Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ТАРИФНИМИ ПАКЕТАМИ КОМПАНІЇ МОБІЛЬНОГО ЗВ'ЯЗКУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1
Личагіна С.М.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри_____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Личагіній Світлані Миколаївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для управління тарифними пакетами компанії мобільного зв'язку

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека JavaFX, система управління базами даних Oracle.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналітичний огляд ПО та інструментів розробки десктопних застосунків з можливістю управління тарифними пакетами компаній мобільного зв'язку.
2. Моделювання застосунку та бази даних для управління тарифними пакетами.
3. Програмна реалізація застосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми управління тарифними пакетами, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Розробка бази даних	21.04.23-30.04.23	
5	Моделювання застосунку	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-25.05.23	
7	Оформлення пояснювальної записки	26.05.23-28.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	06.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____ доц. Тітова О.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 68 с., 2 табл., 30 рис., 1 дод., 40 джерел.

УПРАВЛІННЯ ТАРИФНИМИ ПАКЕТАМИ, JAVA FX, ORACLE DB, СТАТИСТИЧНИЙ АНАЛІЗ ТАРИФНОГО ПЛАНУ, ОБРОБКА КОМАНД, РЕЛЯЦІЙНА БАЗА ДАНИХ.

Об’єктом роботи є управління тарифними планами.

Метою роботи є розробка застосунку для швидкого та автономного управління акаунтом абонентів, який дозволяє користувачам аналізувати власний тариф, змінювати його за власними потребами, та управляти рахунком.

Проведено дослідження існуючих платформ та застосунків для управління тарифними планами, а також аналіз засобів для створення застосунків мовою Java. Змодельована модель бази даних, та реалізовано зберігання застосунком даних.

У результаті роботи здійснена програмна реалізація десктоп застосунку для управління тарифними пакетами компанії мобільного зв’язку за допомогою фреймворку JavaFX, та системою управління базами даних Oracle.

MANAGEMENT OF TARIFF PACKAGES, JAVA FX, ORACLE DB, STATISTICAL ANALYSIS OF THE TARIFF PLAN, COMMAND PROCESSING, RELATIONAL DATABASE.

The object of work is the management of tariff plans.

The aim of the work is to develop an application for quick and autonomous management of subscriber accounts, which allows users to analyze their own tariff, change it according to their own needs, and manage their account.

A study of existing platforms and applications for managing tariff plans was conducted, as well as an analysis of tools for creating applications in Java. A database model was modeled and the application's data storage was implemented.

As a result of the work, the software implementation of a desktop application for managing tariff packages of a mobile company using the JavaFX framework and the Oracle database management system was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GUI – Graphical User Interface (графічний інтерфейс користувача)

ПЗ – програмне забезпечення

CSS – Cascading Style Sheets (каскадні таблиці стилів)

API – Application Programming Interface (інтерфейс прикладного програмування)

AWT – Abstract Window Toolkit (абстрактний віконний інтерфейс)

СУБД – система управління базами даних

РБД – реляційна база даних

СКБД – система керування базами даних

БД – бази даних

ВСТУП

В останні роки спостерігається великий прогрес у розвитку мобільного зв’язку. Сьогодні майже кожен має телефон для спілкування з близькими та друзями на відстані. Також великим проривом стало появлення швидкісної передачі даних, 4G та 5G стали поштовхом для появи багатьох месенджерів, які дозволяють обмінюватися фотографіями, відео, аудіо, а також здійснювати відео дзвінки. Саме завдяки цьому люди навіть знаходячись в різних кутках планети можуть побачити один одного та постійно підтримувати зв’язок. Мобільний інтернет зараз становить величезну частину нашого життя: соціальні мережі, месенджери, перекладачі та багато іншого. Зараз у людини є доступ до обширної кількості інформації, яку можна знати в Інтернеті знаходячись будь-де. Таким чином, мобільний зв’язок становить невід’ємну частину життя сучасної людини.

Напрямком даної роботи є розробка застосунку для управління тарифними пакетами для компанії мобільного зв’язку, що дозволить користувачу швидко та власноруч змінювати тариф, поповнювати рахунок, відстежувати використані кошти, СМС, гігабайти інтернету, та інше. Для цього будуть використані система управління базами даних Oracle, а також дані, що зібрано в результаті наукового дослідження з різних джерел.

Актуальність роботи полягає в тому, що застосунок має забезпечити збір інформації про використані хвилини, мегабайти, залишок коштів на рахунку, а також повідомляти користувачів про необхідність поповнення рахунку для продовження користування тарифом.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ УПРАВЛІННЯ ТАРИФНИМИ ПАЕТАМИ КОМПАНІЇ МОБІЛЬНОГО ЗВ'ЯЗКУ

1.1 Огляд платформ для управління тарифними пакетами

Існує багато платформ для самостійного управління тарифними планами. Сьогодні у кожного є телефон для зв'язку з рідними та виходу в мережу у реальному часі в будь-який момент, для цього компанії мобільного зв'язку створюють застосунки для комфортного використання тарифів.

У першій частині цього розділу буде приведено аналіз преваг та недоліків таких застосунків, а також актуальність кожного з них. Саме завдяки аналізу буде зазначено необхідність розробленого застосунку, виділені оригінальні рішення, які будуть використані у роботі. У другій частині розділу буде розглянуто фреймворки та технології для створення десктопних застосунків, та їх порівняння і аналіз.

Задля аналізу були відділені найпопулярніші платформи управління тарифними планами в Україні, такі як Vodafone, Kyivstar, Lifecell, також було виділено маловідомого оператора PeopleNet. Вони усі вирішують проблеми моніторингу власного рахунку, тарифного пакету, доступних гігабайт інтернету, та інші.

1.1.1 Vodafone

Vodafone є однією з найбільших компаній мобільного зв'язку в Україні. Вона нараховує у собі 15,8 млн клієнтів станом на вересень

2022 року. Хоча компанія і постраждала внаслідок війни, проте за кількістю абонентів вона все одно одна з найбільших [1, 2].

Однією з переваг застосунку My Vodafone є наявність бонусів за використання функцій застосунку, таких як поповнення рахунку через прив'язку картки, Apple Pay чи Google Pay. Власне бонуси можна використовувати через застосунок, для поповнення додаткових хвилин або гігабайтів. Також нещодавно було додано знижки в деяких онлайн магазинах-партнерах. Ця функція наразі не є дуже популярною, проте при розширенні способів застосування вона може набути ширшого використання. Також застосунок має внутрішній месенджер, у якому можна звернутися до служби підтримки. Найбільшою перевагою є можливість керувати різними номерами в одному застосунку [3, 4].

Значним недоліком застосунку My Vodafone є періодичні проблеми з підключенням та авторизацією, які спостерігають абоненти, а також обмежений функціонал при замовленні та оплаті послуг онлайн. Також застосунок має розширений функціонал у платній версії, проте як на мене увесь необхідний функціонал доступний у безкоштовній версії.

1.1.2 Kyivstar

Kyivstar є найпопулярнішою компанією мобільного зв'язку в Україні. Вона нараховує в собі близько 24,4 млн абонентів станом на вересень 2022 року, тобто зараз Kyivstar продовжує існувати та надавати величезній кількості користувачів можливість зв'язатися з рідними [5].

Однією з переваг застосунку Kyivstar є наявність зручного графіку для оцінки власних витрат на тарифний план, дзвінки та інше. Він більш наглядний на відміну від подібного від My Vodafone, оскільки дає наглядне коло, розділене по секціям, та інформацію по витратам в кожній секції, кожного місяця. Kyivstar також має систему бонусів подібно з Vodafone,

проте тут вони реалізовані промокодами, які дають знижки при введені комбінації у компаніях-партнерах. Список доступних промокодів є широким, та використовуваним у повсякденному житті майже кожного [6–8].

Загалом система досить зручна та продумана, вона містить у собі все необхідне для комфортного використання, та інтуїтивно зрозуміла. Та в цілому застосунок має широкий функціонал, а також превагою є можливість без дзвінків до контакт-центру керувати підключенням послуг та змінами в тарифі.

1.1.3 Lifecell

Lifecell (в минулому life:)) є третьою за величиною компанією мобільного зв’язку в Україні. Наразі вона обслуговує близько 8,4 млн. абонентів станом на середину 2022 року. У наслідок війни компанія втратила майже 8,5% абонентів, проте вона наразі продовжує набирати клієнтську базу, і згодом вийде на початковий рівень клієнтської бази [9].

Недоліками застосунку My Lifecell є обмежені можливості використання, тобто багато функцій які доступні при розмові з оператором або на сайті неможливо зробити самостійно. Наприклад неможливо редагувати власний акаунт, для цього треба скористуватися сайтом. Ще одним значним недоліком який відмічають користувачі є нестабільність. Застосунок може працювати неправильно, викликати помилки при спробі виконати прості операції [10–12].

Загалом виділити переваги застосунку важко, адже він має необхідний функціонал, подібний до застосунків My Vodafone та Kyivstar, проте My Lifecell має пароль при авторизації, тобто зайти в акаунт можна не тільки за допомогою СМС за номером телефону прив’язаному до акаунту, а ще і за паролем який встановлено при реєстрації абоненту. Також застосунок є

повністю безкоштовним і є можливість підключити додаткові сервіси, по типу «Знайди мене» або «Відновлення контактів».

1.1.4 PeopleNet

PeopleNet був першим в Україні національним оператором мобільного зв’язку, але з грудня 2020 року, він припинив надання послуг мобільного зв’язку у Харківській області, та залишився лише для Дніпропетровській області. На 2023 рік компанія стала інтернет провайдером, та надає швидкісний доступ до інтернету у містах Київ, Дніпро та Харків [13, 14].

Застосунок PeopleNet виявився недоцільним, адже кількість абонентів була невелика, тому компанія пропонує вести обліковий запис через сайт. Застосунок власне має досить обмежений функціонал, який не є автономним, адже велику кількість дій можна зробити лише через сайт або розмову з оператором. PeopleNet має базовий функціонал, по типу поповнення рахунку, контролю тарифу та статистики витрат, проте немає таких функцій як зміна тарифу, відстеження витрат, бонусних рахунків, і подібних. Зараз застосунок не оновлюється через замалу кількість абонентів.

1.1.5 Результати порівняння застосунків

Для порівняння застосунків були обрані як популярні та використовувані кожним застосунки, так і непопулярні. Взагалі таких платформ існує достатньо велика кількість у різних державах. Проаналізувавши українські застосунки My Vodafone, Kyivstar, My Lifecell та PeopleNet було виявлено деякі спільні проблеми (табл. 1.1).

Таблиця 1.1 – Проблематика розглянутих застосунків

Застосунок	**Функціонал**	**Статистика**	**Інтерфейс**
My Vodafone	Нестабільна робота застосунку; деякі функції доступні лише у платній версії.	Постійно оновлюється	Не дуже зрозуміле налаштування застосунку; часті проблеми з авторизацією.
Kyivstar	Обмежені можливості налаштування застосунку	Постійно оновлюється	Не дуже зрозуміле налаштування застосунку; низька швидкість оновлення сторінок.
My Lifecell	Нестабільна робота застосунку; обмежені можливості редагування аккаунту.	Постійно оновлюється	Повільна робота застосунку; помилки при виконанні деяких операцій;
PeopleNet	Обмежений функціонал; обмежені області використання мобільного оператору.	Не оновлюється	Неможливість налаштувати застосунок; не дуже зрозумілий інтерфейс; низька швидкість роботи.

Таким чином можна побачити, що усі застосунки мають обмежені можливості налаштування, а також у більшості застосунків виникають помилки при виконанні деяких операцій, в деяких платформах є проблеми з інтерфейсом, його складно зрозуміти одразу. Також можна відмітити, що тільки у застосунку My Vodafone є можливість додати декілька номерів.

1.2 Аналіз засобів створення десктоп застосунків мовою Java

У цій частині розділу буде наведено порівняння відомих фреймворків для створення десктоп застосунків мовою Java. Було виділено такі фреймворки та бібліотеки, як JavaFX, Qt Jambi, AWT, Swing.

1.2.1 JavaFX

JavaFX – це фреймворк для розробки графічних інтерфейсів користувача (GUI) у Java. Він містить набір бібліотек та інструментів, що дозволяють розробникам створювати багатофункціональні та зручні для використання графічні інтерфейси. JavaFX працює на різних платформах, включаючи Windows, macOS та Linux, тому він є універсальним рішенням для створення десктопних застосунків, тобто значною перевагою є те, що застосунки виглядають та працюють однаково на всіх платформах та пристроях [15].

JavaFX включає в себе різноманітні компоненти, такі як кнопки, тексти, таблиці, діалогові вікна тощо, що значно спрощує розробку інтерфейсу користувача. Також, фреймворк має вбудовану підтримку 2D та 3D-графіки, що дозволяє створювати вражаючі візуальні ефекти та анімацію. JavaFX є стандартом для розробки графічних інтерфейсів у Java починаючи з версії Java 8. Однак, у новіших версіях Java він більше не входить до складу стандартної бібліотеки Java, але все ще підтримується та розвивається [15–18].

Цікавою особливістю застосунків написаних за допомогою фреймворку JavaFX є те, що їх можна налаштовувати за допомогою каскадних таблиць стилів, тобто за допомогою CSS (Cascading Style Sheets). Цей підхід дозволяє відокремити інтерфейс користувача від реалізації, тому

робота над інтерфейсом може бути здійснена за допомогою графічних дизайнерів. Архітектуру платформи JavaFX наведено на рисунку 1.1 [15].

