК А

КОД ПРОГРАМИ

```
CREATE TABLE `aboutUs` (
`id` int(11) NOT NULL,
`certficateImage` longblob NOT NULL,
`aboutUs` varchar(1000) NOT NULL,
`phone` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `adress` (
`id` int(11) NOT NULL,
`idClient` int(11) NOT NULL,
`city` varchar(50) NOT NULL,
`street` varchar(50) NOT NULL,
`houseNumber` varchar(50) NOT NULL,
`flatNumber` int(11) DEFAULT '0',
`floor` int(11) DEFAULT '0',
`code` int(11) DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `client` (
`id` int(11) NOT NULL,
`name` varchar(50) NOT NULL,
`phone` varchar(11) NOT NULL,
`password` varchar(50) NOT NULL,
`bonuses` int(11) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `notation` (
`id` int(11) NOT NULL,
`idClient` int(11) NOT NULL,
`notation` varchar(500) NOT NULL,
`date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `notification` (
`id` int(11) NOT NULL,
`idClient` int(11) NOT NULL,
`sender` varchar(50) NOT NULL,
`text` varchar(100) NOT NULL,
`date` date NOT NULL,
`seen` tinyint(1) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `orders` (
`id` int(11) NOT NULL,
`idClient` int(11) NOT NULL,
`idProduct` int(11) NOT NULL,
`amount` int(11) NOT NULL,
`cost` float NOT NULL,
`date` date NOT NULL,
`dateOfCreation` date NOT NULL,
`isDone` tinyint(1) NOT NULL
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `product` (
`id` int(11) NOT NULL,
`title` varchar(50) NOT NULL,
`image` longblob NOT NULL,
`description` varchar(500) NOT NULL,
`cost` float NOT NULL,
`discount` float DEFAULT NULL,
`type` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `discount` (
`id` int(11) NOT NULL,
`title` varchar(50) NOT NULL,
`image` blob NOT NULL,
`description` varchar(500) NOT NULL,
`dateFrom` date NOT NULL,
`dateTo` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `worker` (
`id` int(11) NOT NULL,
`name` varchar(50) NOT NULL,
`phone` varchar(11) NOT NULL,
`password` varchar(50) NOT NULL,
`role` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Файл «AccountPage.xaml.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class AccountPage : ContentPage
{
Master masterPage;
public AccountPage(Master _masterPage)
{
masterPage = _masterPage;
InitializeComponent();
this.BindingContext = this;
}
private async void AccountToolBarExit_Clicked(object sender, EventArgs e)
{
masterPage.Navigation.InsertPageBefore(new Login(), masterPage); await
masterPage.Navigation.PopToRootAsync();}
```

Файл «ChooseCity.xaml.cs»

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class ChooseCity : ContentPage
{
public string[] Cities { get; set; }
public ChooseCity()
{
InitializeComponent();
Cities = new string[] {"Дніпро","с.Чумаки","с.Чаплинка","Новомосковскь","Павлоград"};
ListViewCities.HeightRequest = 60 * Cities.Length;
this.BindingContext = this;
```

```
}
private void ListViewCities_ItemTapped(object sender, ItemTappedEventArgs e)
{
ListViewCities.SelectedItem = null;
Navigation.InsertPageBefore(new Master(), this);
Navigation.PopAsync();
}
private void PhoneIcon_Clicked(object sender, EventArgs e)
{
DisplayAlert("", "Hi", "OK");
}
}
}
```

Файл «HistoryCell.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.PlatformConfiguration;
using Xamarin.Forms.PlatformConfiguration.AndroidSpecific;
using Xamarin.Forms.Xaml;
namespace WaterWay
{
public class HistoryCell : Xamarin.Forms.ViewCell
{
Label titleLabel, amountLabel, costLabel, dateLabel;
Image image;
public HistoryCell()
{
image = new Image();
titleLabel = new Label { FontSize = 18 };
amountLabel = new Label();
costLabel = new Label();
dateLabel = new Label();
StackLayout mainCell = new StackLayout();
StackLayout titleLayout = new StackLayout();
titleLayout.Children.Add(titleLabel);
titleLayout.Children.Add(amountLabel);
titleLayout.Children.Add(costLabel);
StackLayout dateLayout = new StackLayout(); dateLayout.HorizontalOptions = LayoutOptions.EndAndExpand;
dateLayout.Children.Add(dateLabel);
mainCell.Orientation = StackOrientation.Horizontal;
mainCell.Children.Add(image);
mainCell.Children.Add(titleLayout);
mainCell.Children.Add(dateLayout);
View = mainCell;
}
public static readonly BindableProperty TitleProperty = BindableProperty.Create("Title", typeof(string),
typeof(HistoryCell), "");
public static readonly BindableProperty ImagePathProperty = BindableProperty.Create("ImagePath",
typeof(ImageSource), typeof(HistoryCell), null);
public static readonly BindableProperty ImageWidthProperty = BindableProperty.Create("ImageWidth", typeof(int),
typeof(HistoryCell), 100);
public static readonly BindableProperty ImageHeightProperty = BindableProperty.Create("ImageHeight", typeof(int),
typeof(HistoryCell), 100);
public static readonly BindableProperty AmountProperty = BindableProperty.Create("Amount", typeof(string),
typeof(HistoryCell), "");
public static readonly BindableProperty CostProperty = BindableProperty.Create("Cost", typeof(string),
typeof(HistoryCell), "");
```

```
public static readonly BindableProperty DateProperty = BindableProperty.Create("Date", typeof(string), typeof(HistoryCell), "");
public string Title
{
get { return (string)GetValue(TitleProperty); }
set { SetValue(TitleProperty, value); }
}
public int ImageWidth
{
get { return (int)GetValue(ImageWidthProperty); }
set { SetValue(ImageWidthProperty, value); }
}
public int ImageHeight
{
get { return (int)GetValue(ImageHeightProperty); }
set { SetValue(ImageHeightProperty, value); }
}
public ImageSource ImagePath
{
get { return (ImageSource)GetValue(ImagePathProperty); }
set { SetValue(ImagePathProperty, value); }
}
public string Amount
{
get { return (string)GetValue(AmountProperty); }
set { SetValue(AmountProperty, value); }
}
public string Cost
{
get { return (string)GetValue(CostProperty); }
set { SetValue(CostProperty, value); }
}
public string Date
{
get { return (string)GetValue(DateProperty); }
set { SetValue(DateProperty, value); }
}
protected override void OnBindingContextChanged()
{
base.OnBindingContextChanged();
if (BindingContext != null)
{
titleLabel.Text = Title;
amountLabel.Text = Amount;
costLabel.Text = Cost;
dateLabel.Text = Date;
image.Source = ImagePath;
image.WidthRequest = ImageWidth;
image.HeightRequest = ImageHeight;
}
}
}
}
```

Файл «HistoryPage.xaml.cs»

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
```

```
using Xamarin.Forms.Xaml;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class HistoryPage : ContentPage
{
public ObservableCollection<History> historyList { get; set; }
public ObservableCollection<History> showHistoryList
{
get
{
return _showHistoryList;
}
set
{
if (_showHistoryList != value)
{
_showHistoryList = value;
OnPropertyChanged("showHistoryList");
}
}
}
public ObservableCollection<History> _showHistoryList;
public HistoryPage()
{
InitializeComponent();
historyList = new ObservableCollection<History>
{
new History{title="Voda",amount="1",cost="40",imageSource="LoginLogo.png",date="15.04.2020"}, new History{title="Voda",amount="3",cost="120",imageSource="LoginLogo.png",date="13.04.2020"}, new History{title="Voda",amount="2",cost="80",imageSource="LoginLogo.png",date="12.04.2020"},
};
showHistoryList = new ObservableCollection<History>(historyList); HistoryListView.HeightRequest = 80 * showHistoryList.Count; this.BindingContext = this;
}
private void HistoryList_CollectionChanged(object sender, System.Collections.Specialized.NotifyCollectionChangedEventArgs e)
{
if (historyList.Count == 0 && showHistoryList.Count == 0)
HistoryEmptyLabel.IsVisible = true;
else
HistoryEmptyLabel.IsVisible = false;
}
private void HistoryButtonClear_Clicked(object sender, EventArgs e)
{
showHistoryList.Clear();
historyList.Clear();
}
private void HistorySearchEntry_TextChanged(object sender, TextChangedEventArgs e)
{
if (HistorySearchEntry.Text == string.Empty)
{
showHistoryList.Clear();
showHistoryList = new ObservableCollection<History>(historyList);
}
else
{
showHistoryList = new ObservableCollection<History>(historyList.Where(item => item.title.ToLower().Contains(HistorySearchEntry.Text.ToLower()) ||
item.amount.ToLower().Contains(HistorySearchEntry.Text.ToLower()) ||
item.cost.ToLower().Contains(HistorySearchEntry.Text.ToLower()) ||
item.date.ToLower().Contains(HistorySearchEntry.Text.ToLower())));
```

```
}
}
private void HistorySearchButton_Clicked(object sender, EventArgs e)
{
if (HistorySearchEntry.IsVisible == true)
{
HistorySearchEntry.IsVisible = false;
HistorySearchEntry.Text = "";
}
else
HistorySearchEntry.IsVisible = true;
}
}
}
```

Файл «Login.xaml.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using MySql.Data.MySqlClient;
using System.Data;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class Login : ContentPage
{
public Login()
{
InitializeComponent();
LoginPhone.MaxLength = 10;
LoginPassword.MaxLength = 30;
LoginButtonVisiblePassword.Clicked += ButtonVisiblePassword_Clicked;
LoginButtonRegistration.Clicked += ButtonRegistration_Clicked;
}
private void ButtonVisiblePassword_Clicked(object sender, EventArgs e)
{
if (LoginPassword.IsPassword == true)
LoginPassword.IsPassword = false;
else
LoginPassword.IsPassword = true;
}
private async void ButtonLogin_Clicked(object sender, EventArgs e)
{
if (LoginPhone.Text == null)
await DisplayAlert("", "Введіть номер телефону", "ok");
else if (LoginPassword.Text == null)
await DisplayAlert("", "Введіть пароль", "ok");
else
{
Navigation.InsertPageBefore(new Master(), this);
await Navigation.PopAsync();
}
}
private async void ButtonRegistration_Clicked(object sender, EventArgs e)
{
await Navigation.PushAsync(new Registration());
LoginPassword.Text = null;
}
```

```
}
}
```