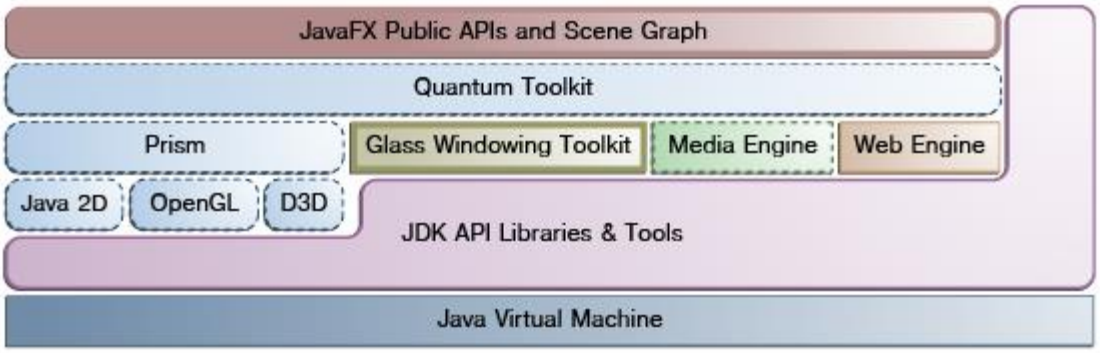

Рисунок 1.1 – Архітектура платформи JavaFX

1.2.2 Qt Jambi

Qt Jambi – це крос-платформний інструментарій для розробки мовою Java, який основується на бібліотеці Qt. Він надає можливості для створення GUI, а також доступ до багатьох інших функціональних можливостей, таких як робота з мережею, базами даних, різноманітними форматами даних, та інше. Однією з головних переваг Qt Jambi була можливість створення застосунків, які працювали на різних операційних системах без необхідності переписувати код для кожної з них [19].

Недоліком інструментарію є те, що він зараз не розвивається та не підтримується, хоча його код все ще у відкритому доступі. Тож, на відміну від JavaFX, який продовжує оновлюватись, Qt Jambi не має підтримки, а тому і не є таким актуальним зараз у нових проєктах. Натомість Qt підтримує розробку крос-платформних ПЗ на мові програмування C++ [20].

1.2.3 AWT

AWT (Abstract Window Toolkit) – це API для створення графічних програм на Java. Цей фреймворк є платформа-залежним, тобто компоненти не однакові на всіх платформах. Відповідно до вигляду платформи, вигляд та поведінка компонентів також змінюється. Саме через залежність від вигляду платформи та відносно довгий час виконання, AWT сьогодні мало коли використовується для розробки ПЗ. Для вирішення цієї проблеми було створено фреймворк Swing, який має більш гнучкі та потужні компоненти, та базується на AWT.

AWT надає базовий набір для створення GUI, таких як кнопки, текстові поля, списки, таблиці, тощо. Схему ієрархії абстрактних віконних інструментів наведено на рисунку 1.2 [21].

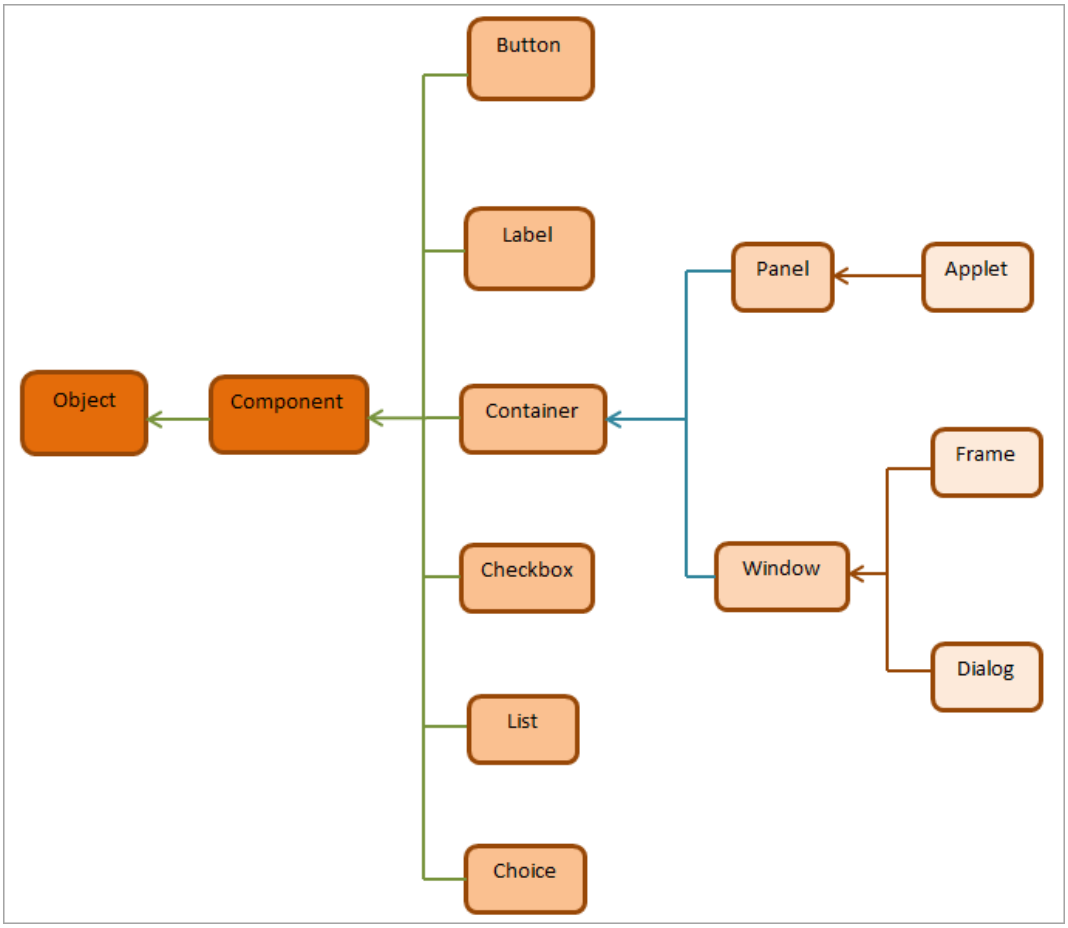

Рисунок 1.2 – Ієрархія AWT в Java

1.2.4 Swing

Swing – це інструментарій для створення GUI мовою програмування Java. Він є частиною бібліотеки базових класів Java, та використовує AWT для відображення графічних елементів на екрані. Завдяки динамічним модулям, до компоненту можна підключити інші, специфічні для даної операції вигляд та поведінку. Основним недоліком є відносна повільна робота ПЗ, хоча зараз, через зростання потужності персональних комп'ютерів ця повільність маже не відчувається [22–24].

Фреймворк Swing побудований поверх фреймворку AWT, саме тому їх архітектура майже однакова, різниця лише в тому, що компоненти Swing мають невелику вагу, та не залежать від платформи. Крім того Swing надає набір розширених елементів управління, таких як дерева, елементи керування таблицями, тощо.

Схема ієрархії класів наведена на рисунку 1.3 [23].

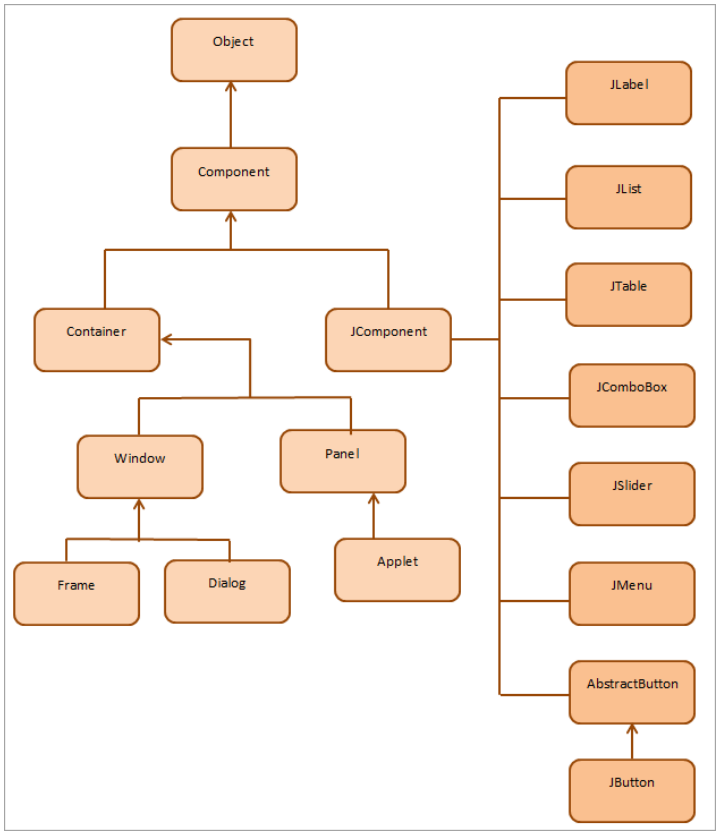

Рисунок 1.3 – Ієрархія Swing API в Java

1.3 Постановка задачі

Після проведення існуючих застосунків в області управління тарифними пакетами, та існуючі засоби для створення десктопних застосунків мовою програмування Java, можна зробити висновок по актуальність проблеми, та висунути ряд технічних та функціональних вимог щодо прототипу застосунку [22].

Об’єктом роботи є управління тарифними планами.

Метою даної кваліфікаційної роботи є розробка застосунку для швидкого та автономного управління акаунтом абонентів, який дозволяє користувачам аналізувати власний тариф, змінювати його за власними потребами, та управляти рахунком.

Застосунок буде призначений для використання абонентів в особистих цілях. Необхідно забезпечити можливість моніторингу власного акаунту у будь-який момент. Наприклад, абонент за власним бажанням може змінити тариф на більш вигідний, поповнити рахунок, або побачити статистику тарифного пакету за поточний місяць.

Для досягнення мети необхідно вирішити такі завдання щодо розробки прототипу застосунку:

– створення зручного та зрозумілого інтерфейсу;

– забезпечення можливості зміни тарифного пакету;

– забезпечення можливості поповнення рахунку;

– реалізація бази даних для збереження інформації;

– забезпечення відображення статистики пакету абонента за поточний місяць;

– забезпечення відображення історії поповнення тарифу;

– забезпечення можливості запам’ятовування абоненту, вхід до застосунку без повторного введення даних.

2 МЕТОДИ І ЗАСОБИ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ МОБІЛЬНОГО ОПРАТОРА

2.1 Визначення характеристик засобів підтримки бази даних

База даних є важливою складовою багатьох програмних продуктів, включаючи мобільні застосунки. Для забезпечення надійності, ефективності та безпеки роботи з базою даних, необхідно визначити правильні засоби її підтримки.

Одним з найпопулярніших засобів для підтримки баз даних є системи управління базами даних (СУБД), такі як MySQL, Oracle, PostgreSQL та інші. Вони забезпечують надійне зберігання даних, можливість доступу до них, а також забезпечують механізми захисту даних від несанкціонованого доступу.

Іншими важливими засобами підтримки баз даних є резервні копії та моніторинг. Резервні копії забезпечують збереження даних в разі виникнення непередбачуваних ситуацій, таких як збій апаратного забезпечення або видалення даних в результаті помилки. Моніторинг, зі свого боку, забезпечує контроль за роботою бази даних, її навантаженням та ресурсами, які вона використовує [25].

Крім того, важливо забезпечити безпеку даних в базі даних, що може бути досягнуто за допомогою різних методів, таких як шифрування даних, обмеження доступу до даних та моніторинг активності користувачів.

Визначення правильних засобів підтримки бази даних є критично важливим для забезпечення надійності та ефективності роботи програмного продукту. Це може бути досягнуто за допомогою використання СУБД, резервних копій, моніторингу, масштабування, індексування даних, а також методів забезпечення безпеки даних.

2.1.1Вибір моделі даних

Реляційна модель даних є однією з найпоширеніших моделей. Вона базується на використанні таблиць, де кожен запис відповідає одному рядку, а кожен стовпець – одному полю. В таблицях можна зберігати дані різних типів, таких як числа, текст, дата та інші. Реляційна модель даних є добре підходом для зберігання структурованих даних, таких як дані про клієнтів, замовлення, фінансову інформацію тощо [24–26].

Реляційна модель є множиною відношень, яка є підмножиною декартового добутку списку доменів. Домен – це множина значень, з якої вилучаються значення для даного атрибута. Ця модель полягає в тому, що база даних складається з простих таблиць, які задовольняють певним обмеженням та можуть розглядатись як математичні відношення. Рядки таких таблиць називаються кортежами, а назви стовпців – атрибутами. Відношення (таблиці) мають декілька атрибутів, які однозначно ідентифікують кортежі та називаються ключами. Відмінність реляційної моделі полягає в тому, що у відносинах (таблицях) реальні об’єкти та зв’язки між ними представлені у базі даних в однорідному, в нормалізованому вигляді, на відміну від сіткової та ієрархічної моделей.

Відношення R, що визначено на сукупності n множин $D_1,...D_n$ є множина кортежів $d_1,...,d_n$ таких, що $d_1 \in D_1, d_2 \in D_2,..., d_n \in D_n$. Множини $D_1,...D_n$ називаються доменами відношення R. Величина n називається ступінню відношення R. Відношення степені n будемо називати n-ним [24].

Отже, домен – деяка множина, значення з котрого може приймати будь-який атрибут конкретної предметної області.

Кожній таблиці надається ім’я, а також кожен стовпець в таблиці повинен мати своє ім’я. Елементи таблиці повинні бути недільними (атомарними). Кожен рядок (кортеж) таблиці повинен мати унікальний ідентифікатор (ключ), який може бути одним з елементів запису. Основний

ключ – це один або декілька атрибутів, які однозначно ідентифікують запис у таблиці. Простий ключ складається з одного атомарного атрибута, значення якого є унікальним, а складений ключ складається з двох або більше атрибутів. Декартовим добутком доменів $D_1, D_2, ..., D_n$ називається множина всіх кортежів довжини *k*, таких, що $d_1 \in D_1, d_2 \in D_2, ..., d_n \in D_n$:

$$D_1 \times D_2 \times ... \times D_n = \{< d_1, d_2, ..., d_n > | \, d_i \in D_i, i = 1...n\}.$$

Отже, відношення R – є таке відношення на множинах $D_1, D_2, ..., D_n$, якщо R є підмножиною декартова добутку $D_1, D_2, ..., D_n$.