Файл «Registration.xaml.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using MySql.Data.MySqlClient;
using System.Data;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class Registration : ContentPage
{
public Registration()
{
InitializeComponent();
RegistrationName.Text = null;
RegistrationSurname.Text = null;
RegistrationPhone.Text = null;
RegistrationPassword.Text = null;
RegistrationName.MaxLength = 30;
RegistrationSurname.MaxLength = 30;
RegistrationPhone.MaxLength = 10;
RegistrationPassword.MaxLength = 30;
}
private void ButtonVisiblePassword_Clicked(Object sender, EventArgs e)
{
if (RegistrationPassword.IsPassword == true)
RegistrationPassword.IsPassword = false;
else
RegistrationPassword.IsPassword = true;
}
private async void ButtonRegistration_Clicked(Object sender, EventArgs e)
{
if (RegistrationName.Text == null)
await DisplayAlert("", "Введіть ім'я", "ok");
else if (RegistrationSurname.Text == null)
await DisplayAlert("", "Введіть прізвище", "ok");
else if (RegistrationPhone.Text == null)
await DisplayAlert("", "Введіть номер телефону", "ok");
else if (RegistrationPassword.Text == null)
await DisplayAlert("", "Введіть пароль", "ok");
else
{
Navigation.InsertPageBefore(new RegistrationSecond(), this);
await Navigation.PopAsync();
}
}
private async void ButtonLogin_Clicked(Object sender, EventArgs e)
{
await Navigation.PopAsync();
}
}
}
```

Файл «MyOrdersPage.xaml.cs»

```
using System;
```

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class MyOrdersPage : ContentPage
{
public ObservableCollection<History> showHistoryList
{
get
{
return _showHistoryList;
}
set
{
if (_showHistoryList != value)
{
_showHistoryList = value;
OnPropertyChanged("showHistoryList");
}
}
}
public ObservableCollection<History> _showHistoryList;
public MyOrdersPage()
{
InitializeComponent();

showHistoryList = new ObservableCollection<History>
{
new
History{title="Voda",amount="1",cost="40",imageSource="LoginLogo.png",date=DateTime.Today.ToShortDateStrin
g()}
};
HistoryListView.HeightRequest = 80 * showHistoryList.Count;
this.BindingContext = this;
}
private async void MyOrdersButtonHistory_Clicked(object sender, EventArgs e)
{
bool result = await DisplayAlert("Корзина", "Очистити корзину?", "Так", "Ні"); if(result)
{
showHistoryList.Clear();
HistoryEmptyLabel.IsVisible = true;
}
}
private void HistoryListView_ItemSelected(object sender, SelectedItemChangedEventArgs e)
{
HistoryListView.SelectedItem = null;
}
}
}
```

Файл «OrderaAccept.xaml.cs»

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class OrderAccept : ContentPage
{
public ObservableCollection<History> showHistoryList
{
get
{
return _showHistoryList;
}
set
{
if (_showHistoryList != value)
{
_showHistoryList = value;
OnPropertyChanged("showHistoryList");
}
}
}
public ObservableCollection<History> _showHistoryList;
int cost = 0;
public OrderAccept(string _title,int _cost)
{
InitializeComponent();
cost = _cost;
OrderAceptTitle.Text = _title;
this.BindingContext = this;
}
private async void OrderAcceptButtonAccept_Clicked(object sender, EventArgs e)
{
await DisplayAlert("", "Підтверджено!", "OK");
await Navigation.PopAsync();
}
private void OrderAcceptStepper_ValueChanged(object sender, ValueChangedEventArgs e)
{
OrderAcceptCost.Text = string.Format("Загальна вартість: {0} грн.", e.NewValue*cost);
OrderAcceptLabelAmount.Text = string.Format("Кількість: {0}", e.NewValue);
}
private void OrderAcceptStepper2_ValueChanged(object sender, ValueChangedEventArgs e)
{
OrderAcceptLabelBack.Text = string.Format("Тара: {0}", e.NewValue);
}
}
}
```

Файл «ProductCell.cs»

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;
namespace WaterWay
{
public class ProductCell : Xamarin.Forms.ViewCell
{
Label title, cost;
Image image;
public ProductCell()
{
```

```
image = new Image();
title = new Label { FontSize = 18 };
cost = new Label() { VerticalOptions=LayoutOptions.EndAndExpand};
StackLayout mainCell = new StackLayout();
StackLayout titleLayout = new StackLayout();
titleLayout.Children.Add(title);
titleLayout.Children.Add(cost);
mainCell.Orientation = StackOrientation.Horizontal;
mainCell.Children.Add(image);
mainCell.Children.Add(titleLayout);
View = mainCell;
}
public static readonly BindableProperty TitleProperty = BindableProperty.Create("Title", typeof(string), typeof(ProductCell), "");
public static readonly BindableProperty ImagePathProperty = BindableProperty.Create("ImagePath", typeof(ImageSource), typeof(ProductCell), null);
public static readonly BindableProperty ImageWidthProperty = BindableProperty.Create("ImageWidth", typeof(int), typeof(ProductCell), 100);
public static readonly BindableProperty ImageHeightProperty = BindableProperty.Create("ImageHeight", typeof(int), typeof(ProductCell), 100);
public static readonly BindableProperty CostProperty =
BindableProperty.Create("Cost", typeof(int), typeof(ProductCell), 100); public string Title
{
get { return (string)GetValue(TitleProperty); }
set { SetValue(TitleProperty, value); }
}
public int ImageWidth
{
get { return (int)GetValue(ImageWidthProperty); }
set { SetValue(ImageWidthProperty, value); }
}
public int ImageHeight
{
get { return (int)GetValue(ImageHeightProperty); }
set { SetValue(ImageHeightProperty, value); }
}
public ImageSource ImagePath
{
get { return (ImageSource)GetValue(ImagePathProperty); }
set { SetValue(ImagePathProperty, value); }
}
public string Cost
{
get { return GetValue(CostProperty).ToString(); }
set { SetValue(CostProperty, value); }
}
protected override void OnBindingContextChanged()
{
base.OnBindingContextChanged();
if (BindingContext != null)
{
title.Text = Title;
cost.Text = Cost;
image.Source = ImagePath;
image.WidthRequest = ImageWidth;
image.HeightRequest = ImageHeight;
}
}
}
}
```

Файл «ProductPage.cs»

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.PlatformConfiguration;
using Xamarin.Forms.PlatformConfiguration.AndroidSpecific;
using Xamarin.Forms.Xaml;
namespace WaterWay
{
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class ProductPage : Xamarin.Forms.TabbedPage
{
public List<Product> productList { get; set; }
public List<Product> accessoriesList { get; set; }
public List<Product> ProductList
{
get { return productList; }
set
{
if(productList!=value)
{
productList = value;
OnPropertyChanged("ProductList");
}
}
}
public ProductPage()
{
InitializeComponent();
ProductList = new List<Product>
{
new Product{title="Вода",cost=40,image="LoginLogo.png"}
};
accessoriesList = new List<Product>
{
new Product{title="Помпа",cost=130,image="SettingsPage.png"} };
ProductWaterListView.HeightRequest = 80 * ProductList.Count; ProductAccessoriesListView.HeightRequest = 80 *
ProductList.Count; this.BindingContext = this;
}
private void ProductAccessoriesListView_ItemTapped(object sender, ItemTappedEventArgs e)
{
ProductAccessoriesListView.SelectedItem = null;
Navigation.PushAsync(new OrderAccept("Помпа",130));
}
private void ProductWaterListView_ItemTapped(object sender, ItemTappedEventArgs e)
{
ProductWaterListView.SelectedItem = null;
Navigation.PushAsync(new OrderAccept("Вода",40));
}
}
}
```

Файл «MVM.cs»

```
using MySql.Data.MySqlClient;
using Plugin.Messaging;
using System;
using System.Collections.ObjectModel;
using System.Data;
using System.Linq;
```

```
using System.Threading.Tasks;
using WaterWay.Models;
using WaterWay.Views;
using Xamarin.Forms;
namespace WaterWay.ViewModels
{
public class MVM : INPC
{
MVM mainViewModel;
public MainPage mainPage;
public MainPageDetail mainPageDetail;
public int idClient;
//MainPageMaster
int masterPageAllMessages;
int masterPageUnreadMessage;
public int MasterPageUnreadMessage
{
get { return masterPageUnreadMessage; }
set
{
masterPageUnreadMessage = masterPageAllMessages– value;
OnPropertyChanged();
if (masterPageUnreadMessage != 0)
{
MasterPageUnreadMessageIsVisible = true;
}
}
}
bool masterPageUnreadMessageIsVisible;
public bool MasterPageUnreadMessageIsVisible
{
get { return masterPageUnreadMessageIsVisible; }
set
{
if (masterPageUnreadMessageIsVisible != value)
{
masterPageUnreadMessageIsVisible = value;
OnPropertyChanged();
}
}
}
ObservableCollection<ModelMainPageMaster> listViewMainPageMaster; public
ObservableCollection<ModelMainPageMaster> ListViewMainPageMaster
{
get { return listViewMainPageMaster; }
set
{
if(listViewMainPageMaster!=value)
{
listViewMainPageMaster = value;
OnPropertyChanged();
}
}
}
public void SetMainPageMaster ()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
```

```
{
connection.Open();
MySqlCommand command = new MySqlCommand("SELECT count(notification.id) FROM notification" +
"  WHERE idClient=@idClient AND sender='admin'", connection); command.Parameters.AddWithValue("@idClient",
idClient); MySqlDataReader reader = command.ExecuteReader();
if (reader.Read())
masterPageAllMessages = Int32.Parse(reader[0].ToString()); reader.Close();
command.CommandText = "SELECT count(notification.id) FROM notification" +
"  WHERE idClient=@idClient AND seen=true AND sender='admin'";
reader = command.ExecuteReader();
if (reader.Read())
MasterPageUnreadMessage = Int32.Parse(reader[0].ToString());
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
ListViewMainPageMaster = new ObservableCollection<ModelMainPageMaster>
{
new ModelMainPageMaster{Logo="ProductPage.png",Title="Головна сторінка"}, new
ModelMainPageMaster{Logo="DiscountPage.png",Title="Акції"},
new ModelMainPageMaster{Logo="BasketPage.png",Title="Кошик"},
new ModelMainPageMaster{Logo="HistoryPage.png",Title="Історія замовлень"}, new
ModelMainPageMaster{Logo="AccountPage.png",Title="Налаштування"}, new
ModelMainPageMaster{Logo="NotificationPage.png",Title="Підтримка",UnreadMessages=MasterPageUnreadMessa
ge,UnreadMessagesIsVisi ble=M asterPageUnreadMessageIsVisible},
new ModelMainPageMaster{Logo="AboutUsPage.png",Title="Про нас"}
};
}
//PoductPage
public ModelProductPage modelProductPage; ObservableCollection<ModelProductPage>
listViewProductPageFirstTabPage; public ObservableCollection<ModelProductPage>
ListViewProductPageFirstTabPage {
get { return listViewProductPageFirstTabPage; }
set
{
if (listViewProductPageFirstTabPage != value)
listViewProductPageFirstTabPage = value;
OnPropertyChanged();
}
}
ObservableCollection<ModelProductPage> listViewProductPageSecondTabPage; public
ObservableCollection<ModelProductPage> ListViewProductPageSecondTabPage
{
get { return listViewProductPageSecondTabPage; }
set
{
if (listViewProductPageSecondTabPage != value)
listViewProductPageSecondTabPage = value;
OnPropertyChanged();
}
}
public void SetProductPage()
{
ListViewProductPageFirstTabPage = new ObservableCollection<ModelProductPage>();
ListViewProductPageSecondTabPage = new ObservableCollection<ModelProductPage>(); string connectionString =
```

```
@"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("SELECT * FROM product WHERE type='water'", connection);
MySqlDataReader reader = command.ExecuteReader();
while (reader.Read())
ListViewProductPageFirstTabPage.Add(new ModelProductPage(mainViewModel)
{
Id = reader.GetInt32(0),
Title = reader.GetString(1),
Image= reader.GetString(2),
FloatCost = reader.GetFloat(4),
Discount = reader.GetFloat(5)
});
reader.Close();
command.CommandText = "SELECT * FROM product WHERE type = 'accessory'"; reader =
command.ExecuteReader();
while (reader.Read())
ListViewProductPageSecondTabPage.Add(new ModelProductPage(mainViewModel)
{
Id = reader.GetInt32(0),
Title = reader.GetString(1),
Image = reader.GetString(2),
FloatCost = reader.GetFloat(4),
Discount = reader.GetFloat(5)
});
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
connection.Close();
}
//AboutProductPage
string aboutProductPagePhone;
public string AboutProductPagePhone
{
get { return aboutProductPagePhone; }
set
{
if(aboutProductPagePhone!=value)
{
aboutProductPagePhone = value;
OnPropertyChanged();
}
}
}
int IdAboutProductPage;
string titleAboutProductPage;
public string TitleAboutProductPage
{
```

```
get { return titleAboutProductPage; }
set
{
if (titleAboutProductPage != value)
titleAboutProductPage = value;
OnPropertyChanged();
}
}
string iconAboutProductPage;
public string IconAboutProductPage
{
get { return iconAboutProductPage; }
set
{
if (iconAboutProductPage != value)
iconAboutProductPage = value;
OnPropertyChanged();
}
}
string imageAboutProductPage;
public string ImageAboutProductPage
{
get { return imageAboutProductPage; }
set
{
if (imageAboutProductPage != value)
imageAboutProductPage = value;
OnPropertyChanged();
}
}
float costAboutProductPage;
public float CostAboutProductPage
{
get { return costAboutProductPage; }
set
{
if (costAboutProductPage != value)
costAboutProductPage = value;
OnPropertyChanged();
}
}
int amountAboutProductPage;
public int AmountAboutProductPage
{
get { return amountAboutProductPage; }
set
{
if (amountAboutProductPage != value)
amountAboutProductPage = value;
OnPropertyChanged();
}
}
float totalCostAboutProductPage;
public float TotalCostAboutProductPage
{
get { return totalCostAboutProductPage; }
set
{
if (totalCostAboutProductPage != value)
totalCostAboutProductPage = value;
OnPropertyChanged();
}
```

```
}
string descriptionAboutProductPage;
public string DescriptionAboutProductPage
{
get { return descriptionAboutProductPage; }
set
{
if (descriptionAboutProductPage != value)
descriptionAboutProductPage = value;
OnPropertyChanged();
}
}
public Command LaunchAboutProductPageMinusButton_Clicked { get; }
public Command LaunchAboutProductPagePlusButton_Clicked { get; }
public Command LaunchAboutProductPageAddInBusketButton_Clicked { get; }
public Command LaunchAboutProductPageCall { get; }
void AboutProductPageSetPhone()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
MySqlCommand command = new MySqlCommand(" SELECT phone FROM aboutUs WHERE id = 1", connection);
connection.Open();
var reader = command.ExecuteReader();
if (reader.Read())
{
AboutProductPagePhone = "+38"+reader[0].ToString();
}
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
void AboutProductPageCall()
{
var phoneDialer = CrossMessaging.Current.PhoneDialer;
if (phoneDialer.CanMakePhoneCall)
phoneDialer.MakePhoneCall(AboutProductPagePhone);
}
public void SetAboutProductPage(int id)
{
IconAboutProductPage = "ProductPage.png";
AmountAboutProductPage = 0;
IdAboutProductPage = id;
TotalCostAboutProductPage = 0;
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
```

```
{
connection.Open();
MySqlCommand command = new MySqlCommand("SELECT * FROM product WHERE id=@id", connection);
command.Parameters.AddWithValue("@id", id);
MySqlDataReader reader = command.ExecuteReader();
while (reader.Read())
{
TitleAboutProductPage = reader[1].ToString();
ImageAboutProductPage = reader[2].ToString();
DescriptionAboutProductPage = reader[3].ToString();
CostAboutProductPage = (float)reader[4];
}
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
void AboutProductPageMinusButton_Clicked()
{
if (AmountAboutProductPage > 0)
AmountAboutProductPage—;
TotalCostAboutProductPage = (AmountAboutProductPage * CostAboutProductPage);
}
void AboutProductPagePlusButton_Clicked()
{
AmountAboutProductPage++;
TotalCostAboutProductPage = (AmountAboutProductPage * CostAboutProductPage);
}
void AboutProductPageAddInBusketButton_Clicked()
{
try
{
if (ListViewBasketPage.Where(x => x.Id == IdAboutProductPage).Count() == 0)
{
if (AmountAboutProductPage > 0)
{
ListViewBasketPage.Add(new ModelBasketPage(mainViewModel) { Id = IdAboutProductPage,Image=imageAboutProductPage, Title = TitleAboutProductPage, FloatCost = CostAboutProductPage, Amount = AmountAboutProductPage, TotalCost = AmountAboutProductPage * CostAboutProductPage });
TotalCostBasketPage += AmountAboutProductPage * CostAboutProductPage; basketPage.IconImageSource = "BasketPagePlus.png";
modelProductPage.Cost = modelProductPage.FloatCost + " грн. x " + AmountAboutProductPage; modelProductPage.CostButtonColor = "DeepPink";
modelProductPage.InBusket = true;
App.Current.MainPage.Navigation.PopAsync();
}
else
App.Current.MainPage.DisplayAlert("", "Введіть кількість більше нуля!", "OK");
}
else
{
App.Current.MainPage.Navigation.PopAsync();
mainPageDetail.SelectedItem = null;
mainPageDetail.SelectedItem = mainPageDetail.Children[1];
```

```
App.Current.MainPage.DisplayAlert("Повідомлення", "Товар вже доданий в корзину!", "OK");
}
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "OK");
}
}
//AboutUsPage
string aboutUsPageInfo;
public string AboutUsPageInfo
{
get { return aboutUsPageInfo; }
set
{
if(aboutUsPageInfo!=value)
{
aboutUsPageInfo = value;
OnPropertyChanged();
}
}
}
string aboutUsPageRights;
public string AboutUsPageRights
{
get { return aboutUsPageRights; }
set
{
if (aboutUsPageRights != value)
{
aboutUsPageRights = value;
OnPropertyChanged();
}
}
}
string aboutUsPageCertificate;
public string AboutUsPageCertificate
{
get { return aboutUsPageCertificate; }
set
{
if (aboutUsPageCertificate != value)
{
aboutUsPageCertificate = value;
OnPropertyChanged();
}
}
}
public void SetAboutUsPage()
{
AboutUsPageRights = "Copyright © WaterWay.inc " + System.DateTime.Today.Year.ToString(); string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("SELECT * FROM aboutUs WHERE id=1", connection); MySqlDataReader reader = command.ExecuteReader(); if (reader.Read())
{
```

```
AboutUsPageCertificate = reader[1].ToString();
AboutUsPageInfo = reader[2].ToString();
}
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
//AcceptOrderPage
string acceptOrderPageAdress;
public string AcceptOrderPageAdress
{
get { return acceptOrderPageAdress; }
set
{
if (acceptOrderPageAdress != value)
{
acceptOrderPageAdress = value;
OnPropertyChanged();
}
}
}
DateTime acceptOrderDatePicker = System.DateTime.Now;
public DateTime AcceptOrderDatePicker
{
get { return acceptOrderDatePicker; }
set
{
if (acceptOrderDatePicker != value)
{
acceptOrderDatePicker = value;
OnPropertyChanged();
}
}
}
public Command LaunchAcceptOrderButton_Clicked { get; }
async void AcceptOrderButton_Clicked()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("INSERT INTO orders(idClient,idProduct,amount,cost,date,dateOfCreation)" +
" VALUES (@idClient, @idProduct, @amount, @cost, @date, @dateOfCreation)", connection); foreach (var item in ListViewBasketPage)
{
command.Parameters.Clear();
command.Parameters.AddWithValue("@idClient", idClient); command.Parameters.AddWithValue("@idProduct", item.Id); command.Parameters.AddWithValue("@amount", item.Amount); command.Parameters.AddWithValue("@cost", item.TotalCost); command.Parameters.AddWithValue("@date",
```