Кожен домен має ім'я. Множина Nat – множина імен та атрибутів та $Nrel$ – множина імен відношень.

Для будь-якого $A \in Nat$ визначена множина значень цього атрибута, яке має співпадати з одним із доменів $D_1, D_2, ..., D_k$. При цьому, вважається заданою функція привласнювання множини значень

$$Dom: Nat \rightarrow \{D_1, ..., D_k\}.$$

Отже, пара $< A, Dom, A >$, де $A \in \text{Nat}$, називається атрибутом де A – це ім'я та $Dom\, A$ – область значень.

Список імен атрибутів відношення $R(A_1, ..., A_n)$, де $R \in Nel$, $A_1, ..., A_n \in Nat$ називається схемою відношення з іменем R. Множина схем відношень називається системою бази даних, а поточне значення – базою даних.

Реляційна база даних (РБД) є однією з найпоширеніших і найбільш використовуваних типів баз даних у світі. РБД має свої переваги та недоліки, які слід розглянути перед використанням [25, 26].

Переваги реляційної бази даних:

– структура даних. Реляційна база даних забезпечує чітку структуру даних і забезпечує можливість розширення бази даних без необхідності вносити зміни в код застосунків, що використовують ці дані;

– універсальність. Реляційна база даних підтримує багато мов запитів, таких як SQL, який підтримується більшістю відомих СУБД. Це робить РБД більш універсальною та зручною для використання;

– зручність для аналізу даних. РБД дозволяє використовувати різні методи для аналізу даних, включаючи звіти та запити. Це забезпечує зручний інтерфейс для користувачів для отримання необхідної інформації;

– легкість у використанні. РБД може бути дуже простою у використанні, якщо база даних не має складної структури;

– безпека. РБД забезпечує можливість управління доступом до даних і забезпечення їх конфіденційності.

Недоліки реляційної бази даних:

– складність. Реляційна база даних може бути дуже складною в розробці, якщо структура даних недостатньо чітко визначена. Надійне визначення структури даних може зайняти багато часу і зусиль;

– «Перформанс». При використанні складних запитів, РБД може працювати повільно та неефективно. Це може бути особливо важливо для великих баз даних з багатою кількістю даних;

– строгість типів даних. РБД вимагає, щоб типи даних були строго визначені. Це може бути проблемою для деяких застосунків, де дані можуть бути неоднозначні;

– обмеження. Реляційна база даних може мати обмеження на кількість записів, яку вона може зберігати або на кількість користувачів, які можуть одночасно працювати з базою даних;

– не підходить для всіх типів даних. Реляційна база даних не підходить для всіх типів даних, таких як великі об’єми неструктурованих

даних, таких як відео або зображення, які можуть бути збережені в інших типах баз даних, наприклад, NoSQL.

Узагальнюючи, реляційна база даних має свої переваги та недоліки. Перед тим, як обрати РБД для своєї системи, було ретельно вивчено всі аспекти, пов’язані з її використанням.

2.1.2 Вибір концептуальної моделі

Концептуальна модель даних «сутність-зв’язок» (Entity-Relationship Model) є ефективним інструментом проектування баз даних для багатьох типів застосунків.

Сутність – це об’єкт, про який ведеться інформація, такий як користувач, товар, замовлення. Кожна сутність має свій унікальний ідентифікатор (ключ), який відрізняє її від інших сутностей.

Зв’язок – це відношення між двома або більше сутностями. Він визначає, які сутності взаємодіють між собою та які атрибути мають зв’язані сутності. Зв’язок може бути однонаправленим або більшим, в залежності від того, чи можуть зв’язані сутності взаємодіяти в обидва способи.

Модель сутність-зв’язок допомагає створити логічну структуру бази даних, яка відображає бізнес-логіку системи, що дозволяє легко зрозуміти взаємодію між сутностями. Вона є ефективним інструментом для аналізу і проектування баз даних.

У випадку з застосунком мобільного оператора та керування тарифами, модель сутність-зв’язок є найбільш підходящою з кількох причин:

– простота використання та зрозумілість. Модель «сутність-зв’язок» дозволяє легко відобразити відносини між різними об’єктами в застосунку. Це дозволяє розробникам та архітекторам баз даних легко зрозуміти взаємодії між об’єктами і відображати їх у базі даних;

– простота розширення. Модель «сутність-зв’язок» легко розширюється для включення нових сутностей та зв’язків між ними. Це дозволяє розробникам застосунків легко додавати нові функції та можливості до своїх застосунків;

– розуміння бізнес-логіки. Модель «сутність-зв’язок» дозволяє легко відобразити бізнес-логіку застосунку, оскільки вона заснована на концепції сутностей, що відображають різні аспекти бізнес-даних. Це дозволяє забезпечити точність та якість даних в базі даних та зменшити можливість помилок у застосунку;

– незалежність від конкретної СУБД. Модель «сутність-зв’язок» не залежить від конкретної системи управління базами даних (СУБД). Це дозволяє розробникам застосунків використовувати будь-яку СУБД за власним вибором, що дозволяє забезпечити більшу гнучкість та простоту в розробці та підтримці застосунку.

Одною з основних переваг концептуальної моделі даних «сутність-зв’язок» є те, що вона дає можливість описувати більш складні зв’язки між даними, ніж реляційна модель даних. Наприклад, в моделі «сутність-зв’язок» можна легко описати «багато-до-багатьох» відносини та інші складні залежності між даними. В реляційній моделі для опису таких залежностей доводиться використовувати складні запити та підзапити, що може призводити до складнощів при розробці та оптимізації роботи з БД.

Крім того, модель «сутність-зв’язок» дозволяє зосередитися на логіці застосунку та бізнес-процесах, що збільшує зрозумілість моделі даних для всіх учасників проєкту. Кожна сутність та її взаємозв’язки можуть бути легко визначені та проаналізовані окремо, що дозволяє зосередитися на деталях та допомагає забезпечити високу якість проєкту [27].

Однак, модель «сутність-зв’язок» має деякі недоліки. Одним з них є складність перетворення моделі на конкретну фізичну базу даних, що може зайняти багато часу та зусиль. Крім того, модель сутність-зв’язок є менш стандартизованою, ніж реляційна модель даних, що може призвести до

труднощів у спілкуванні з іншими командами розробників та підтримкою продукту в майбутньому, проте у даному випадку, така проблема виключена, а отже «сутність-зв’язок» є добрим варіантом для застосунків, які мають складні взаємозв’язки між даними і розробляються невеликими командами.

2.2 Розробка концептуальної моделі бази даних

Таблиця «clients» є ключовою, оскільки вона містить основну інформацію про клієнтів, що входять до системи мобільного оператора, включаючи їх ідентифікатори та тарифи, які вони обрали. Інші таблиці мають зв’язок з цією таблицею через зовнішні ключі, що дозволяє відслідковувати історію використання послуг та опцій, які були замовлені клієнтом, та керувати їхніми балансами і тарифами. Сутності та атрибути концептуальної моделі бази даних показано у таблиці 2.1.

Таблиця 2.1 – Сутності та атрибути

Сутність	**Атрибути**	**PK**
1	**2**	**3**
Клієнт	№Клієнта, логін, пароль, номер телефону, №тарифу, баланс, дата початку використання тарифу, дата «сплачено до», залишок гігабайт, залишок хвилин на інші оператори, залишок хвилин за кордон.	№Клієнта
Тариф	№Тарифу, назва, вартість, кількість гігабайт, кількість хвилин на інші оператори, кількість хвилин за кордон.	№Тарифу
Історія Тарифів	№Клієнта, №Тарифу, дата початку використання, дата кінця використання.	(№Клієнта, №Тарифу)
Додаткові опції	№Опції, назва, описання, значення, ціна.	№Опції

Продовження таблиці 2.1

1	2	3
Додаткові опції	№Опції, назва, описання, значення, ціна.	№Опції
Історія поповнень	№Поповнення, №Клієнта, сума, дата поповнення.	№Поповнення
Клієнт-опція	№Клієнта, №Опції, дата «сплачено до», залишок.	(№Клієнта, №Опції)

Виконавши аналіз сутностей та їх зв’язків – побудуємо UML-діаграму (рис. 2.1) зв’язків сутностей засобами інтернет-ресурсу [28].

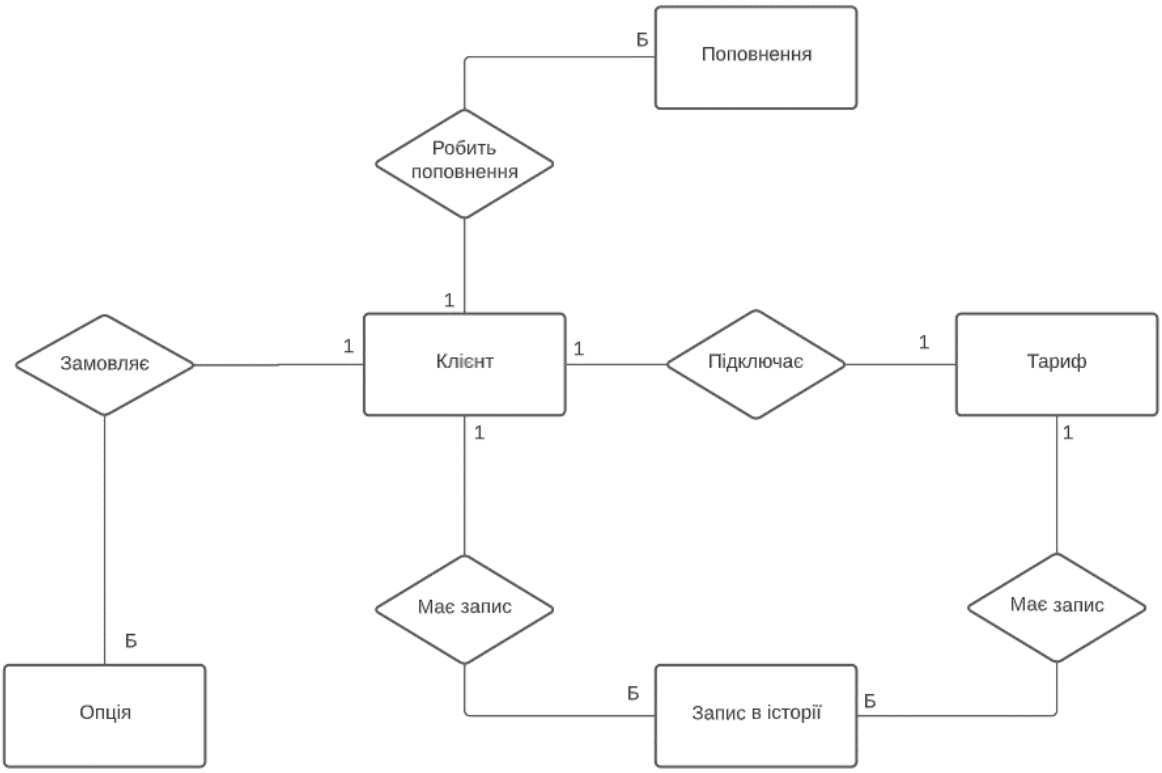

Рисунок 2.1 – UML-діаграма зв’язків сутностей предметної області «Мобільний оператор»

2.3 Розробка структури БД

Управління структурою бази даних (БД) та її інформаційною моделлю є важливим етапом у розробці та підтримці БД. Інформаційна модель БД описує структуру даних у вигляді сутностей та їх взаємозв’язків, що дозволяє розуміти, як дані зберігаються та взаємодіють між собою. Управління структурою БД включає в себе такі етапи, як проектування БД, створення БД, зміна та модифікація її структури [26, 27].

Один з найважливіших етапів проектування БД – це розробка інформаційної моделі. Інформаційна модель повинна відображати всі сутності та їх взаємозв’язки, що використовуються в БД. Вона повинна бути якомога простішою та зрозумілою для всіх користувачів, що займаються розробкою та підтримкою БД.

В даній базі даних реалізовано систему керування тарифними планами для клієнтів телекомунікаційної компанії. База даних складається з 6 таблиць:

– таблиця «tariffs» містить інформацію про тарифні плани, які пропонуються компанією. Кожен тарифний план має унікальний ідентифікатор, назву, вартість, кількість гігабайтів, кількість хвилин на вихідних та кількість хвилин на міжнародних дзвінках;

– таблиця «clients» містить інформацію про клієнтів компанії. Кожен клієнт має унікальний ідентифікатор, логін, пароль, номер телефону, посилання на тарифний план, баланс, дату початку тарифного плану, дату оплати тарифного плану, кількість гігабайтів, кількість хвилин на вихідних та кількість хвилин на міжнародних дзвінках, які залишилися на рахунку;

– таблиця «usage_history» містить інформацію про використання тарифних планів. Кожен запис має унікальний ідентифікатор, посилання на клієнта, посилання на тарифний план, дату початку та дату закінчення періоду використання;

– таблиця «options» містить інформацію про додаткові опції, які можуть придбати клієнти. Кожна опція має унікальний ідентифікатор, назву, опис, значення опції та вартість;

– таблиця «Client_Option» містить інформацію про опції, які підключені до конкретного клієнта. Кожен запис має посилання на клієнта, посилання на опцію, дату до якої сплачено опцію та значення опції, яке залишилося на рахунку клієнта;

– таблиця «Top_Ups» містить інформацію про поповнення балансу клієнтів. Кожне поповнення має унікальний ідентифікатор запису про поповнення, який генерується автоматично за допомогою функції GENERATED ALWAYS AS IDENTITY, ідентифікатор клієнта, який здійснив поповнення балансу, дата поповнення балансу, сума поповнення балансу.

Після проектування БД, необхідно створити її за допомогою відповідної системи управління базами даних (СУБД). В процесі створення БД можна виконувати різні налаштування, наприклад, встановити рівень безпеки БД, змінити розмір файлів БД, створити резервні копії тощо.