```
AcceptOrderDatePicker); command.Parameters.AddWithValue("@dateOfCreation", System.DateTime.Today);
command.ExecuteNonQuery();
}
}
}
catch (Exception ex)
{
await App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
while (ListViewBasketPage.Count > 0)
ListViewBasketPage[0].Amount = 0;
TotalCostBasketPage = 0;
await App.Current.MainPage.Navigation.PopAsync();
await App.Current.MainPage.DisplayAlert("Повідомлення", "Замовлення створено успішно!", "OK");
}
//AccountPage
string accountPageAdressButtonColor = "LightGray";
public string AccountPageAdressButtonColor
{
get { return accountPageAdressButtonColor; }
set
{
if (accountPageAdressButtonColor != value)
{
accountPageAdressButtonColor = value;
OnPropertyChanged();
}
}
}
string accountPageInfoButtonColor = "#13A5E4";
public string AccountPageInfoButtonColor
{
get { return accountPageInfoButtonColor; }
set
{
if (accountPageInfoButtonColor != value)
{
accountPageInfoButtonColor = value;
OnPropertyChanged();
}
}
}
bool accountPageAdressIsVisible = false;
public bool AccountPageAdressIsVisible
{
get { return accountPageAdressIsVisible; }
set
{
if (accountPageAdressIsVisible != value)
{
accountPageAdressIsVisible = value;
OnPropertyChanged();
}
}
}
bool accountPageInfoIsVisible = true;
public bool AccountPageInfoIsVisible
{
```

```
get { return accountPageInfoIsVisible; }
set
{
if (accountPageInfoIsVisible != value)
{
accountPageInfoIsVisible = value;
OnPropertyChanged();
}
}
}
string accountPageNamePlaceholder;
public string AccountPageNamePlaceholder
{
get { return accountPageNamePlaceholder; }
set
{
if (accountPageNamePlaceholder != value)
{
accountPageNamePlaceholder = value;
OnPropertyChanged();
}
}
}
string accountPagePhonePlaceholder;
public string AccountPagePhonePlaceholder
{
get { return accountPagePhonePlaceholder; }
set
{
if (accountPagePhonePlaceholder != value)
{
accountPagePhonePlaceholder = value;
OnPropertyChanged();
}
}
}
bool accountPageIsPassword;
public bool AccountPageIsPassword
{
get { return accountPageIsPassword; }
set
{
if (accountPageIsPassword != value)
{
accountPageIsPassword = value;
OnPropertyChanged();
}
}
}
string accountPageCityPlaceholder;
public string AccountPageCityPlaceholder
{
get { return accountPageCityPlaceholder; }
set
{
if (accountPageCityPlaceholder != value)
{
accountPageCityPlaceholder = value;
OnPropertyChanged();
}
}
}
```

```
string accountPageStreetPlaceholder;
public string AccountPageStreetPlaceholder
{
get { return accountPageStreetPlaceholder; }
set
{
if (accountPageStreetPlaceholder != value)
{
accountPageStreetPlaceholder = value;
OnPropertyChanged();
}
}
}
string accountPageHouseNumberPlaceholder;
public string AccountPageHouseNumberPlaceholder
{
get { return accountPageHouseNumberPlaceholder; }
set
{
if (accountPageHouseNumberPlaceholder != value)
{
accountPageHouseNumberPlaceholder = value;
OnPropertyChanged();
}
}
}
string accountPageFlatNumberPlaceholder="";
public string AccountPageFlatNumberPlaceholder
{
get { return accountPageFlatNumberPlaceholder; }
set
{
if (accountPageFlatNumberPlaceholder != value)
{
accountPageFlatNumberPlaceholder = value;
OnPropertyChanged();
}
}
}
string accountPageFloorPlaceholder;
public string AccountPageFloorPlaceholder
{
get { return accountPageFloorPlaceholder; }
set
{
if (accountPageFloorPlaceholder != value)
{
accountPageFloorPlaceholder = value;
OnPropertyChanged();
}
}
}
string accountPageCodePlaceholder;
public string AccountPageCodePlaceholder
{
get { return accountPageCodePlaceholder; }
set
{
if (accountPageCodePlaceholder != value)
{
accountPageCodePlaceholder = value;
OnPropertyChanged();
```

```
}
}
}
string accountPageName;
public string AccountPageName
{
get { return accountPageName; }
set
{
if (accountPageName != value)
{
accountPageName = value;
OnPropertyChanged();
}
}
}
string accountPagePhone;
public string AccountPagePhone
{
get { return accountPagePhone; }
set
{
if (accountPagePhone != value)
{
accountPagePhone = value;
OnPropertyChanged();
}
}
}
string accountPagePassword;
public string AccountPagePassword
{
get { return accountPagePassword; }
set
{
if (accountPagePassword != value)
{
accountPagePassword = value;
OnPropertyChanged();
}
}
}
string accountPageCity;
public string AccountPageCity
{
get { return accountPageCity; }
set
{
if (accountPageCity != value)
{
accountPageCity = value;
OnPropertyChanged();
}
}
}
string accountPageStreet;
public string AccountPageStreet
{
get { return accountPageStreet; }
set
{
if (accountPageStreet != value)
```

```
{
accountPageStreet = value;
OnPropertyChanged();
}
}
}
string accountPageHouseNumber;
public string AccountPageHouseNumber
{
get { return accountPageHouseNumber; }
set
{
if (accountPageHouseNumber != value)
{
accountPageHouseNumber = value;
OnPropertyChanged();
}
}
}
string accountPageFlatNumber;
public string AccountPageFlatNumber
{
get { return accountPageFlatNumber; }
set
{
if (accountPageFlatNumber != value)
{
accountPageFlatNumber = value;
OnPropertyChanged();
}
}
}
string accountPageFloor;
public string AccountPageFloor
{
get { return accountPageFloor; }
set
{
if (accountPageFloor != value)
{
accountPageFloor = value;
OnPropertyChanged();
}
}
}
string accountPageCode;
public string AccountPageCode
{
get { return accountPageCode; }
set
{
if (accountPageCode != value)
{
accountPageCode = value;
OnPropertyChanged();
}
}
}
bool accountPageAdressFlatIsVisible;
public bool AccountPageAdressFlatIsVisible
{
get { return accountPageAdressFlatIsVisible; }
```

```
set
{
if(accountPageAdressFlatIsVisible != value)
{
accountPageAdressFlatIsVisible = value;
OnPropertyChanged();
}
}
}
public Command LaunchAccountPageExit { get; }
public Command LauncAccountPageChangeButton { get; }

public Command LaunchAccountPageChangeIsPassword { get; }
public Command LaunchAccountPageSave { get; }
void AccountPageExit()
{
App.Current.MainPage.Navigation.InsertPageBefore(new LogInPage(), mainPage);
App.Current.MainPage.Navigation.PopAsync();
}
void AccountPageChangeButton()
{
AccountPageInfoIsVisible = !AccountPageInfoIsVisible;
AccountPageAdressIsVisible = !AccountPageAdressIsVisible;
if(AccountPageAdressIsVisible==true)
{
AccountPageAdressButtonColor = "#13A5E4";
AccountPageInfoButtonColor = "LightGray";
}
else
{
AccountPageAdressButtonColor = "LightGray";
AccountPageInfoButtonColor = "#13A5E4";
}
}
void AccountPageChangeIsPassword()
{
if (AccountPageIsPassword == true)
AccountPageIsPassword = false;
else
AccountPageIsPassword = true;
}
async Task AccountPageSave()
{
if (AccountPagePhone == null || AccountPagePhone == "" || AccountPagePhone.Length == 10)
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
MySqlCommand command = new MySqlCommand("UPDATE adress SET city=@city, street=@street,
houseNumber=@houseNumber, flatNumber=@flatNumber, floor=@floor, code=@code" +
" WHERE adress.idClient = @idClient", connection); command.Parameters.AddWithValue("@idClient", idClient); if
(AccountPageCity == null || AccountPageCity == "") command.Parameters.AddWithValue("@city",
AccountPageCityPlaceholder); else
command.Parameters.AddWithValue("@city", AccountPageCity); if (AccountPageStreet == null || AccountPageStreet
== "") command.Parameters.AddWithValue("@street", AccountPageStreetPlaceholder); else
command.Parameters.AddWithValue("@street", AccountPageStreet);
if (AccountPageHouseNumber == null || AccountPageHouseNumber == "")
command.Parameters.AddWithValue("@houseNumber", AccountPageHouseNumberPlaceholder); else
```

```
command.Parameters.AddWithValue("@houseNumber", AccountPageHouseNumber); if (AccountPageFlatNumber == null || AccountPageFlatNumber == "") command.Parameters.AddWithValue("@flatNumber", AccountPageFlatNumberPlaceholder); else
command.Parameters.AddWithValue("@flatNumber", AccountPageFlatNumber); if (AccountPageFloor == null || AccountPageFloor == "") command.Parameters.AddWithValue("@floor", AccountPageFloorPlaceholder); else
command.Parameters.AddWithValue("@floor", AccountPageFloor); if (AccountPageCode == null || AccountPageCode == "") command.Parameters.AddWithValue("@code", AccountPageCodePlaceholder); else
command.Parameters.AddWithValue("@code", AccountPageCode);
connection.Open();
command.ExecuteNonQuery();
command.Parameters.Clear();
if (AccountPagePassword == null || AccountPagePassword == "") command.CommandText = "UPDATE client SET name = @name, phone = @phone" +
" WHERE client.id = @idClient";
else
{
command.CommandText = "UPDATE client SET name = @name, phone = @phone, password = @password" +
"  WHERE client.id = @idClient"; command.Parameters.AddWithValue("@password", AccountPagePassword);
}
command.Parameters.AddWithValue("@idClient", idClient); if (AccountPageName == null || AccountPageName == "") command.Parameters.AddWithValue("@name", AccountPageNamePlaceholder); else
command.Parameters.AddWithValue("@name", AccountPageName); if (AccountPagePhone == null || AccountPagePhone == "") command.Parameters.AddWithValue("@phone", AccountPagePhonePlaceholder); else
command.Parameters.AddWithValue("@phone", AccountPagePhone);
command.ExecuteNonQuery();
}
}
catch (Exception ex)
{
await App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
await App.Current.MainPage.DisplayAlert("Повідомлення", "Оновлено успішно!", "ok"); SetAccountPage();
}
else
await App.Current.MainPage.DisplayAlert("Повідомлення", "Некоректний номер телефону!", "ok");
}
public void SetAccountPage()
{
AccountPageName = "";
AccountPagePhone = "";
AccountPageCity = "";
AccountPageStreet = "";
AccountPageHouseNumber = "";
AccountPageFlatNumber = "";
AccountPageFloor = "";
AccountPageCode = "";
AccountPageIsPassword = true;
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
MySqlCommand command = new MySqlCommand(" SELECT adress.city, "+
"  adress.street," +
"  adress.houseNumber," +
"  adress.flatNumber," +
```