У якості СУБД, як було вказано раніше, було обрано СУБД Oracle, оскілки на мові програмування Java виконати під’єднання до БД можна за допомогою одного класу та бібліотеки ORACLE JDBC.

Кілька причин, чому краще використовувати саме СУБД Oracle:

– відмінна продуктивність: Oracle є однією з найшвидших та найбільш продуктивних СУБД, що дозволяє забезпечити швидкий доступ до даних та високу продуктивність в операціях з базами даних;

– висока надійність та стійкість: Oracle є дуже надійною та стійкою СУБД. Вона може забезпечити відмінний рівень захисту даних та забезпечити максимальну доступність бази даних;

– розширюваність та масштабованість: Oracle є дуже розширюваною та масштабованою СУБД, що дозволяє використовувати її для будь-яких проєктів, незалежно від їх розміру та складності;

– багатий функціонал та можливості: Oracle має величезний набір функцій та можливостей, що дозволяє розробникам створювати складні та потужні застосунки з високим рівнем функціональності;

– підтримка та документація: Oracle має велику та активну спільноту користувачів та розробників, що надає високий рівень підтримки та документації. Відкритий код дозволяє досліджувати та змінювати функціонал СУБД, якщо це необхідно.

Саме ці причини роблять Oracle однією з кращих СУБД для Java-проєктів, де потрібна висока продуктивність, надійність та масштабованість.

За допомогою інтегрованого середовища розробки SQL DEVELOPER та мови програмування SQL було створено таблиці бази даних.

ER-діаграма сутностей наведена у рисунку 2.2.

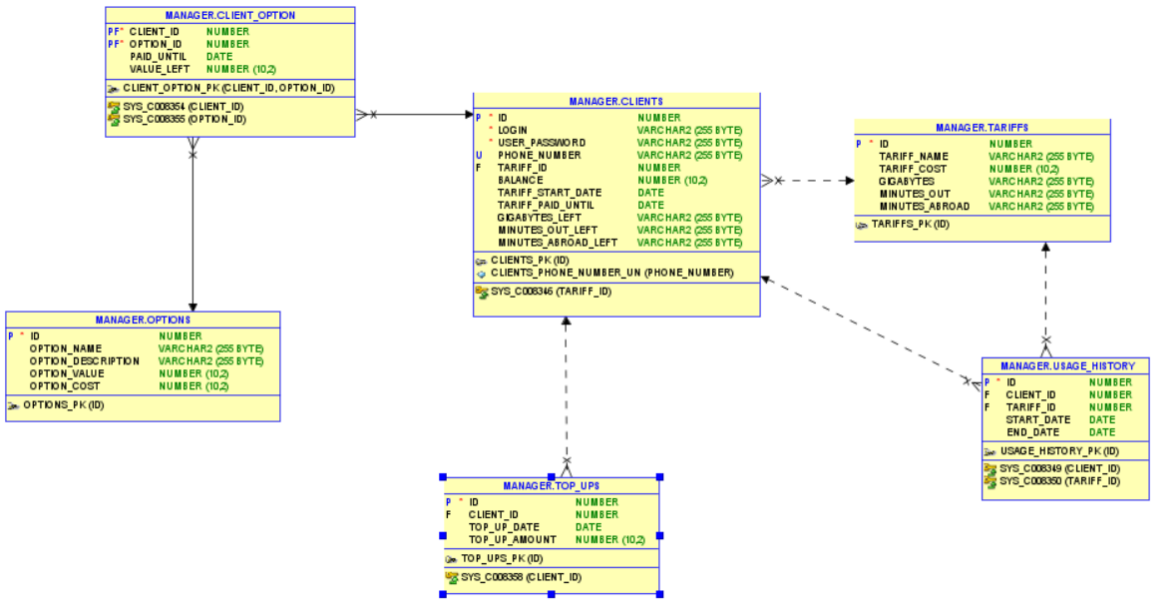

Рисунок 2.2 – ER-діаграма сутностей

2.4 Аналіз алгоритму роботи застосунку

Алгоритм роботи застосунку та зв’язку із БД можна поділити на підзадачі:

– підключення до БД. Для того, щоб застосунок міг працювати з БД, необхідно спочатку встановити з’єднання. Для цього зазвичай використовують спеціальні бібліотеки, такі як JDBC. У застосунку використовуються конфігураційні файли, в яких зберігаються дані для підключення до БД (наприклад, URL, ім’я користувача та пароль). Для підключення до БД використовується бібліотека Oracle JDBC;

– виконання запитів до БД. Після успішного підключення до БД, застосунок може виконувати запити до БД. Запити можуть бути виконані як на вибірку даних, так і на їх оновлення. Для цього використовуються SQL-запити, що формуються в програмі і виконуються на стороні сервера БД. Результат запиту може бути отриманий у вигляді об’єктів, що відповідають структурі БД, або у вигляді рядків з даними;

– обмін даними між застосунком і БД. Застосунок може обмінюватися даними з БД в різних форматах. Наприклад, можна використовувати XML-формат для збереження та передачі даних. В такому випадку, застосунок може генерувати XML-файли з даними, а БД може зберігати ці файли як BLOB-об’єкти. Також, для мови програмування Java, можна використовувати для передачі даних між застосунком та БД інтерфейс ResultSet, який працює напряму із БД у режимі реального часу. Використовуючи цей інтерфейс можна одночасно і отримувати дані і їх модифікувати.

Виходячи з отриманих даних при ініціалізації застосунку, на екрані зображується повна інформація про акаунт клієнта, від імені котрого було здійснено авторизацію.

Усі дії користувача миттєво відображаються у БД. Будь-які перебої у роботі машини користувача, «виліт» програми та інші несправності не впливають на дані, що зберігаються в базі даних. На машині користувача може зберігатися лише інформація, щодо його авторизації (якщо користувач при авторизації відмітив «Remember me» – дані авторизації будуть збережені та завантажені при майбутній ініціалізації застосунку у файлі data.txt директорії src/main/resources/com/nure/tariffmanager/LocalData/data.txt).

Стосовно безпеки, було прийняте рішення, зберігати дані щодо авторизаційних даних у БД у вигляді хешу.

Хеш – це криптографічний термін, яким позначаються дані, отримані в результаті пропуску вихідної інформації через хеш-функцію. Крім того, цей результат може також називатися хеш-значенням, хеш-кодом або дайджестом.

Хеш-функції – це певні математичні алгоритми, що перетворюють будь-яку інформацію в хеш фіксованого розміру (довжини). Найчастіше, використовуються комбінації з шістнадцяти символів, в яких застосовуються цифри від 0 до 9 і букви від A до F. Застосований алфавіт, зрозуміло – виключно латиниця [29].

Хеш-функції мають багато застосувань, але найбільш поширеним є використання хеш-функцій для зберігання та перевірки цілісності даних. Наприклад, у більшості веб-сайтів паролі користувачів не зберігаються у відкритому вигляді, а зберігаються в базі даних у вигляді хеш-значень. Якщо зловмисник отримує доступ до бази даних, він не зможе отримати паролі відразу, оскільки вони зберігаються у вигляді хеш-значень. В залежності від вибраного алгоритму хешування, зловмисник може спробувати зламати паролі шляхом «брутфорсу» (спроба всіх можливих комбінацій), але це займе надзвичайно багато часу [30].

Інший приклад застосування хеш-функцій полягає в забезпеченні цілісності даних, які передаються по мережі. Хеш можна обчислити на

даних, які передаються, і передавати його разом з даними. Отримувач може також обчислити хеш на даних, які він отримав, і порівняти його з хешем, який було передано разом з даними. Якщо хеші співпадають, можна стверджувати, що дані були передані без помилок або внесені якісь зміни в самі дані.

Отже, з метою безпеки, дані у БД зберігаються у вигляді хешу, а пароль знає лише користувач.

У випадку, якщо користувач загубив пароль, він може натиснути кнопку «Забули пароль», після чого програма згенерує новий пароль, внесе його хеш до БД, а сам пароль буде відправлений в SMS повідомленні за допомогою SMS API інтернет-ресурсу [31].

Проте, оскільки застосунок є прототипом, та усі номери клієнтів було згенеровано автоматично, а повідомлення має бути відправлено на реальний номер і тестування функції викликало зайві фінансові витрати – було прийняте рішення видалити цю функцію і залишити у застосунку формальне повідомлення про те, що новий пароль було відправлено в SMS.

2.5 Переваги та недоліки обраної конфігурації системи

Основним недоліком продукту є те що, усі перевірки тарифів та опцій на дійсність та внутрішні обчислення виконуються на стороні клієнта, що є умовно невірним. Для поточної реалізації такий підхід є прийнятним, оскільки користувач бачить достовірні дані про стан рахунків, а дані у БД змінюються перед ініціалізацією графічного інтерфейсу, проте у випадку справжнього оператора – обчислення мають відбуватися на сервері, до якого підключається застосунок під час ініціалізації [32].

Застосунок було збудовано таким чином, щоб відокремити серверні функції від функцій користувацького застосунку. У випадку необхідності

перетворення застосунку у повноцінний програмний продукт масового використання – програмний код окремих функцій переноситься на сервер, який виконує усі оновлення в режимі реального часу.

Головна перевага такої реалізації полягає в тому, що для обчислень використовується одна машина, усі функції працюють із мінімальними затратами пам’яті та часу процесора, тим не менш надають користувачу достовірну інформацію стосовно його балансів та стану рахунку.

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Огляд інструментів розробки

У великому і швидкозростаючому світі програмного забезпечення, належне використання потужних інструментів розробки вирішально важливе. Правильний вибір інструментарію може визначити успіх проєкту та його майбутнє розширення. У цьому розділі буде розглянуто ключові інструменти розробки, використані для створення застосунку для керування тарифними планами оператору мобільного зв'язку [32–34]:

– JavaFX: JavaFX є потужною багатоплатформовою бібліотекою інтерфейсу користувача для розробки багатофункціональних застосунків. Вона надає засоби для створення сучасних і зручних графічних інтерфейсів з використанням Java. JavaFX пропонує широкий набір готових елементів управління, анімаційних можливостей та підтримку CSS для налаштування зовнішнього вигляду застосунку. Використання JavaFX в проєкті дозволило забезпечити користувачам зручний і привабливий інтерфейс для керування тарифними планами;

– Oracle Database (OracleDB): OracleDB є однією з найпотужніших реляційних систем керування базами даних (СКБД) на ринку. Вона забезпечує надійне та ефективне зберігання даних, а також надає широкий набір функцій для оптимізації роботи з базами даних. OracleDB пропонує мову запитів SQL для роботи з даними та розширені можливості управління даними, такі як транзакції, індексація, забезпечення цілісності даних та багато іншого. Використання OracleDB в проєкті дозволило створити потужну та масштабовану базу даних для зберігання і керування тарифними планами, а також пов'язаною інформацією про абонентів та їх послуги;

– SQLDeveloper: SQLDeveloper є інтегрованим середовищем розробки для роботи з базами даних Oracle. Він надає зручний інтерфейс для створення, редагування та виконання SQL-запитів, а також для адміністрування бази даних. SQLDeveloper має вбудовані засоби для моделювання бази даних, налагодження запитів та аналізу продуктивності. Використання SQLDeveloper допомогло в розробці та налагодженні запитів для отримання, збереження та оновлення даних у базі даних OracleDB.

Ці інструменти разом створюють потужну та ефективну платформу для розробки застосунку, що дозволяє керувати тарифними планами оператора стільникової мережі (далі TariffManage). JavaFX дозволяє розробити зручний та привабливий інтерфейс користувача, в той час як OracleDB та SQLDeveloper надають потужні засоби для зберігання, керування та обробки даних. Завдяки цим інструментам, було створено застосунок, який забезпечив потужні інструменти для керування тарифними планами та надання якісних послуг абонентам.

3.2 Розробка бази даних

База даних – це певний набір даних, які пов’язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом.

Об’єднання великої кількості даних в єдину базу дає змогу для формування безлічі варіації групування інформації – особисті дані клієнта, історія замовлень, каталог товарів та будь-що інше.

Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв’язок

інформації: зміна в одному рядку може спричинити зміни в інших рядках – це допомагає працювати з інформацією простіше і швидше.

Щоб створити запит до бази даних звичайно використовують Structured Query Language (SQL). SQL дає змогу додавати, редагувати та видаляти інформацію, що міститься у таблицях [33].

Приклад SQL запиту, що повертає кортежі із атрибутами назва тарифу, початок використання, кінець використання, де id клієнта = 1 наведено у лістингу 3.1.

Лістинг 3.1 SQL запит історії тарифів окремого користувача:

```
SELECT T.TARIFF_NAME, U.START_DATE, U.END_DATE
FROM Usage_History U INNER JOIN Tariffs T ON U.tariff_id = T.id
WHERE U.client_id=1
```

Загальна структура SQL запиту виглядає наступним чином.

Лістинг 3.2 Структура SQL-запиту:

```
SELECT [DISTINCT | ALL] select_expression
FROM table_references
[WHERE where_definition]
[GROUP BY {unsigned_integer | col_name | formula}]
[HAVING where_definition]
[ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]
```

Опис структури SQL-запиту:

– SELECT – поля, або розрахунки, які необхідно отримати в результаті роботи запиту. DISTINCT – використовується для видалення дублікатів (за замовчуванням – ALL);

– FROM – таблиці, з яких необхідно отримати вищевказані поля. Необхідно вказувати хоча б одну таблицю (або більше з використанням операцій «JOIN»);

– WHERE – умова, яка має бути виконана для записів, які будуть обрані у SELECT. Якщо умова не вказана – будуть обрані усі записи;

– GROUP BY – використовується для групування результатів за агрегатними функціями (MAX, SUM, AVG, тощо);

– HAVING використовується лише у парі із GROUP BY (можна сказати, що HAVING – це WHERE для результатів виконання агрегатних функцій);

– ORDER BY – використовується для сортування записів.