```
"  adress.floor," +
"  adress.code" +
"  FROM adress" +
"  WHERE adress.idClient = @idClient", connection); command.Parameters.AddWithValue("@idClient", idClient);
connection.Open();
var reader = command.ExecuteReader();
if (reader.Read())
{
AccountPageCityPlaceholder = reader[0].ToString();
AccountPageStreetPlaceholder = reader[1].ToString();
AccountPageHouseNumberPlaceholder = reader[2].ToString();
AccountPageFlatNumberPlaceholder = reader[3].ToString();
AccountPageFloorPlaceholder = reader[4].ToString();
AccountPageCodePlaceholder = reader[5].ToString();
}
reader.Close();
command.CommandText = " SELECT client.name,client.phone,client.bonuses"+
"  FROM client"+
" WHERE client.id = @idClient"; command.Parameters.Clear(); command.Parameters.AddWithValue("@idClient",
idClient); reader = command.ExecuteReader();
if (reader.Read())
{
AccountPageNamePlaceholder = reader[0].ToString(); AccountPagePhonePlaceholder = reader[1].ToString();
}
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
if((AccountPageFlatNumberPlaceholder == "0" || AccountPageFlatNumberPlaceholder == null) &&
(AccountPageFloorPlaceholder == "0" || AccountPageFloorPlaceholder == null) && (AccountPageCodePlaceholder ==
"0" || AccountPageCodePlaceholder == null)) AccountPageAdressFlatIsVisible = false;
else
AccountPageAdressFlatIsVisible = true;
}
//BasketPage
public NavigationPage basketPage;
bool labelBasketPageEmptyIsVisible = true;
public bool LabelBasketPageEmptyIsVisible
{
get { return labelBasketPageEmptyIsVisible; }
set
{
if (labelBasketPageEmptyIsVisible != value)
{
labelBasketPageEmptyIsVisible = value;
OnPropertyChanged();
}
}
}
bool labelBasketPageTotalCostIsVisible;
public bool LabelBasketPageTotalCostIsVisible
{
get { return labelBasketPageTotalCostIsVisible; }
set
{
```

```
if (labelBasketPageTotalCostIsVisible != value)
{
labelBasketPageTotalCostIsVisible = value;
OnPropertyChanged();
}
}
}
bool buttonBasketPageMakeOrderIsVisible;
public bool ButtonBasketPageMakeOrderIsVisible
{
get { return buttonBasketPageMakeOrderIsVisible; }
set
{
if (buttonBasketPageMakeOrderIsVisible != value)
{
buttonBasketPageMakeOrderIsVisible = value;
OnPropertyChanged();
}
}
}
float totalCostBasketPage;
public float TotalCostBasketPage
{
get { return totalCostBasketPage; }
set
{
if (totalCostBasketPage != value)
{
totalCostBasketPage = value;
OnPropertyChanged();
if (listViewBasketPage.Count == 0)
{
LabelBasketPageEmptyIsVisible = true;
LabelBasketPageTotalCostIsVisible = false;
ButtonBasketPageMakeOrderIsVisible = false;
}
else
{
LabelBasketPageEmptyIsVisible = false;
LabelBasketPageTotalCostIsVisible = true;
ButtonBasketPageMakeOrderIsVisible = true;
}
}
}
}
ObservableCollection<ModelBasketPage> listViewBasketPage;
public ObservableCollection<ModelBasketPage> ListViewBasketPage
{
get { return listViewBasketPage; }
set
{
if (listViewBasketPage != value)
{
listViewBasketPage = value;
OnPropertyChanged();
}
}
}
public Command LaunchBasketPageClear { get; }
public Command LaunchBasketPageAddInBusketButton_Clicked { get; }
void BasketPageAddInBusketButton_Clicked()
{
```

```
if (AccountPageFlatNumberPlaceholder != "0" && AccountPageFlatNumberPlaceholder != null)
AcceptOrderPageAdress = "м. " + AccountPageCityPlaceholder +
"  вул. " + AccountPageStreetPlaceholder +
"  " + AccountPageHouseNumberPlaceholder+
"  квартира "+ AccountPageFlatNumberPlaceholder;
else
AcceptOrderPageAdress = "м. " + AccountPageCityPlaceholder +
"  вул. " + AccountPageStreetPlaceholder +
"  "  +  AccountPageHouseNumberPlaceholder;  App.Current.MainPage.Navigation.PushAsync(new
AcceptOrderPage(mainViewModel));
}
async void BasketPageClear()
{
bool result = await App.Current.MainPage.DisplayAlert("Видалення...", "Очистити корзину?", "Ні", "Так"); if
(!result)
{
while (ListViewBasketPage.Count > 0)
ListViewBasketPage[0].Amount = 0;
TotalCostBasketPage = 0;
}
}
//DiscountPage
ObservableCollection<ModelDiscountPage> listViewDiscountPage;
public ObservableCollection<ModelDiscountPage> ListViewDiscountPage
{
get { return listViewDiscountPage; }
set
{
if (listViewDiscountPage != value)
{
listViewDiscountPage = value;
OnPropertyChanged();
}
}
}
public void SetDiscountPage()
{
string  connectionString  =  @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
MySqlCommand command = new MySqlCommand(" SELECT * FROM discount", connection); connection.Open();
var reader = command.ExecuteReader();
ListViewDiscountPage=new ObservableCollection<ModelDiscountPage>();
while (reader.Read())
{
ListViewDiscountPage.Add(new ModelDiscountPage
{
Title = reader[1].ToString(),
ImageSource = reader[2].ToString(),
Description = reader[3].ToString(),
DateFrom=((DateTime)reader[4]).ToShortDateString(),
DateTo = ((DateTime)reader[5]).ToShortDateString()
});
}
reader.Close();
}
}
catch (Exception ex)
```

```
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
//HistoryPage
public ObservableCollection<ModelHistoryPage> HistoryList { get; set; }
public ObservableCollection<ModelHistoryPage> showHistoryList;
public ObservableCollection<ModelHistoryPage> ShowHistoryList
{
get
{
return showHistoryList;
}
set
{
if (showHistoryList != value)
{
showHistoryList = value;
OnPropertyChanged();
if (HistoryList.Count == 0 && ShowHistoryList.Count == 0)
HistoryEmptyLabelIsVisible = true;
else
HistoryEmptyLabelIsVisible = false;
}
}
}
public Command LaunchHistorySearchButton_Clicked { get; }
string historySearchEntry;
public string HistorySearchEntry
{
get { return historySearchEntry; }
set
{
if (historySearchEntry != value)
{
historySearchEntry = value;
OnPropertyChanged();
HistorySearchEntry_TextChanged();
}
}
}
bool historyEmptyLabelIsVisible;
public bool HistoryEmptyLabelIsVisible
{
get { return historyEmptyLabelIsVisible; }
set
{
if(historyEmptyLabelIsVisible!=value)
{
historyEmptyLabelIsVisible = value;
OnPropertyChanged();
}
}
}
bool historySearchEntryIsVisible;
public bool HistorySearchEntryIsVisible
{
get { return historySearchEntryIsVisible; }
```

```
set
{
if (historySearchEntryIsVisible != value)
{
historySearchEntryIsVisible = value;
OnPropertyChanged();
}
}
}
void HistorySearchEntry_TextChanged()
{
if (HistorySearchEntry == string.Empty && ShowHistoryList!=null)
{
ShowHistoryList.Clear();
ShowHistoryList = new ObservableCollection<ModelHistoryPage>(HistoryList);
}
else
{
ShowHistoryList = new ObservableCollection<ModelHistoryPage>(HistoryList.Where(item =>
item.Title.ToLower().Contains(HistorySearchEntry.ToLower()) ||
item.Amount.ToLower().Contains(HistorySearchEntry.ToLower()) ||
item.Cost.ToString().ToLower().Contains(HistorySearchEntry.ToLower()) ||
item.Date.ToLower().Contains(HistorySearchEntry.ToLower())));
}
}
void HistorySearchButton_Clicked()
{
HistorySearchEntry = "";
HistorySearchEntryIsVisible = !HistorySearchEntryIsVisible;
}
public async void SetHistoryPage()
{
HistoryEmptyLabelIsVisible = false;
HistorySearchEntryIsVisible = false;
HistoryList = new ObservableCollection<ModelHistoryPage>();
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("Select
product.id,product.title,product.image,orders.amount,orders.cost,orders.date,orders.dateOfCreation,orders.isDone"+
"  FROM product, orders"+
" WHERE product.id = orders.idProduct AND orders.idClient = @idClient", connection);
command.Parameters.AddWithValue("@idClient", idClient);
MySqlDataReader reader = command.ExecuteReader(); while (reader.Read())
{
HistoryList.Add( new ModelHistoryPage
{
IdProduct=(int)reader[0], Title = reader[1].ToString(), Image= reader[2].ToString(), Amount = reader[3].ToString(),
Cost = (float)reader[4],
Date = ((DateTime)reader[5]).ToShortDateString(), DateOfCreation = ((DateTime)reader[6]).ToShortDateString(),
StatusText=reader[7].ToString()
});
}
reader.Close();
}
ShowHistoryList = new ObservableCollection<ModelHistoryPage>(HistoryList.OrderBy(i=>i.DateOfCreation));
}
```

```
catch (Exception ex)
{
await mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
//NotificationPage
string notificationPageMessage;
public string NotificationPageMessage
{
get { return notificationPageMessage; }
set
{
if(notificationPageMessage!=value)
{
notificationPageMessage = value;
OnPropertyChanged();
}
}
}
ObservableCollection<ModelNotificationPage> listViewNotificationPage; public
ObservableCollection<ModelNotificationPage> ListViewNotificationPage
{
get { return listViewNotificationPage; }
set
{
if (listViewNotificationPage != value)
{
listViewNotificationPage = value;
OnPropertyChanged();
}
}
}
public Command LaunchNotificationPageClear_Clicked { get; }
public Command LaunchNotificationPageSent_Clicked { get; }
async void NotificationPageClear_Clicked()
{
bool result = await App.Current.MainPage.DisplayAlert("Видалення...", "Очистити чат?", "Ні", "Так");
if (!result)
{
ListViewNotificationPage.Clear();
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("Delete FROM notification" +
" WHERE idClient = @idClient", connection); command.Parameters.AddWithValue("@idClient", idClient); var
resultOfDeleting = command.ExecuteNonQuery(); if(resultOfDeleting>0)
await mainPage.DisplayAlert("Повідомлення!", "Чат очищено!", "ok");
}
}
catch (Exception ex)
{
await mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
```

```
finally
{
connection.Close();
}
}
}
async void NotificationPageSent_Clicked()
{
if (NotificationPageMessage != null && NotificationPageMessage != "")
{
ListViewNotificationPage.Add(new ModelNotificationPage
{
Sender = "client",
Text = NotificationPageMessage,
Date = System.DateTime.Today.ToShortDateString()
});
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("INSERT INTO notification(idClient,sender,text,date)" + "VALUES(@idClient,@sender,@text,@date)", connection);
command.Parameters.AddWithValue("@idClient", idClient);
command.Parameters.AddWithValue("@sender", "client");
command.Parameters.AddWithValue("@text", NotificationPageMessage);
command.Parameters.AddWithValue("@date", System.DateTime.Today);
command.ExecuteNonQuery();
}
NotificationPageMessage = "";
}
catch (Exception ex)
{
await mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
public void SetNotificationPage()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("UPDATE notification set seen=true WHERE idClient=@idClient AND sender='admin'", connection);
command.Parameters.AddWithValue("@idClient", idClient);
command.ExecuteNonQuery();
command.CommandText = "Select sender,text,date,seen" +
"  FROM notification" +
"  WHERE idClient = @idClient"; MySqlDataReader reader = command.ExecuteReader();
ListViewNotificationPage = new ObservableCollection<ModelNotificationPage>(); while (reader.Read())
```

```
{
ListViewNotificationPage.Add(new ModelNotificationPage
{
Sender = reader[0].ToString(), Text = reader[1].ToString(),
Date = ((DateTime)reader[2]).ToShortDateString(), BoolSeen = (bool)reader[3]
});
}
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
public MVM()
{
mainViewModel = this;
AboutProductPageSetPhone();
//PoductPage
//AboutProductPage
LaunchAboutProductPageMinusButton_Clicked = new Command(AboutProductPageMinusButton_Clicked);
LaunchAboutProductPagePlusButton_Clicked = new Command(AboutProductPagePlusButton_Clicked);
LaunchAboutProductPageAddInBusketButton_Clicked = new
Command(AboutProductPageAddInBusketButton_Clicked);
//BasketPage
ListViewBasketPage = new ObservableCollection<ModelBasketPage>();
LaunchBasketPageClear = new Command(BasketPageClear);
LaunchBasketPageAddInBusketButton_Clicked = new Command(BasketPageAddInBusketButton_Clicked);
basketPage = new NavigationPage();
modelProductPage = new ModelProductPage(this);
//AcceptOrderPage
LaunchAcceptOrderButton_Clicked = new Command(AcceptOrderButton_Clicked);
//AccountPage
LaunchAccountPageExit = new Command(AccountPageExit); LaunchAboutProductPageCall = new
Command(AboutProductPageCall); LauncAccountPageChangeButton = new Command(AccountPageChangeButton);
LaunchAccountPageChangeIsPassword = new Command(AccountPageChangeIsPassword); LaunchAccountPageSave
= new Command(async ()=> await AccountPageSave());
//HistoryPage
LaunchHistorySearchButton_Clicked = new Command(HistorySearchButton_Clicked);
//NotificationPage
LaunchNotificationPageClear_Clicked = new Command(NotificationPageClear_Clicked);
LaunchNotificationPageSent_Clicked = new Command(NotificationPageSent_Clicked);
}
}
}
```

Файл «MVVMAdmin.cs»