Мова SQL ділиться на 4 підмови (DDL, DML, DCL, TCL):

– DDL – (Data Defenition Language) запити визначення структури бази даних чи схеми;

– DML – (Data Manipulation Language) запити для управління даними;

– DCL – Data Control Language. Видача прав користувачам бази даних, тощо (GRANT, REVOKE);

– TCL – (Transaction Control) запити, що використовуються для управління змінами, зробленими пропозиціями DML;

– для отримання інформації із таблиць (лістинг 3.2) використовується підмова DML.

3.2.1 Обґрунтування вибору СКБД

Oracle Database (OracleDB) є однією з провідних систем керування базами даних (СКБД) на ринку, але перед прийняттям рішення про вибір даної СКБД для розробки застосунку TariffManage, слід провести порівняльний аналіз з іншими популярними СКБД. У цьому підрозділі буде

розглянуто декілька ключових аспектів та порівняно OracleDB з іншими СКБД, а також аргументовано вибір OracleDB для поточного проєкту:

– надійність та масштабованість: OracleDB відома своєю високою надійністю та масштабованістю. Вона добре підтримує великі обсяги даних і високі навантаження. Механізми резервного копіювання, відновлення та відмово-стійкості, присутні в OracleDB, забезпечують надійну роботу бази даних навіть у випадку аварій. Крім того, OracleDB має розподілену архітектуру, яка дозволяє легко масштабувати систему за потребами проєкту;

– функціональність: OracleDB пропонує широкий набір функціональних можливостей, які включають оптимізацію запитів, роботу з транзакціями, розподілені операції, кластеризацію, забезпечення цілісності даних, безпеку та багато іншого. Вона підтримує мову запитів SQL, що є стандартом для реляційних баз даних, а також має власну мову PL/SQL для написання більш складних процедур та функцій. Функціональність OracleDB дозволяє ефективно працювати з складними запитами та операціями над даними;

– сумісність: OracleDB є відкритою системою, що підтримує стандарти SQL та JDBC, що робить її легко інтегрованою з іншими системами та застосунками. Вона також підтримує реплікацію та зовнішні ключі для забезпечення цілісності даних і взаємодії з іншими базами даних. Підтримка багатьох платформ у OracleDB дозволяє запускати її на різних операційних системах, забезпечуючи гнучкість інфраструктури;

– підтримка та спільнота: Oracle є відомим постачальником СКБД з великою активною спільнотою користувачів та розробників. Це означає, що існує велика кількість документації, ресурсів та форумів, де можна знайти відповіді на питання та отримати підтримку в разі потреби.

Порівняння OracleDB з іншими СКБД допоможе з’ясувати переваги і недоліки кожної системи та підкріпити вибір OracleDB для нашого проєкту. Ось порівняльний аналіз декількох популярних СКБД:

– MySQL – є однією з найпопулярніших відкритих СКБД і використовується для різноманітних проєктів. В порівнянні з OracleDB, MySQL має меншу функціональність і обмежену підтримку для оптимізації запитів та управління даними. Він частіше використовується для менших проєктів з обмеженими вимогами до масштабованості та навантаження;

– PostgreSQL – PostgreSQL є потужною відкритою СКБД, яка надає багатий набір функцій та можливостей. У порівнянні з OracleDB, PostgreSQL має схожу функціональність, але може бути менш масштабованим та менш надійним. Враховуючи вимоги проєкту для керування тарифними планами, OracleDB може забезпечити кращу підтримку для розподілених операцій та високих навантажень;

– Microsoft SQL Server – Microsoft SQL Server є комерційною СКБД, яка підтримує великі обсяги даних і надає різноманітні функції для розробки та управління базами даних. У порівнянні з OracleDB, Microsoft SQL Server може бути менш масштабованим і обмежувати кількість одночасних з’єднань та користувачів. Крім того, OracleDB надає ширшу підтримку для розподіленого середовища та гнучкіше конфігурування;

– MongoDB – MongoDB є документ-орієнтованою СКБД, яка підходить для ситуацій, коли потрібно зберігати та обробляти неструктуровані дані, такі як JSON. У порівнянні з OracleDB, MongoDB має іншу модель даних і меншу підтримку для складних транзакцій та оптимізації запитів. Вибір між OracleDB та MongoDB залежить від потреб проєкту: якщо основна потреба – робота з реляційними даними та потужними функціями, то OracleDB є кращим варіантом;

– SQLite – SQLite є легковаговою, вбудованою СКБД, яка зазвичай використовується для мобільних застосунків або простих проєктів. В порівнянні з OracleDB, SQLite має обмежену функціональність та масштабованість, але він простий у використанні і не вимагає окремого серверу для роботи.

Проаналізувавши аналоги, що в даний момент існують на ринку, найкращим варіантом, що забезпечує необхідний функціонал та стабільність, було обрано СКБД OracleDB.

3.2.2 Розробка табличного простору та ініціалізація таблиць

Усі таблиці та їх наповнення реалізовані можливості мови SQL. Не зважаючи на наявність в SQLDeveloper інструменту для створення таблиць в автоматичному режимі без використання коду – створення таким способом дає розробнику можливість налаштовувати таблиці та їх атрибути таким чином, як цього потребує проєкт.

У лістингу 3.3 наведено приклад коду створення та заповнення таблиці «clients».

Лістинг 3.3 Приклад створення та заповнення таблиць:

```
CREATE TABLE clients (
    id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    login VARCHAR2(255) NOT NULL,
    user_password VARCHAR2(255) NOT NULL,
    phone_number VARCHAR2(255) UNIQUE,
    tariff_id NUMBER,
    balance DECIMAL(10,2),
    tariff_start_date DATE,
    tariff_paid_until DATE,
    gigabytes_left VARCHAR2(255),
    minutes_out_left VARCHAR2(255),
    minutes_abroad_left VARCHAR2(255),
    FOREIGN KEY (tariff_id) REFERENCES tariffs(id) ON DELETE
CASCADE);
```

INSERT INTO clients (login, user_password, tariff_id, balance, tariff_start_date, tariff_paid_until, phone_number, gigabytes_left, minutes_out_left, minutes_abroad_left)

VALUES ('user1', '7c6a180b36896a0a8c02787eeafb0e4c', 1, 100.00, TO_DATE('2023-02-15', 'yyyy/mm/dd'), TO_DATE('2023-05-01', 'yyyy/mm/dd'), '+380474082457','3.56',75,0);

Як видно із лістингу, в таблицю занесено одного користувача із логіном «user1» та хешем, що згенеровано хеш-функцією MD5. Окрім інформації для входу, кожен запис у таблиці «clients» має значення поточного балансу, ідентифікатор поточного тарифу, дату початку підключення поточного тарифу (для історії тарифів), дату, до якої сплачено тариф, номер телефону, залишок гігабайт, залишок хвилин та залишок хвилин за кордон. Ідентифікатор тарифу – це зовнішній ключ, який поєднує таблицю із таблицею «tariffs» (під час відображення залишків у застосунку, максимальним значенням ініціалізується значення, що вказано для відповідного тарифу в таблиці «tariffs», а поточним значенням – значення із таблиці «clients»).

Із таблицею «clients», окрім таблиці «tariffs» пов’язано таблиці «Top_Ups», «Client_Option» та «Usage_history».

«Top_Ups» – це таблиця, в якій відображено історії поповнень. Оскільки кожне поповнення відбувається на конкретний акаунт – один із атрибутів цієї таблиці це ідентифікатор клієнта, акаунт якого було поповнено.

«Client_Option» – таблиця, шо містить інформації стосовно додаткових опцій підключених клієнту. Оскільки опція підключена на окремий

акаунт – один із атрибутів цієї таблиці це ідентифікатор клієнта.

«Usage_history» – таблиця, що містить у собі історію тарифних планів клієнтів.

Таблиці «Usage_history» та «Top_Ups», у випадку справжнього комерційного застосунку будуть мати безліч кортежів записів, отже необхідно не забувати про своєчасне видалення даних, як, наприклад, вчинив оператор Vodafone. Від нещодавнього часу в історії поповнень та списань коштів відображається інформація про три останні місяці. Ймовірно, старіші дані видаляються або зберігаються на сервері в окремі бази даних з метою швидкодії.

3.3 Розробка програмного продукту

Першим етапом розробки є під’єднання бази даних до застосунку. Для реалізації цієї задачі найпростішим варіантом є використання JDBC.

JDBC – це загальновживана коротка форма для підключення до бази даних Java. Використовуючи JDBC, з’являється можливість взаємодіяти з різними типами реляційних баз даних, зокрема, із OracleDB [35].

В основі JDBC лежить концепція так званих драйверів, що дозволяють отримувати з’єднання з базою даних по спеціально описаному URL. Драйвери можуть завантажуватись динамічно (під час роботи програми). Завантаживши, драйвер сам реєструє себе й викликається автоматично, коли програма вимагає URL, що містить протокол, за який драйвер «відповідає» [34–36].

На рисунку 3.1 зображено, як бібліотека JDBC під’єднується до застосунку на Java. Оскільки, у випадку TariffManage, використовується OracleDB – використовується Oracle JDBC 11, який доступний на офіційному сайті Oracle.

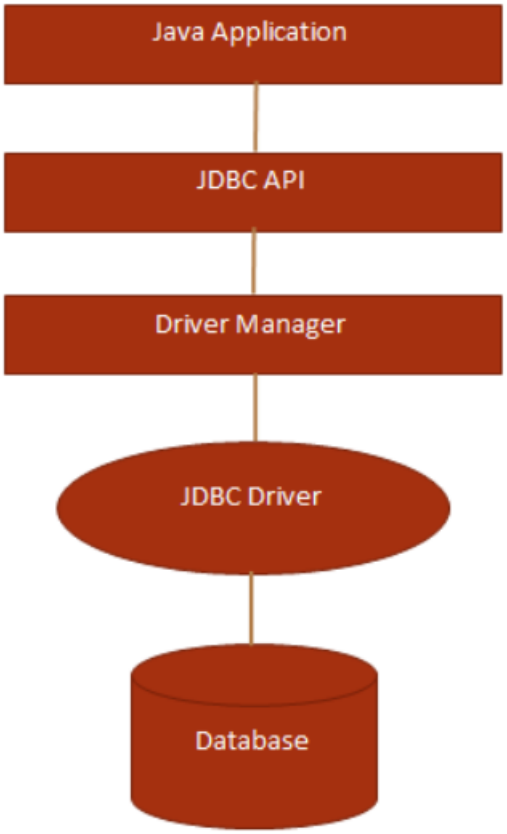

Рисунок 3.1 – Ієрархія взаємодії компонентів JDBC із Java

За роботу із JDBC відповідає клас DBConnection. Код класу представлений у лістингу 3.4.

Лістинг 3.4 Код класу DBConnection:

```
public final class DBConnection {
    private static Connection connection;
    private static boolean isDriverLoaded = false;
    static {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        isDriverLoaded = true;
    }
    private final static String DBName = "ORCL";
    private final static String url="jdbc:oracle:thin:@localhost:1521:"+
DBname
```

```
    private final static String user="manager";
    private final static String password="manager";

    public static Connection getConnection() throws SQLException{
      if(connection != null){
        return connection;
      }
      if(isDriverLoaded){
        connection = DriverManager.getConnection(url,user,password);
        System.out.println("Connection established");
        return connection;
      }
    }
}
```

З цього класу було видалено, обробку виключень, другорядні функції, тощо для скорочення коду, що представлений у лістингу 3.4.

Описання підключення:

Змінна connection – зміна, що ініціалізується під час першого запиту до БД та повертається класом кожного разу, коли застосунок проводить дії із базою даних.

Змінна *isDriverLoaded* – інформує, чи було завантажено драйвер JDBC.

Статичний блок static {…} – реалізує завантаження JDBC до пам’яті та змінює значення isDriverLoaded на true.

Змінна *url* – зміна, що зберігає у собі посилання на підключення до бази даних. Оскільки база даних розміщена на комп’ютері, на якому запускається застосунок – адресом є localhost та порт за замовчуванням: 1521.

Змінні *username* та *password* – логін та пароль доступу до бази даних.

Метод getConnection() – перевіряє, чи на поточний момент завантажено під'єднання до БД. Якщо завантажено – повертає об'єкт під'єднання. Якщо не завантажене – перевіряє, чи завантажений драйвер JDBC и відкриває під'єднання до БД [37].

Наступний етап розробки – це графічний інтерфейс. GUI реалізовано засобами JavaFX та «кастомної» бібліотеки TilesFX.

TilesFX – це бібліотека, яка дозволяє, використовуючи невелику кількість коду, відображати на екрані, так звані, плитки. GUI головного екрану застосунку розроблений виключно із використанням плиток. На рисунку 3.2 зображено набір плиток, які можна відображати, використовуючи TilesFX.

Рисунок 3.2 – Можливості бібліотеки TilesFX

Стилізацію застосунку розроблено за допомогою CSS. CSS (абревіатура від Cascading Style Sheets, що в перекладі означає каскадні таблиці стилів) – це спеціальна мова (мова стилів), за допомогою якої описують вигляду документів (як і де відображати елементи форми/сторінки), написаних мовами розмітки даних. Найчастіше CSS

використовується для документів, котрі розмічені мовою XML (HTML, FXML, тощо) [38]. Для стилізації використовують класи стилів або назви елементів. У випадку використання назви елементу, усі елементи, які керуються конкретним файлом .css будуть стилізовані тим стилем, що вказано у блоці.

Окрім простої стилізації CSS реалізують можливість динамічно керувати стилем об’єкта, використовуючи псевдокласи (focused, hover, тощо). Для реалізації анімації натискання кнопки та зміни її кольору використовуються класи hover і pressed відповідно. При попаданні курсору на кнопку – колір фону стає темнішим, забезпечуючи анімацію виділення, а при натисканні кнопка зменшується на 5%, що забезпечує анімацію натискання. Код для анімації кнопки наведено у лістингу 3.5.

Лістинг 3.5 CSS код для анімації дій із кнопкою:

```
.button:pressed {
   -fx-scale-y: 0.95;
   -fx-scale-x: 0.95;
}
.button:hover {
   -fx-background-color: #1E7CB6!important;
}
.button.exit:hover {
   -fx-text-fill:  #b4faff!important;
}
```

Правило !important використовується для перевизначення значення стилю, яке було вказано раніше. Як видно із третього блоку коду, після .button вказано клас .exit. Це означає, що цей стиль відноситься лише до тієї кнопка, в якій класом вказано «exit». Важливо розуміти, що для кнопки, для

якої вказано клас «exit» пріоритетним є блок *.button.exit:hover*, тож блок коду без вказання класу для цієї кнопки буде проігноровано.

Перед ініціалізацією основного застосунку користувачу необхідно пройти процедуру аутентифікації. Для цього він вводить логін та пароль у відповідні поля. У випадку, якщо користувач бажає не виходити із системи після закриття застосунку – він відмічає поле «Remember Me». У цьому випадку дані авторизації будуть збережені у бінарному вигляді в файл до директорії застосунку, внаслідок чого, під час наступного запуску застосунку, вони будуть зчитані і авторизація відбудеться автоматично (у випадку якщо користувач при виході із застосунку не натисне кнопку «Log out»).

У випадку, якщо користувач вводить невірні дані – на екрані відобразиться помилка входу, а якщо одне з полів буде пустим – користувач побачить анімацію смикання поля, яке він не заповнив. Такий функціонал реалізовано за допомогою бібліотеки AnimateFX, яка містить у собі низку методів для анімаційних дій різного роду із вікнами або елементами.

Після натискання кнопки «Login» ініціалізується з’єднання застосунку із базою даних та зчитування інформації про користувача з введеними паролем та логіном (пароль після введення користувачем перетворюється на числову послідовність – хеш). У випадку, якщо співпадіння знайдено – система ініціалізує головний екран і записує ідентифікатор користувача, у зміну, яку буде використовувати для витягання з бази даних саме тих записів, які відносяться для поточного користувача.

Процес поповнення балансу запускається при натискання користувачем на плитку балансу. Вартість тарифу автоматично буде списано з балансу користувача у наступний день, після дати, яка вказана в полі tariff_paid_until таблиці «clients». Також, з балансу списується сума при перезамовленні тарифу користувачем, при переході на новий тариф або при підключенні додаткової опції. Перезамовлення тарифу – це опція, що

дозволяє користувачу у поточний момент сплатити вартість тарифу та оновити усі залишки (у цьому випадку дата наступної сплати виставляється на рівні: поточна дата + 28 днів). Оскільки застосунок є прототипом – баланс поповнюється автоматично на суму, яку вказав користувач, а на екрані комп'ютера імітовано відкриття сторінки браузера для оплати з автоматично – вказаною сумою, яку обрав користувач (рис. 3.3).

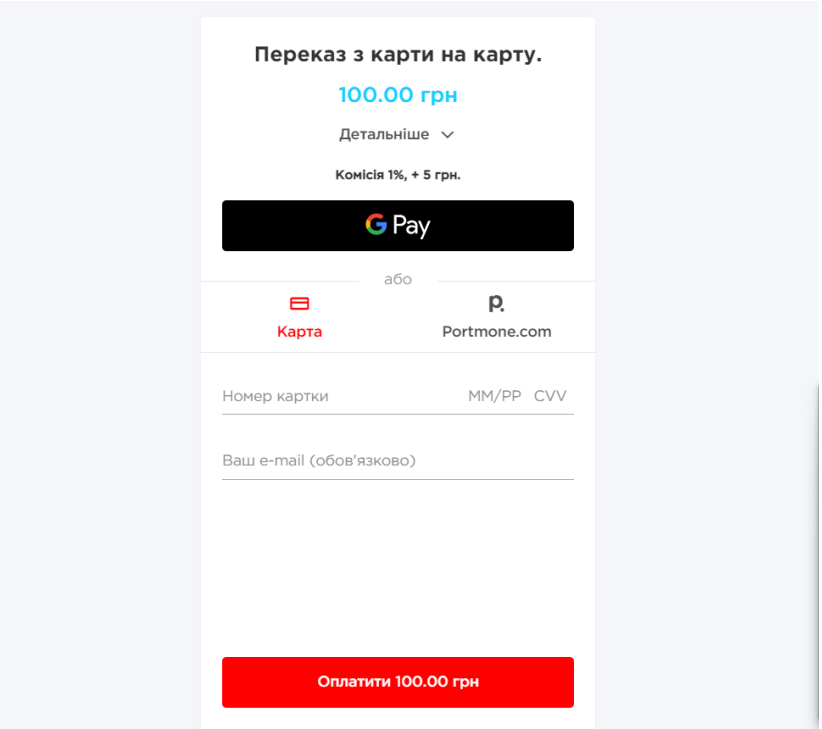

Рисунок 3.3 – Імітація процесу поповнення балансу

Такого результату було досягнено завдяки використанню сервісу Portmone. Після реєстрації у сервісі було створено 5 посилань для оплати (20 грн, 50 грн, 100 грн, 200 грн, 500 грн) за вказаними реквізитами. Відповідне посилання відкривається в залежності від того, яка сума з запропонованих буде обрана користувачем. Процес оплати було створено за прикладом застосунку німецького оператору Telekom.

У вкладці зміни тарифу користувач може обрати новий тариф із запропонованих. Кожен тариф відображається на окремій плитці. Генерація плиток відбувається автоматично, тож у випадку видалення тарифу із бази

даних – він просте не буде відображатись на екрані. Такої поведінки було досягнуто завдяки методу, який запускається для кожного тарифу із бази даних та ініціалізує нову ідентичну іншим плитку (із своїм наповненням тарифу, назвою та ціною). Після ініціалізації плитка додається з правої частини екрану. Отже, якщо з якоїсь причини необхідно буде змінити розташування тарифів на екрані – все що необхідно, це змінити їх порядок в базі даних [39].

Кожна плитка тарифу є анімованою, вона збільшується при наведенні курсору та зменшується, коли курсор з неї виходить. За таке поводження відповідає клас ScaleTransition пакету javafx.animation. Замість прописування однакового коду для кожної плитки тарифу було розроблено два класи, що імплементують інтерфейс EventHandler<MouseEvent, тобто класи є обробниками подій. Перший клас MouseEnteredEventHandler відповідає за виставлення збільшення плитки у 1,05 рази по осі *X* та *Y* за 0,3 секунди.

Лістинг 3.6 Головний метод класу MouseEnteredEventHandler:

```
@Override
public void handle(MouseEvent event) {
   ScaleTransition scaleTransition=new
ScaleTransition(Duration.seconds( 0.3, vBox);
   scaleTransition.setToX(1.05);
   scaleTransition.setToY(1.05);
   scaleTransition.play();
}
```

Клас MouseExitedEventHandler виконує зворотній процес. Він повертає значення плитки до 1*початкове значення *X* та 1*початкове значення *Y* за 0,3 секунди.

Лістинг 3.7 Головний метод класу MouseExitedEventHandler:

```
    public void handle(MouseEvent event) {
        ScaleTransition scaleTransition = new
ScaleTransition(Duration.seconds( 0.3), vBox);
        scaleTransition.setToX(1);
        scaleTransition.setToY(1);
        scaleTransition.play();
    }
```

Окрім цих слухачів подій, кожній кнопці зміни тарифу додано слухач подій натискання, який перезаписує тариф користувача, баланс, дати та його залишки у базі даних. Після чого оновлює дані на головному екрані, щоб зобразити нові баланси та залишки (залишки максимальні після оновлення тарифу).

У вкладці статистики користувач може подивитись на історію своїх поповнень. У версії застосунку, що розробляється не реалізовано процес видалення даних із таблиці історії.

Для історії тарифів було реалізовано складну ієрархію умов (if), щоб після вибору двох дат (початок і кінець) користувач бачив усі тарифи, які використовував за цей період. Ієрархія умов дозволяє побачити у цьому списку усі випадки перетину дат використання тарифів із обраним діапазоном, а саме:

– ситуація, коли дата початку обслуговування до мінімальної дати, а дата закінчення після максимальної (або тариф ще не закінчився);

– ситуація, коли мінімальна і максимальні дати входять в діапазон;

– ситуація, коли мінімальна дата входить в діапазон, а максимальна не входить;

– ситуація, коли мінімальна дата не входить в діапазон, а максимальна – входить.

Окрім історії тарифів, користувач має можливість ознайомитись із історією своїх поповнень. Із таблиці історії поповнень дані, аналогічно таблиці історії тарифів, не видаляються.

Перед відображенням на екрані історії поповнень або тарифів, дані необхідно відсортувати, а, оскільки предметом сортування є дата – JVM автоматично не може відсортувати цю інформацію, тому необхідно створити клас – Comparator (клас для порівняння).

Створення об’єктів із конструкторами та «гетерами» тягне за собою від 30 строчок коду, до яких доведеться додати ще й метод порівняння об’єктів цього класу. Саме для таких ситуацій у Java 14 було введено записи (records). Запис – звичайний клас із незмінними (final) полями, який уміщує в одній строчці конструктор, «гетери» для усіх полів, перевизначені методи equals, hash, toString.

Використовуючи записи, вдалося зекономити час на написанні зайвого коду та покращити його зовнішній вид.

Лістинг 3.8 Приклад використання запису замість класу:

```
public record TariffUsageHistory(String tariffName, Date beginingDate, Date endDate) implements Comparable<TariffUsageHistory> {
    @Override
    public int compareTo(TariffUsageHistory o) {
        if ((beginingDate.equals(o.beginingDate))) return 0;
        else if (beginingDate.after(o.beginingDate)) return -1;
        else return 1;
    }
}
```

Як видно з лістингу 3.8, запис імплементує інтерфейс Comparable. При спробі відсортувати список записів JVM буде користуватись перевизначеним методом цього класу compareTo для порівняння дат,

завдяки чому останні тарифні плани будуть відображатись нагорі списку. Аналогічний принцип реалізовано для запису історії поповнень.

3.4 Тестування застосунку

Під час першого запуску застосунку відкривається вікно авторизації користувача (рис. 3.4).

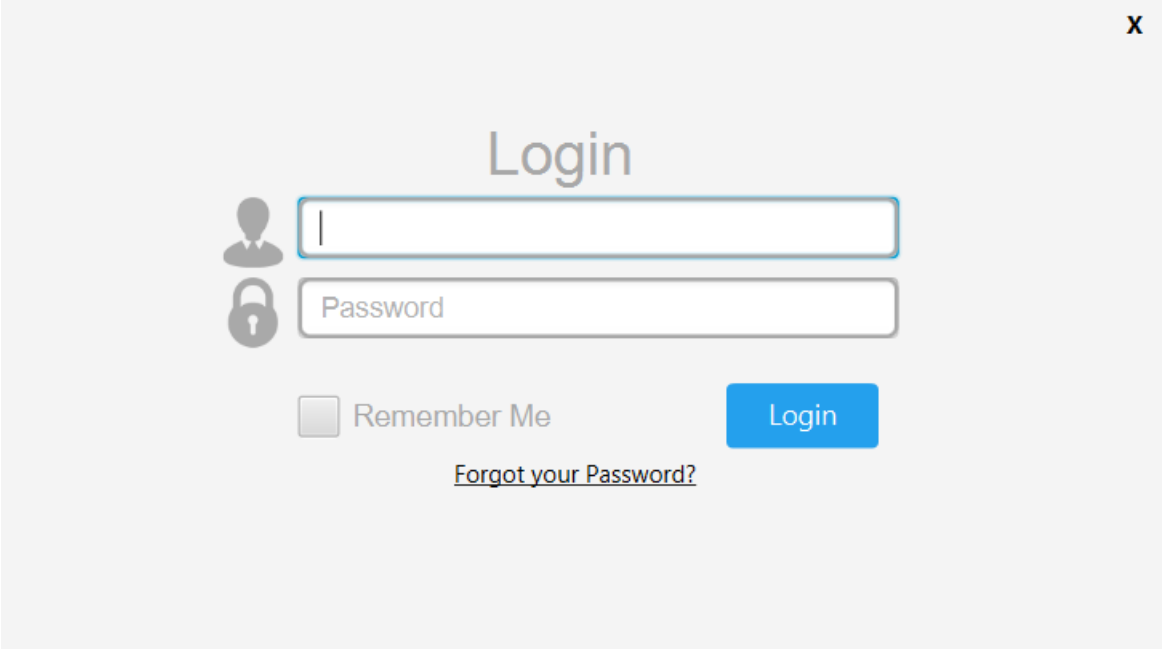

Рисунок 3.4 – Вікно авторизації

Для тестування було створено 10 тестових користувачів: [{user1, password1}, {user2, password2},…,{user10, password10}].

Якщо користувач не хоче кожного разу виконувати авторизації – він помічає поле «Remember Me». У цьому випадку, під час наступного запуску застосунку, авторизація відбудеться автоматично.

Якщо користувач вводить неправильні дані, застосунок видає помилку авторизації (рис. А.1), а якщо нащо тискає на кнопку «Forgot your

Password?» – отримає повідомлення, що тимчасовий пароль було відправлено на його номер телефону (рис. А.2).

Після успішної авторизації перед користувачем відкривається головний екран із поточною інформацією, щодо його тарифу (рис. 3.5).