```
using Admin.Models;
using Admin.Models.ModelAdmin;
using Admin.View;
using Admin.View.ViewAdmin;
using MySql.Data.MySqlClient;
using Plugin.Messaging;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

```
using System.Data;
using System.Linq;
using System.Text;
using Xamarin.Forms;
namespace Admin.ModelViewViewModel
{
public class MVVMAdmin:INPC
{
public MVVMAdmin mainViewModel;
public AdminMainPage mainPage;
public AdminMainPageDetail mainPageDetail;
public NavigationPage productListPage,workerPage,clientPage,aboutUsPage,discountPage; public int IdAdmin { get; set; }
//MainPageMaster
ObservableCollection<ModelMainPageMaster> listViewMainPageMaster; public ObservableCollection<ModelMainPageMaster> ListViewMainPageMaster
{
get { return listViewMainPageMaster; }
set
{
if (listViewMainPageMaster != value)
{
listViewMainPageMaster = value;
OnPropertyChanged();
}
}
}
public Command LaunchMainPageMasterExit_Clicked { get; }
void MainPageMasterExit_Clicked()
{
mainPage.Navigation.InsertPageBefore(new LogInPage(), mainPage);
mainPage.Navigation.PopAsync();
}
public void SetMainPageMaster()
{
ListViewMainPageMaster = new ObservableCollection<ModelMainPageMaster>
{
new ModelMainPageMaster{Logo="MainPage.png",Title="Головна сторінка"}, new ModelMainPageMaster{Logo="ProductPage.png",Title="Товари"},
new ModelMainPageMaster{Logo="DiscountPage.png",Title="Акції"},
new ModelMainPageMaster{Logo="WorkerPage.png",Title="База робітників"}, new ModelMainPageMaster{Logo="ClientInfoListPage.png",Title="База клієнтів"}, new ModelMainPageMaster{Logo="AboutUsPage.png",Title="Про нас"}
};
}
//AboutUsPage
string aboutUsPagePhone;
public string AboutUsPagePhone
{
get { return aboutUsPagePhone; }
set
{
aboutUsPagePhone = value;
OnPropertyChanged();
}
}
string aboutUsPagePhonePlaceholder;
public string AboutUsPagePhonePlaceholder
{
get { return aboutUsPagePhonePlaceholder; }
set
{
```

```
aboutUsPagePhonePlaceholder = value;
OnPropertyChanged();
}
}
string aboutUsPageAboutUs;
public string AboutUsPageAboutUs
{
get { return aboutUsPageAboutUs; }
set
{
aboutUsPageAboutUs = value;
OnPropertyChanged();
}
}
string aboutUsPageAboutUsPlaceholder;
public string AboutUsPageAboutUsPlaceholder
{
get { return aboutUsPageAboutUsPlaceholder; }
set
{
aboutUsPageAboutUsPlaceholder = value;
OnPropertyChanged();
}
}
public Command LaunchAboutUsPageButton { get; }
void AboutUsPageButton()
{
if (AboutUsPagePhone == null || AboutUsPagePhone == "")
AboutUsPagePhone = AboutUsPagePhonePlaceholder;
if (AboutUsPageAboutUs == null || AboutUsPageAboutUs == "")
AboutUsPageAboutUs = AboutUsPageAboutUsPlaceholder;
if (AboutUsPagePhone.Length == 10)
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("UPDATE aboutUs SET phone=@phone,aboutUs=@aboutUs WHERE id=1", connection))
{
try
{
connection.Open();
command.Parameters.AddWithValue("@phone", AboutUsPagePhone);
command.Parameters.AddWithValue("@aboutUs", AboutUsPageAboutUs); var result = command.ExecuteNonQuery();
if (result > 0)
App.Current.MainPage.DisplayAlert("", "Оновлено успішно!", "OK");
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
AboutUsPagePhone = "";
AboutUsPageAboutUs = "";
SetAboutUsPage();
```

```
}
else
App.Current.MainPage.DisplayAlert("", "Некоректний номер телефону!", "OK");
}
public void SetAboutUsPage()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("Select * FROM aboutUs" +
" Where id=1", connection))
{
try
{
connection.Open();
var reader = command.ExecuteReader();
if (reader.Read())
{
AboutUsPageAboutUsPlaceholder = reader[2].ToString();
AboutUsPagePhonePlaceholder = reader[3].ToString();
}
reader.Close();
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
}
//ClientInfoPage
public string ClientInfoPageName { get; set; }
public string ClientInfoPagePhone { get; set; }
string clientInfoPageAdress;
public string ClientInfoPageAdress
{
get { return clientInfoPageAdress; }
set
{
if(clientInfoPageAdress!=value)
{
clientInfoPageAdress = value;
OnPropertyChanged();
}
}
}
ObservableCollection<ModelAdditionPage> clientInfoPageNotationList; public
ObservableCollection<ModelAdditionPage> ClientInfoPageNotationList
{
get { return clientInfoPageNotationList; }
set
{
if (clientInfoPageNotationList != value)
{
clientInfoPageNotationList = value;
OnPropertyChanged();
}
```

```
}
}
public Command LaunchClientInfoPageCallButton { get; }
void ClientInfoPageCallButton()
{
var phoneDialer = CrossMessaging.Current.PhoneDialer;
if (phoneDialer.CanMakePhoneCall)
phoneDialer.MakePhoneCall(ClientInfoPagePhone);
}
public void SetClientInfoPage(int idClient)
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("Select notation,date FROM notation" + " Where idClient=@idClient", connection))
{
try
{
connection.Open();
command.Parameters.AddWithValue("@idClient", idClient);
var reader = command.ExecuteReader();
ClientInfoPageNotationList = new ObservableCollection<ModelAdditionPage>();
while (reader.Read())
{
ClientInfoPageNotationList.Add(new ModelAdditionPage
{
Date = ((DateTime)reader[1]).ToShortDateString(),
Notation = reader[0].ToString()
});
}
reader.Close();
ClientInfoPageNotationList = new ObservableCollection<ModelAdditionPage>(ClientInfoPageNotationList.OrderByDescending(i => i.Date));
command.CommandText = "SELECT name,phone FROM client WHERE id=@idClient";
reader = command.ExecuteReader();
if (reader.Read())
{
ClientInfoPageName = reader[0].ToString();
ClientInfoPagePhone = "+38"+reader[1].ToString();
}
reader.Close();
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
}
//DiscountPage
public Command LaunchDiscountPageAddButton { get; }
void DiscountPageAddButton()
{
mainPage.Navigation.PushAsync(new DiscountAddPage(mainViewModel));
}
```

```
ObservableCollection<ModelDiscountPage> listViewDiscountPage;
public ObservableCollection<ModelDiscountPage> ListViewDiscountPage
{
get { return listViewDiscountPage; }
set
{
if (listViewDiscountPage != value)
{
listViewDiscountPage = value;
OnPropertyChanged();
}
}
}
public void SetDiscountPage()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("Select * FROM discount", connection))
{
try
{
connection.Open();
var reader = command.ExecuteReader();
ListViewDiscountPage = new ObservableCollection<ModelDiscountPage>(); while (reader.Read())
{
ListViewDiscountPage.Add(new ModelDiscountPage(mainViewModel)
{
Title=reader[1].ToString(),
DateFrom=((DateTime)reader[4]).ToShortDateString(),
DateTo= ((DateTime)reader[5]).ToShortDateString()
});;
}
reader.Close();
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
}
//ProductListPage
string productListPageAmount;
public string ProductListPageAmount
{
get { return productListPageAmount; }
set
{
if(productListPageAmount!=value)
{
productListPageAmount = value;
OnPropertyChanged();
}
}
}
ObservableCollection<ModelProductListPage> listViewProductListPage; public
```

```
ObservableCollection<ModelProductListPage> ListViewProductListPage
{
get { return listViewProductListPage; }
set
{
if(listViewProductListPage!=value)
{
listViewProductListPage = value;
OnPropertyChanged();
}
}
}
public Command LaunchProductListPageAddButton { get; }
void ProductListPageAddButton()
{
App.Current.MainPage.Navigation.PushAsync(new ProductAddPage(mainViewModel));
}
public void SetProductListPage()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("Select * FROM product", connection))
{
try
{
connection.Open();
var reader = command.ExecuteReader();
ListViewProductListPage = new ObservableCollection<ModelProductListPage>(); while (reader.Read())
{
ListViewProductListPage.Add(new ModelProductListPage(mainViewModel)
{
IdProduct=(int)reader[0],
Title=reader[1].ToString(),
Type=reader[6].ToString()
});
}
reader.Close();
ListViewProductListPage = new
ObservableCollection<ModelProductListPage>(ListViewProductListPage.OrderByDescending(i =>
i.Type));
ProductListPageAmount = ListViewProductListPage.Count.ToString();
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
}
//ProductEditPage
int ProductEditPageId { get; set; }
string productEditPageTitle;
public string ProductEditPageTitle
{
get { return productEditPageTitle; }
set
```

```
{
if(productEditPageTitle!=value)
{
productEditPageTitle = value;
OnPropertyChanged();
}
}
}
string productEditPageTitlePlaceholder;
public string ProductEditPageTitlePlaceholder
{
get { return productEditPageTitlePlaceholder; }
set
{
if (productEditPageTitlePlaceholder != value)
{
productEditPageTitlePlaceholder = value;
OnPropertyChanged();
}
}
}
string productEditPageCost;
public string ProductEditPageCost
{
get { return productEditPageCost; }
set
{
if (productEditPageCost != value)
{
productEditPageCost = value;
OnPropertyChanged();
}
}
}
string productEditPageCostPlaceholder;
public string ProductEditPageCostPlaceholder
{
get { return productEditPageCostPlaceholder; }
set
{
if (productEditPageCostPlaceholder != value)
{
productEditPageCostPlaceholder = value;
OnPropertyChanged();
}
}
}
string productEditPageType;
public string ProductEditPageType
{
get { return productEditPageType; }
set
{
if(productEditPageType != value)
{
productEditPageType = value;
OnPropertyChanged();
}
}
}
string productEditPageDescription;
public string ProductEditPageDescription
```

```
{
get { return productEditPageDescription; }
set
{
if (productEditPageDescription != value)
{
productEditPageDescription = value;
OnPropertyChanged();
}
}
}
string productEditPageDescriptionPlaceholder;
public string ProductEditPageDescriptionPlaceholder
{
get { return productEditPageDescriptionPlaceholder; }
set
{
if (productEditPageDescriptionPlaceholder != value)
{
productEditPageDescriptionPlaceholder = value;
OnPropertyChanged();
}
}
}
public Command LaunchProductEditPageSaveButton { get; }
void ProductEditPageSaveButton()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("UPDATE product SET title=@title,description=@description,cost=@cost,type=@type WHERE id=@idProduct", connection))
{
try
{
connection.Open();
command.Parameters.AddWithValue("@idProduct", ProductEditPageId);
if (ProductEditPageTitle != null && ProductEditPageTitle != "")
command.Parameters.AddWithValue("@title", ProductEditPageTitle);
else
command.Parameters.AddWithValue("@title", ProductEditPageTitlePlaceholder);
if (ProductEditPageDescription != null && ProductEditPageDescription != "")
command.Parameters.AddWithValue("@description", ProductEditPageDescription); else
command.Parameters.AddWithValue("@description", ProductEditPageDescriptionPlaceholder);
if (ProductEditPageCost != null && ProductEditPageCost != "")
command.Parameters.AddWithValue("@cost", ProductEditPageCost);
else
command.Parameters.AddWithValue("@cost", ProductEditPageCostPlaceholder);
if (ProductEditPageType != null && ProductEditPageType != "")
command.Parameters.AddWithValue("@type", ProductEditPageType);
else
command.Parameters.AddWithValue("@type", ProductEditPageType);
var result = command.ExecuteNonQuery();
if (result > 0)
App.Current.MainPage.DisplayAlert("", "Оновлено успішно!", "OK");
SetProductEditPage(ProductEditPageId);
SetProductListPage();
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
```

```
}
}
}
}
public void SetProductEditPage(int idProduct)
{
ProductEditPageTitle = "";
ProductEditPageDescription = "";
ProductEditPageCost = "";
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("SELECT * FROM product WHERE id = @idProduct", connection))
{
try
{
connection.Open();
command.Parameters.AddWithValue("@idProduct", idProduct);
var reader = command.ExecuteReader();
if (reader.Read())
{
ProductEditPageId = (int)reader[0];
ProductEditPageTitlePlaceholder = reader[1].ToString();
ProductEditPageDescriptionPlaceholder = reader[3].ToString();
ProductEditPageCostPlaceholder = reader[4].ToString();
ProductEditPageType = reader[6].ToString();
}
reader.Close();
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
}
}
}
//ProductAddPage
string productAddPageTitle;
public string ProductAddPageTitle
{
get { return productAddPageTitle; }
set
{
if (productAddPageTitle != value)
{
productAddPageTitle = value;
OnPropertyChanged();
}
}
}
string productAddPageType="water";
public string ProductAddPageType
{
get { return productAddPageType; }
set
{
if (productAddPageType != value)
{
productAddPageType = value;
OnPropertyChanged();
}
```

```
}
}
string productAddPageCost;
public string ProductAddPageCost
{
get { return productAddPageCost; }
set
{
if (productAddPageCost != value)
{
productAddPageCost = value;
OnPropertyChanged();
}
}
}
string productAddPageDescription;
public string ProductAddPageDescription
{
get { return productAddPageDescription; }
set
{
if (productAddPageDescription != value)
{
productAddPageDescription = value;
OnPropertyChanged();
}
}
}
public Command LaunchProductAddPageSaveButton { get; }
void ProductAddPageSaveButton()
{
if (ProductAddPageTitle!=null && ProductAddPageTitle!="" && ProductAddPageDescription!=null && ProductAddPageDescription!="" && ProductAddPageCost != null && ProductAddPageCost!="" && ProductAddPageType!=null && ProductAddPageType!="")
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("INSERT INTO product(title,description,cost,type)" + " VALUES(@title,@description,@cost,@type)", connection))
{
try
{
connection.Open();
command.Parameters.AddWithValue("@title", ProductAddPageTitle);
command.Parameters.AddWithValue("@description", ProductAddPageDescription);
command.Parameters.AddWithValue("@cost", ProductAddPageCost); command.Parameters.AddWithValue("@type", ProductAddPageType);
var result = command.ExecuteNonQuery();
if (result > 0)
{
App.Current.MainPage.DisplayAlert("", "Збережено успішно!", "OK");
App.Current.MainPage.Navigation.PopAsync();
SetProductListPage();
}
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
}
```

```
}
}
else
App.Current.MainPage.DisplayAlert("", "Заповніть всі поля", "OK");
}
//SupportListPage
bool supportListPageLabelEmptyIsVisible = false;
public bool SupportListPageLabelEmptyIsVisible
{
get { return supportListPageLabelEmptyIsVisible; }
set
{
if (supportListPageLabelEmptyIsVisible != value)
{
supportListPageLabelEmptyIsVisible = value;
OnPropertyChanged();
}
}
}
public ModelSupportListPage modelListPage; ObservableCollection<ModelSupportListPage> listViewSupportListPage; public ObservableCollection<ModelSupportListPage> ListViewSupportListPage
{
get { return listViewSupportListPage; }
set
{
if (listViewSupportListPage != value)
{
listViewSupportListPage = value;
OnPropertyChanged();
}
}
}
public void SetSupportListPage()
{
ListViewSupportListPage = new ObservableCollection<ModelSupportListPage>(); string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("Select client.id,adress.city,adress.street,adress.houseNumber,adress.flatNumber" +
"  FROM adress,client" +
"  WHERE client.id=adress.idClient " +
"  GROUP BY client.id", connection))
{
try
{
connection.Open();
var reader = command.ExecuteReader();
while (reader.Read())
if (reader[4].ToString() == "0")
ListViewSupportListPage.Add(new ModelSupportListPage(mainViewModel)
{
IdClient = (int)reader[0],
Adress = "м. " + reader[1].ToString() + " вул. " + reader[2].ToString() + " " + reader[3].ToString(),
AmountOfMessages=0,
UnreadMessage=0
});
else
ListViewSupportListPage.Add(new ModelSupportListPage(mainViewModel)
{
IdClient = (int)reader[0],
```

```
Adress = "м. " + reader[1].ToString() + " вул. " + reader[2].ToString() + " " + reader[3].ToString() + " квартира " +
reader[4].ToString(),
AmountOfMessages = 0,
UnreadMessage=0
});
reader.Close();
command.CommandText = "SELECT client.id,count(notification.id) FROM notification,client " +
"  WHERE notification.sender='client' AND client.id = notification.idClient" +
"  GROUP BY client.id";
reader = command.ExecuteReader();
while (reader.Read())
foreach (var item in ListViewSupportListPage)
if (item.IdClient == (int)reader[0])
{
item.AmountOfMessages = Int32.Parse((reader[1]).ToString());
item.UnreadMessage = 0;
}
reader.Close();
command.CommandText = "SELECT count(notification.seen),client.id FROM notification,client" +"       WHERE seen
= true AND" +
"  client.id = notification.idClient AND" +
"  notification.sender = 'client'" +
"  GROUP BY client.id";
reader = command.ExecuteReader();
while (reader.Read())
foreach (var item in ListViewSupportListPage)
if (item.IdClient == (int)reader[1])
item.UnreadMessage = Int32.Parse((reader[0]).ToString());
connection.Close();
ListViewSupportListPage                                                    =                                                    new
ObservableCollection<ModelSupportListPage>(ListViewSupportListPage.OrderByDescending(i                                    =>
i.UnreadMessage));
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
}
}
if (ListViewSupportListPage.Count == 0)
SupportListPageLabelEmptyIsVisible = true;
}
//SupportPage
string supportPageTitle;
public string SupportStringTitle
{
get { return supportPageTitle; }
set
{
if(supportPageTitle!=value)
{
supportPageTitle = value;
OnPropertyChanged();
}
}
}
int supportPageIdClient;
public int SupportPageIdClient
{
get { return supportPageIdClient; }
set
{
```

```
if(supportPageIdClient!=value)
{
supportPageIdClient = value;
OnPropertyChanged();
}
}
}
string supportPageMessage;
public string SupportPageMessage
{
get { return supportPageMessage; }
set
{
if (supportPageMessage != value)
{
supportPageMessage = value;
OnPropertyChanged();
}
}
}
ObservableCollection<ModelSupportPage> listViewSupportPage;
public ObservableCollection<ModelSupportPage> ListViewSupportPage
{
get { return listViewSupportPage; }
set
{
if (listViewSupportPage != value)
{
listViewSupportPage = value;
OnPropertyChanged();
}
}
}
public Command LaunchSupportPageClear_Clicked { get; }
public Command LaunchSupportPageSent_Clicked { get; }
async void SupportPageClear_Clicked()
{
bool result = await App.Current.MainPage.DisplayAlert("Видалення...", "Очистити чат?", "Ні", "Так"); if (!result)
{
ListViewSupportPage.Clear();
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("Delete FROM notification" +
" WHERE idClient = @idClient", connection); command.Parameters.AddWithValue("@idClient",
supportPageIdClient); var resultOfDeleting = command.ExecuteNonQuery();
if (resultOfDeleting > 0)
await mainPage.DisplayAlert("Повідомлення!", "Чат очищено!", "ok");
}
}
catch (Exception ex)
{
await mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
```

```
}
}
}
async void SupportPageSent_Clicked()
{
if (SupportPageMessage != null && SupportPageMessage != "")
{
ListViewSupportPage.Add(new ModelSupportPage
{
Sender = "admin",
Text = SupportPageMessage,
Date = System.DateTime.Today.ToShortDateString()
});
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("INSERT INTO notification(idClient,sender,text,date)" + "VALUES(@idClient,@sender,@text,@date)", connection);
command.Parameters.AddWithValue("@idClient", SupportPageIdClient);
command.Parameters.AddWithValue("@sender", "admin");
command.Parameters.AddWithValue("@text", SupportPageMessage);
command.Parameters.AddWithValue("@date", System.DateTime.Today);
command.ExecuteNonQuery();
}
SupportPageMessage = "";
}
catch (Exception ex)
{
await mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
}
public void SetSupportPage(int _supportPageIdClient)
{
SupportPageIdClient = _supportPageIdClient;
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
if (connection.State == ConnectionState.Closed)
{
connection.Open();
MySqlCommand command = new MySqlCommand("UPDATE notification set seen=true WHERE idClient=@idClient AND sender='client'", connection);
command.Parameters.AddWithValue("@idClient", SupportPageIdClient);
command.ExecuteNonQuery();
command.CommandText = "Select sender,text,date,seen" +
"  FROM notification" +
" WHERE idClient = @idClient"; MySqlDataReader reader = command.ExecuteReader(); ListViewSupportPage = new ObservableCollection<ModelSupportPage>(); while (reader.Read())
{
ListViewSupportPage.Add(new ModelSupportPage
```

```
{
Sender = reader[0].ToString(), Text = reader[1].ToString(),
Date = ((DateTime)reader[2]).ToShortDateString(), BoolSeen=(bool)reader[3]
});
}
reader.Close();
}
}
catch (Exception ex)
{
mainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
finally
{
connection.Close();
}
}
//WorkerPage
ObservableCollection<ModelWorkerPage> listViewWorkerPage;
public ObservableCollection<ModelWorkerPage> ListViewWorkerPage
{
get { return listViewWorkerPage; }
set
{
listViewWorkerPage = value;
OnPropertyChanged();
}
}
public Command LaunchWorkerPageAddButton { get; }
void WorkerPageAddButton()
{
mainPage.Navigation.PushAsync(new WorkerAddPage(mainViewModel));
}
public void SetWorkerPage()
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("SELECT * FROM worker WHERE id <> @id", connection))
{
try
{
connection.Open();
command.Parameters.AddWithValue("@id",IdAdmin);
var reader = command.ExecuteReader();
ListViewWorkerPage = new ObservableCollection<ModelWorkerPage>();
while (reader.Read())
ListViewWorkerPage.Add(new ModelWorkerPage(mainViewModel)
{
Id=(int)reader[0],
Name=reader[1].ToString(),
Phone="+38"+reader[2].ToString(),
Role=reader[4].ToString()
});
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
}
}
```

```
}
//WorkerAddPage
string workerAddPageName;
public string WorkerAddPageName
{
get { return workerAddPageName; }
set
{
workerAddPageName = value;
OnPropertyChanged();
}
}
string workerAddPagePhone;
public string WorkerAddPagePhone
{
get { return workerAddPagePhone; }
set
{
workerAddPagePhone = value;
OnPropertyChanged();
}
}
string workerAddPagePassword;
public string WorkerAddPagePassword
{
get { return workerAddPagePassword; }
set
{
workerAddPagePassword = value;
OnPropertyChanged();
}
}
string workerAddPageRole;
public string WorkerAddPageRole
{
get { return workerAddPageRole; }
set
{
workerAddPageRole = value;
OnPropertyChanged();
}
}
public Command LaunchWorkerAddPageSaveButton { get; }
void WorkerAddPageSaveButton()
{
if (WorkerAddPageName != null && WorkerAddPageName != "" &&
WorkerAddPagePhone != null && WorkerAddPagePhone != "" &&
WorkerAddPagePassword != null && WorkerAddPagePassword != "" &&
WorkerAddPageRole != null && WorkerAddPageRole != "")
{
if (WorkerAddPagePhone.Length == 10)
{
string connectionString = @"server=artemtnv.beget.tech;database=artemtnv_waterw;user
id=artemtnv_waterw;password=WaterWay.inc;CHARSET=utf8";
using (var connection = new MySqlConnection(connectionString))
{
using (var command = new MySqlCommand("INSERT INTO worker(name,phone,password,role) " + "
VALUES(@name,@phone,@password,@role)", connection))
{
try
{
connection.Open();
```

```
command.Parameters.AddWithValue("@name", WorkerAddPageName);
command.Parameters.AddWithValue("@phone", WorkerAddPagePhone);
command.Parameters.AddWithValue("@password", WorkerAddPagePassword);
command.Parameters.AddWithValue("@role", WorkerAddPageRole);
var result = command.ExecuteNonQuery();
if (result > 0)
{
App.Current.MainPage.DisplayAlert("", "Збережено успішно!", "ok");
mainViewModel.SetWorkerPage();
mainPage.Navigation.PopAsync();
}
}
catch (Exception ex)
{
App.Current.MainPage.DisplayAlert("Error", ex.ToString(), "ok");
}
}
}
}
else
App.Current.MainPage.DisplayAlert("", "Некоректний номер телефону!", "OK");
}
else
App.Current.MainPage.DisplayAlert("", "Заповніть всі поля!", "OK");
}
public MVVMAdmin()
{
mainViewModel = this;
//MainPageMaster
LaunchMainPageMasterExit_Clicked = new Command(MainPageMasterExit_Clicked);
//ProductListPage
LaunchProductListPageAddButton = new Command(ProductListPageAddButton);
//ProductEditPage
LaunchProductEditPageSaveButton = new Command(ProductEditPageSaveButton);
//ProductAddPage
LaunchProductAddPageSaveButton = new Command(ProductAddPageSaveButton);
//AboutUsPage
LaunchAboutUsPageButton = new Command(AboutUsPageButton);
//ClientInfoPage & SupportPage
LaunchClientInfoPageCallButton = new Command(ClientInfoPageCallButton); LaunchSupportPageClear_Clicked = new Command(SupportPageClear_Clicked); LaunchSupportPageSent_Clicked = new Command(SupportPageSent_Clicked);
//DiscountPage
LaunchDiscountPageAddButton = new Command(DiscountPageAddButton);
//WorkerPage
LaunchWorkerPageAddButton = new Command(WorkerPageAddButton);
//WorkerAddPage
LaunchWorkerAddPageSaveButton = new Command(WorkerAddPageSaveButton);
productListPage = new NavigationPage(new ProductListPage(mainViewModel)); workerPage = new NavigationPage(new WorkerPage(mainViewModel)); clientPage = new NavigationPage(new SupportListPage(mainViewModel)); aboutUsPage = new NavigationPage(new AboutUsPage(mainViewModel)); discountPage = new NavigationPage(new DiscountPage(mainViewModel));
}
}
}
```

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