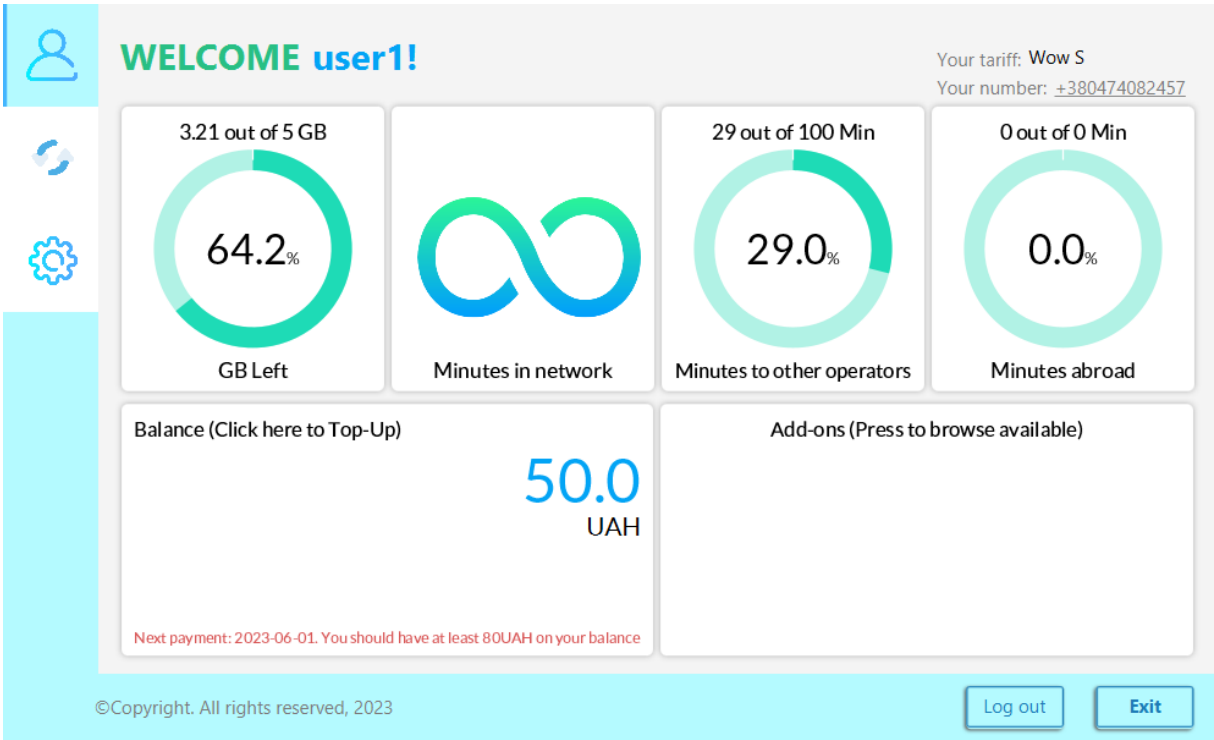

Рисунок 3.5 – Головний екран застосунку

На верхній половині екрану відображаються залишки тарифу. У нижній частині екрану плитки балансу та додаткових опцій. Інтерактивний напис «Welcome <username>» змінюється автоматично в залежності від того, який користувач входить у систему. Напис «Your tariff: <tariff_name>» змінюється в залежності від поточного тарифу користувача. На рисунку А.3 зображено альтернативний вигляд головного екрану для користувача «user4».

Окрім інформації залишків, на головному екрані користувач має можливість поповнити баланс. Натиснувши на плитку балансу, перед ним відкривається вікно із запропонованими сумами (рис. 3.6), а при виборі суми відкривається вікно поповнення із відповідною сумою (рис. 3.3).

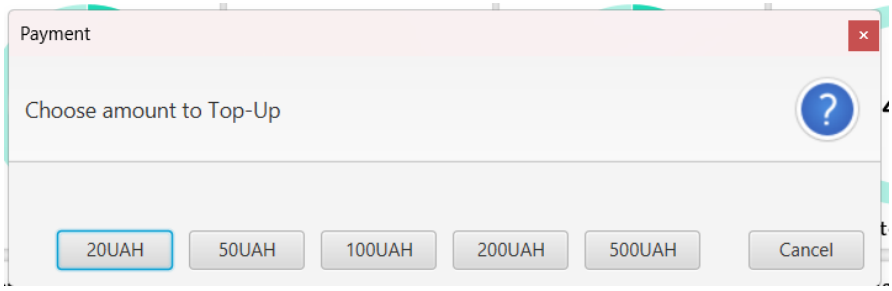

Рисунок 3.6 – Вікно вибору суми

На плитці балансу зображено дату наступної сплати та, у випадку, якщо баланс менше вартості тарифу, повідомлення, що необхідно мати на рахунку суму, не менше ніж *n*, де *n* – вартість тарифу. Колір напису змінюється в залежності від того, чи вистачає коштів на наступну сплату. Ознайомитись із ситуацією, коли коштів на наступну сплату вистачає можна на рисунку А.3.

Також, на головному екрані користувач має можливість перезамовити пакет послуг або підключити додаткову опцію. Процес перезамовлення відбувається наступним чином: Користувач натискає на назву свого тарифу у верхній правій частині екрану та підтверджує свій намір у вікні, що з’являється на екрані (рис. 3.7). Якщо коштів недостатньо – він бачить відповідне повідомлення (рис. А.4).

Рисунок 3.7 – Вікно підтвердження перезамовлення тарифу

Якщо коштів достатньо – відбувається списання з балансу вартості тарифу та оновлення залишків і дати наступної сплати.

Для тестування було поповнено баланс користувача «user1» на 50 гривень та перезамовлення тариф. Із результатами перезамовлення можна ознайомитись на рисунку А.5.

Процес підключення додаткової опції виглядає наступним чином: Користувач натискає на плитку додаткових опцій та обирає необхідну із списку опцій (рис. 3.8).

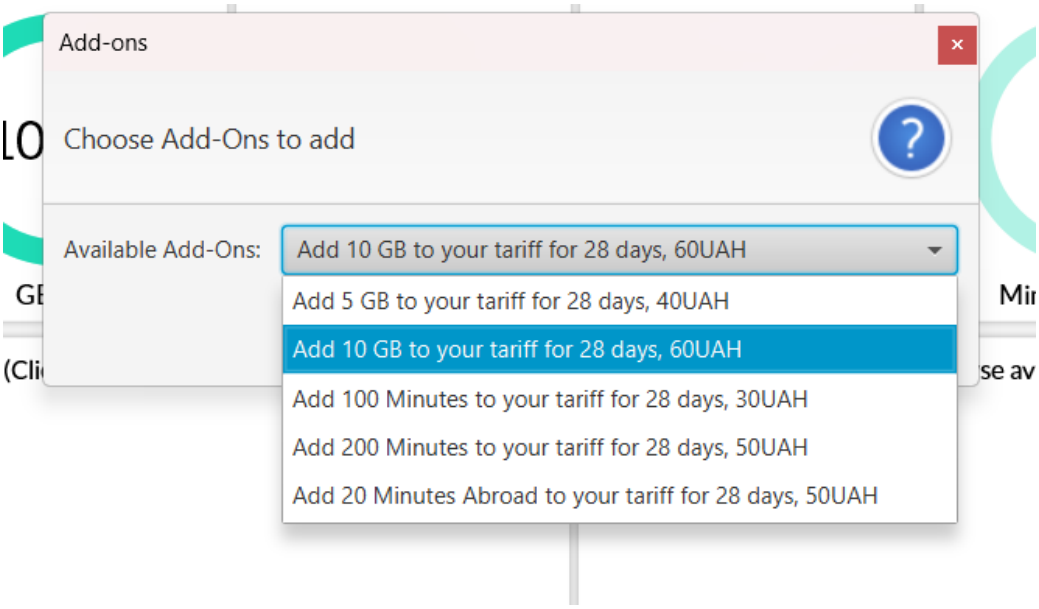

Рисунок 3.8 – Обирання додаткової опції

Припустимо, що «user1» бажає підключити 10GB на 28 днів за 60UAH. Після підтвердження кнопкою «OK» з’являється повідомлення, що на балансі недостатньо коштів (рис. А.6). Тепер необхідно поповнити баланс на 50 гривень і спробувати ще раз. Тепер на плитці Add-ons відображається підключена послуга та дата, до якої вона діє (рис. 3.9).

На рисунку А.7 можна ознайомитись із зовнішнім виглядом плитки «add-ons», якщо частину підключеної послуги вже використано. Навіть якщо усю послугу використав користувач, ця послуга буде знаходитись активною на плитці додаткових послуг до вказаної на ній дати закінчення. Якщо послуга вже активна, підключити її ще раз неможливо, а із списку додаткових послуг вона зникає (рис. 3.10).

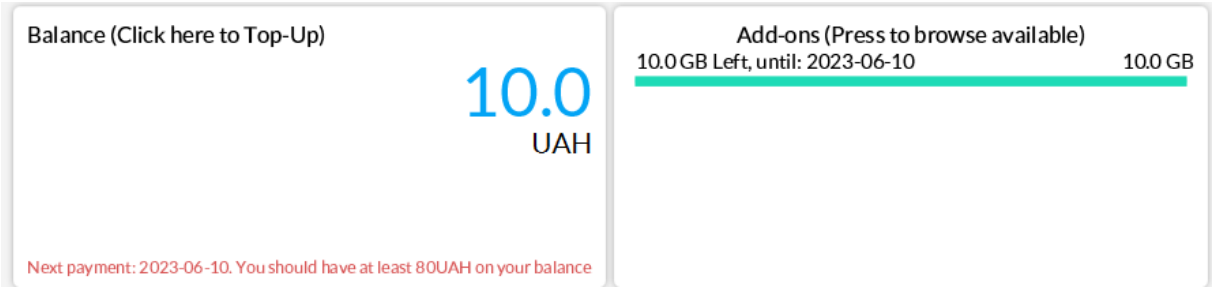

Рисунок 3.9 – Результат активації додаткової послуги

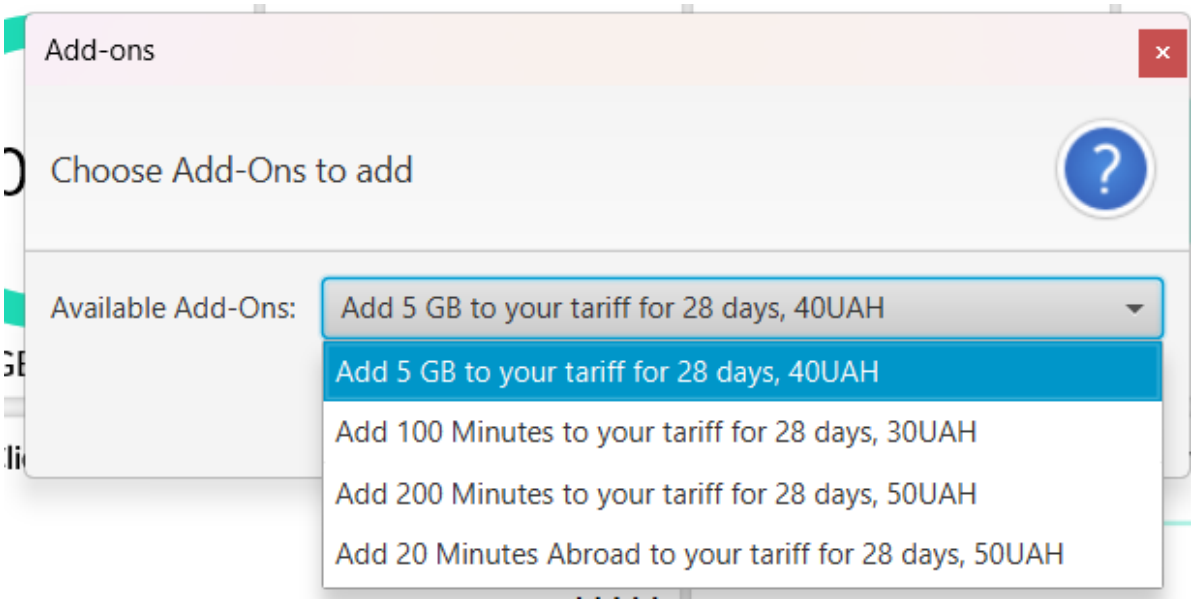

Рисунок 3.10 – Вікно додаткових при підключеній послузі

На наступній вкладці користувач має можливість змінити свій тариф на інший. Кнопка «Change» на поточному тарифі змінена на «Your tariff» та недоступна до натискання (рис. 3.11).

Кожна плитка тарифу плавно збільшується при наведенні мишкою, та зменшується при видаленні з нею курсору. Це наглядно зображено на прикладі тарифу «UNLIMITED» на рисунку А.8. Кожна плитка генерується автоматично, тож видалення запису про тариф UNLIMITED не призведе до помилки, але призведе до видалення користувачів за властивістю SQL – правила «ON DELETE CADCADE», що забезпечує цілісність даних в БД. Для реалізації видалення тарифу було б краще реалізувати додатковий атрибут «isAvailable» таблиці «tariffs» або примусово перевести клієнтів з

деякої дати на інший тариф, проте на поточний момент цей функціонал не реалізовано, тому при видалені тарифу, усі користувачі, що його використовують, а саме: user5 та user9, будуть видалені із системи. Із зовнішнім виглядом вікна тарифів після видалення тарифу «UNLIMITED» можна ознайомитись на рисунку А.9.

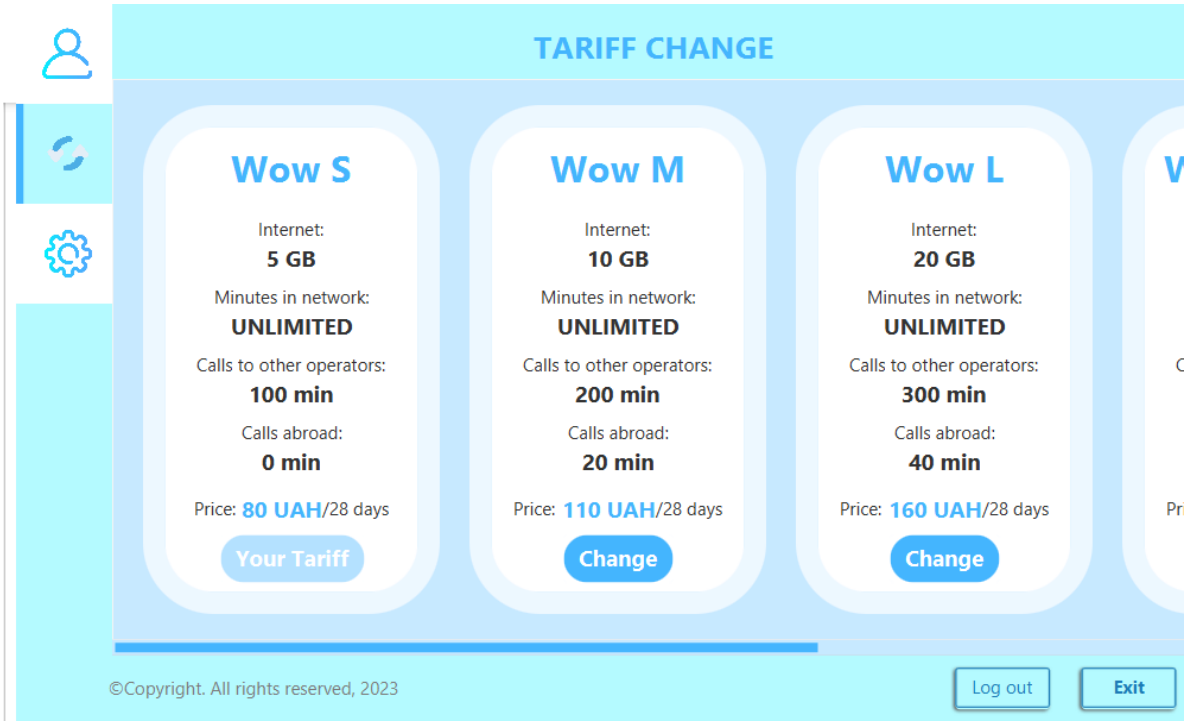

Рисунок 3.11 – Вікно зміни тарифного плану

Для підключення тарифу користувач може натиснути кнопку відповідного тарифу. У випадку, якщо коштів недостатньо він побачить повідомлення аналогічне рисункам А.4 та А.6. Якщо коштів достатньо з’явиться вікно підтвердження намірів користувача (рис. 3.12). Для проведення цієї операції рахунок користувача «user1» було поповнено ще на 200 гривень.

З метою запобігання багаторазової зміни тарифу або його перезамовлення, в програму вбудовано захист, який полягає в обмежені змінення тарифу або його перезамовлення більше, ніж 1 раз на 24 години.

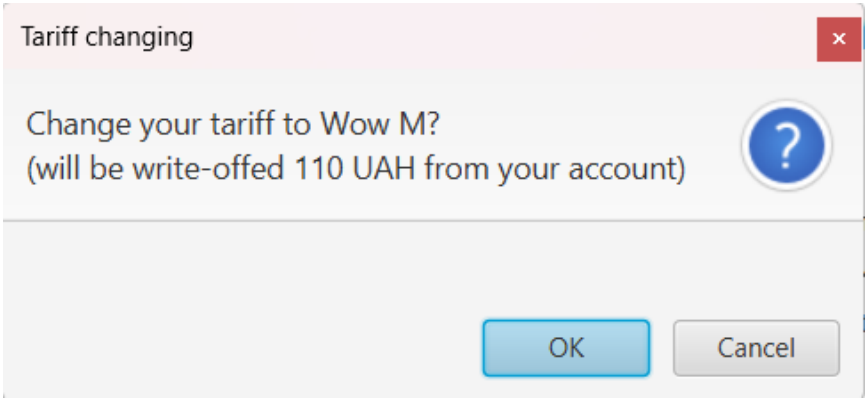

Рисунок 3.12 – Вікно підтвердження зміни тарифу

Після успішної зміни тарифу користувач побачить підтвердження, а кнопка тарифу, на який він перейшов автоматично зміниться на «Your Tariff» (рис. 3.13).

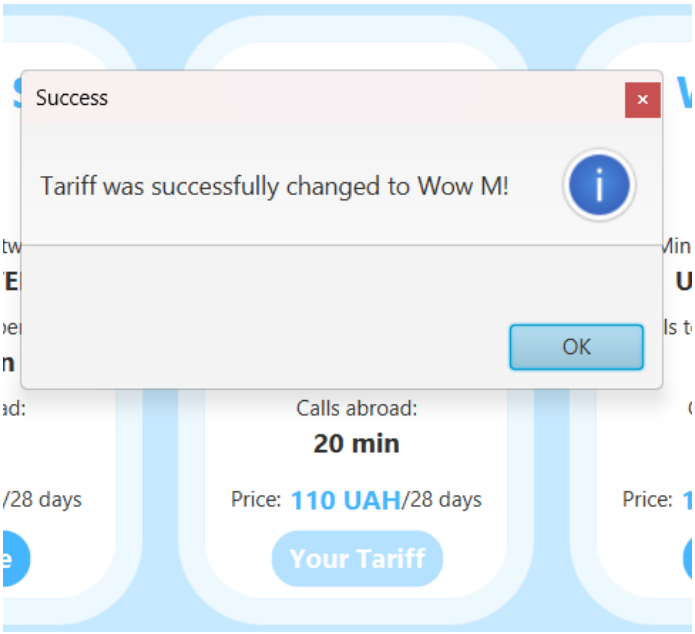

Рисунок 3.13 – Повідомлення про успішну зміну тарифу

Якщо тариф було змінено – головний екран оновиться із новими даними (рис. А.10).

На вкладці статистики користувач може побачити інформації про історію поповнень та тарифів. Необхідно ввести початкову та кінцеві дати,

після чого натиснути відповідну кнопку. На рисунку 3.14 зображено історію поповнень тарифу, а на рисунку 3.15 історію змін тарифів, які відсортовано від більшої дати до меншої. Історія включає у себе усі операції, які було зроблено під час розробки, тестування, а також під час створення звіту.

Рисунок 3.14 – Історія поповнень

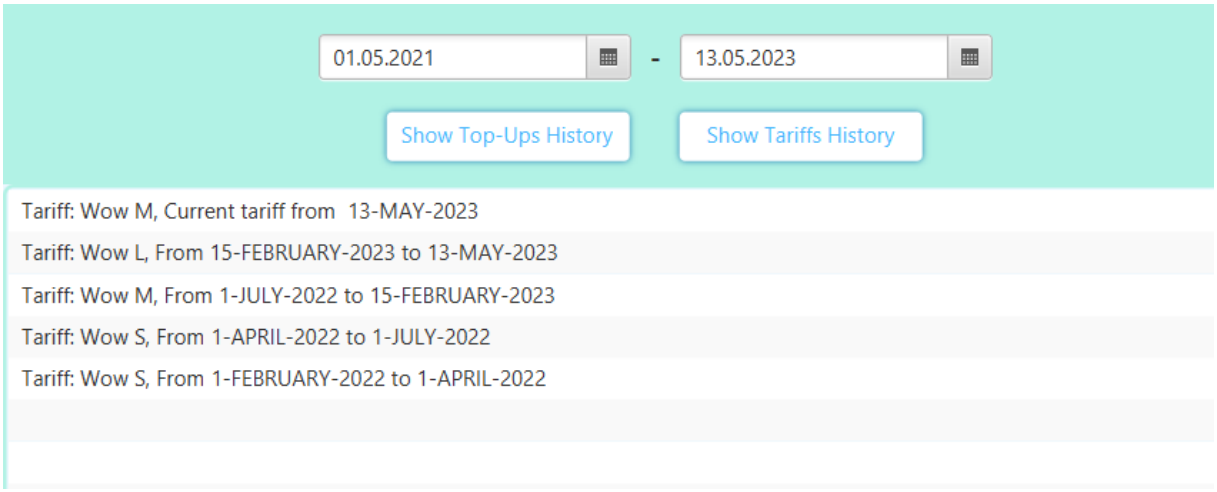

Рисунок 3.15 – Історія тарифів

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблена і реалізована модель управляння тарифами мобільного зв’язку, яка дозволяє користувачам відстежувати свій тариф та змінювати його. Розроблений застосунок призначений для контролю тарифного плану, гігабайтів інтернету, коштів на власному рахунку, та інше. Він може бути корисним для кожного, хто хоче у швидкому доступі бачити свій тариф, та мати можливість автономно змінити його.

Був проведений аналіз подібних сервісів для управління тарифними планами, які використовуються в Україні. Все існуючі платформи мають ряд проблем, серед яких є проблеми з розумінням інтерфейсу користувача, нестабільною роботою застосунку, обмеженими можливостями користування, та інше. Розроблена система спрямована на вирішення більшості з цих проблем, адже вона дозволяє інтуїтивно розуміти інтерфейс користувача, надає детальну статистику щодо тарифного плану, та дозволяє змінювати тариф за власним бажанням та потребами у даний момент часу. Правильність роботи застосунку було перевірено на тестових даних.

Результати роботи апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vodafone. URL: https://www.vodafone.ua/news/vodafone-ukrana-v-3-kvartali-vidnovlennya-merezhi-ta-dopomoga-krani (дата звернення 10.04.2023).

2. Vodafone Україна. URL: https://uk.m.wikipedia.org/wiki/ Vodafone _Україна (дата звернення: 10.04.2023).

3. Sitnikov, D., Titova, O., Minukhin, S., Kovalenko, A., & Titov, S. (2018, October). Informativity of association rules from the viewpoint of information theory. In *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)* (pp. 595-598). IEEE.

4. Sitnikov, D., Ryabov, O., Titova, O., & Kovalenko, A. (2018, May). Assessment of extended aggregated association rules. In *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)* (pp. 93-97). IEEE.

5. Kyivstar. URL: https://uk.m.wikipedia.org/wiki/Київстар (дата звернення: 12.04.2023).

6. MyKyivstar. URL: https://kyivstar.ua/miykyivstar (дата звернення: 13.04.2023).

7. Тітова, О. В. (2018). Основи інформаційного забезпечення управління.

8. Тітов, С. В., & Тітова, О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології. *Вісник Харківської державної академії культури. Серія: Соціальні комунікації*, (47), 127-134.

9. Lifecell. URL: https://uk.wikipedia.org/wiki/Lifecell (дата звернення: 16.04.2023).

10. MyLifecell. URL: https://www.lifecell.ua/uk/mij-lifecell/ (дата звернення: 16.04.2023).

11. Grishin, I. Y., Timirgaleeva, R. R., Titov, S. V., & Titova, Y. V. (2017, May). Technology of wireless transmission of energy to remote objects based on multi-frequency system of transmitters. In *2017 XI International Conference on Antenna Theory and Techniques (ICATT)* (pp. 443-447). IEEE.

12. Titova, O., & Vysotskyi, D. (2021). The development of a subsystem for palliative patients information support.

13. PEOPLEnet. URL: https://uk.wikipedia.org/wiki/PEOPLEnet (дата звернення: 17.04.2023).

14. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

15. JavaFX documentation. URL: https://dut.edu.ua/ua/news-1-626-6031-mova-programuvannya-java-ta-platforma-javafx-prikladi-zastosuvannya_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення: 20.04.2023).

16. Sharan, K., & Späth, P. (2022). *Learn JavaFX*.

17. Alkhars, A., & Mahmoud, W. (2019). Cross-Platform Desktop Development (JavaFX vs. Electron) pp. 7-15.

18. *17: Building User Experience and Interfaces with Java*. Apress.

19. QtJambi. URL: https://wiki.qt.io/Qt_Jambi (дата звернення: 21.04.2023).

20. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, Сучасні інформаційні системи, 7(1), С. 5-13.

21. AWT documentation. URL: https://uk.myservername.com/what-is-java-awt (дата звернення: 22.04.2023).

22. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору

ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 5-12.

23. Swing documentation. URL: https://uk.myservername.com/java-swing-tutorial-container (дата звернення: 23.04.2023).

24. Carpenter, J., & Hewitt, E. (2022). *Cassandra: The Definitive Guide,(Revised)*. " O'Reilly Media, Inc.".

25. Доценко, С. І. (2023). Організація та системи керування базами даних.

26. Типи баз даних. URL: https://dou.ua/lenta/articles/types-of-databases/ (дата звернення: 27.04.2023).

27. Yanholenko, O., Cherednichenko, O., Yakovleva, O., & Arkatov, D. (2020). A Model for Estimating the Security Level of Mobile Applications: a Fuzzy Logic Approach. In *IntelITSIS* (pp. 252-266).

28. Lucidchart. URL: https://www.lucidchart.com/ (дата звернення: 29.04.2023).

29. Що таке Хеш? URL: https://exbase.io/uk/wiki/shho-take-khesh (дата звернення: 29.04.2023).

30. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, IEEE Access, 10, pp. 124738-124746.

31. Turbosms. URL: https://turbosms.ua/ (дата звернення: 29.04.2023)

32. Тітов, С. В., & Тітова, О. В. (2013). Web-сайти органів місцевого самоврядування як складова впровадження е-урядування в Україні.

33. Jones, D. (2014). *Learn SQL Server Administration in a Month of Lunches*. Simon and Schuster.

34. Sitnikov, D., Titova, O., Romanenko, O., & Ryabov, O. (2009). A method for finding minimal sets of features adequately describing discrete information objects. *WIT Transactions on Information and Communication Technologies*, *42*, 143-153.

35. What is JDBC? URL: https://uk.myservername.com/java-jdbc-tutorial-what-is-jdbc (дата звернення: 01.05.2023).

36. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, International Journal of Engineering and Information Systems, 6(2), pp. 40-48.

37. Java_Database_Connectivity. URL: https://www.wik.uk-ua.nina.az/Java_Database_Connectivity.html (дата звернення: 02.05.2023).

38. CSS URL: https://css.in.ua/article/shcho-take-html_10 (дата звернення: 02.05.2023).

39. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

40. Личагіна С. М. (2023) Розробка застосунку для управління тарифними пакетами компанії мобільного зв’язку: Радіоелектроніка і молодь у XXI столітті: 27 міжнародний молодіжний форум. (Харків 10-12 травня 2023 р.) с. 90-91.