Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Капусти Максима Олеговича*
(ПІБ)

академічної групи *122-17-2*
(шифр)

спеціальності *122 Комп'ютерні науки та інформаційні технології*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки та інформаційні технології*
(назва освітньої програми)

на тему: *Розробка автоматизованої інформаційної системи для Інтернет банкінгу*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-17-2* *Капусти Максима Олеговича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка автоматизованої інформаційної системи для Інтернет банкінгу*

затверджена наказом ректора НТУ «ДП» від *07.06.2021 р.* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***14.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***28.05.2021 р.***

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Капуста М.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *10.02.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *14.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 60 с., 30 рис., 0 табл., 3 дод., 20 джерел.

Об'єкт розробки: клієнт-серверний додаток у вигляді веб-сайту.

Мета кваліфікаційної роботи: користування та управління рахунками за допомогою розробленого веб-додатку.

У вступі розглядається сучасний стан проблеми, конкретизується мета кваліфікаційної роботи, актуальність та галузь її застосування, уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

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 програми, що надає можливість керувати особовим рахунком.

Актуальність інформаційної системи визначається великим попитом на подібні розробки, що дають можливості для створювання зручної інформаційної системи для управління рахунком.

Список ключових слів: КОМП’ЮТЕР, ІНФОРМАЦІЙНА СИСТЕМА, МЕНЮ, ВЕБ-ДОДАТОК, ПРОЕКТУВАННЯ.

ABSTRACT

Explanatory note: 60 p., 30 figs., 0 tabl., 3 add., 20 sources.

Development object: client-server application in the form of a website.

The purpose of the diploma project: use and manage accounts using a developed web application.

The introduction examines the current state of the problem, specifies the purpose of the qualification work, relevance and the field of its application, specifies the task statement.

In the first section, the analysis of the subject area was carried out, the relevance of the task and the designation of the development was determined, the task statement was developed, the requirements for the software implementation, the technologies and software tools were developed.

The second section provides: the purpose of the program, the description of the applied mathematical methods, the description of the used technologies and programming languages, description of the structure of the program and algorithms of its operation, and detailed description of the work of the developed software product.

In the economic section the complexity of the developed information system is determined, the calculation of the cost of work on the creation of the program is calculated and the time for its creation is calculated.

The practical value is to create a program that allows you to manage a personal account.

The relevance of the information system is determined by the high demand for such developments, which provide opportunities for creating a convenient information system for account management.

Keywords: COMPUTER, INFORMATION SYSTEM, MENU, WEB APPLICATION, DESIGN.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система;

JSP –java server pages;

БД – база даних;

ІТ – інформаційні технології;

СУБД – система управління базою даних;

CSS – Cascading Style Sheets «каскадні таблиці стилів»;

ПК – персональний компю’тер;

HTML – HyperText Markup Language «мова розмітки гіпертексту»;

MVC – Model-View-Controller;

DAO – Data access object «об'єкт доступу до даних»;

ВСТУП

Важко уявити сучасний світ без використання інтернету та сучасних технологій. Тож у наш час майже не лишилось сфери діяльності, яка з того чи іншого боку не використовувала переваги сучасних технологій. Розвиток інформаційних технологій (IT) значно покращує та прискорює виконання та ефективність роботи з інформацією.

Швидкий розвиток технічного прогресу також впливає і на банківську сферу. Розвиток технологічного прогресу дозволяє не тільки підвищити ефективність виконання фінансових операцій, але й розширити клієнтську базу.

На сьогодення більшість банків по всьому світу використовують системи, для віддаленого доступу та передачі інформації. Такі системи називаються інтернет-банкінгом. За допомогою цих систем, клієнт банку може здійснити майже будь-які операції за своїм рахунком. Зазвичай, йому для цього потрібен лише доступ до інтернету.

Завдяки системам інтернет-банкінгу значно підвищується рівень обслуговування клієнтів, оскільки у більшості випадків не треба звертатися до відділення банку. Також, завдяки інтернет банкінгу банки також отримують вигоду: зменшуються витрати на персонал, на утримання філіалів, на оренду приміщень.

У нас час, навіть є банки без філій, ці банки пропонують клієнтам доступ до банківских операцій виключно через інтернет-банкінг. Це допомагає економити на філіалах, та впроваджувати різні фішки та акції, які приваблюють клієнтів, та покращують враження від користування послугами банку.

На мій погляд, найзручнішим видом взаємодії між банком і клієнтом є використання веб-застосунку. Тому, що для користування веб-застосунком клієнту не потрібно мати спеціального обладнання, потрібен лише пристрій з доступом до мережі інтернет. Також розробка такого додатку вигідніша для замовника у фінансовому плані, в порівнянні з іншими видами застосунків.

Метою дипломного проекту є створення веб-додатку, за допомогою якого клієнт зміг би керувати своїми рахунками, переглядати за ними інформацією та здійснювати платежі. Основною мовою розробки буде використано Java, оскільки ця мова розробки є зручною, та досить популярна на ринку технологій.

Згідно цій інформації була розроблена тема дипломної роботи: «Розробка автоматизованої інформаційної системи для інтернет банкінгу».

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості с предметної галузі

Інтернет-банкінг є одним із видів дистанційного банківського обслуговування. Послуги дистанційного банкінгу – це послуги віддаленного доступу до різних банківських операцій, які банк пропонує своїм клієнтам.

Першим видом дистанційного банкінгу був телефонний банкінг, який вперше почали використовувати у Америці наприкінці двадцятого століття. Усього можна виділити чотири основних види дистанційного банківського обслуговування: PC-Банкінг, SMS-банкінг, телефонний банкінг, та інтернет-банкінг.

PC-Банкінг – система яка була створена в основному для юридичних осіб, для її використання потрібне спеціальне обладнання та програмне забезпечення зі сторони банку, та зі сторони клієнту. Взаємодія з банком здійснювалася за допомогою спеціального модему який напряму з’єднувався з сервером банку. Наразі ця система майже не використовується, оскільки є не зручною, та потребує спеціального обладнання.

Телефонний банкінг – це інформаційна система, яка дозволяє керувати рахунком, та проводити фінансові операції за допомогою телефону. Ця система довгий час користується популярністю, оскільки для її використання потрібен лише телефон з можливістю сотового зв’язку. Взаємодія здійснюється або через зв’язок з оператором Call-центру, або через автоматизовану систему. Але її недоліком є те, що оператори можуть бути зайняті і очікування клієнта на запит може бути довгим. А проведення операцій через автоматизовану систему, може бути не зрозумілим деяким клієнтам через відсутність зручного графічного інтерфейсу.

SMS банкінг – комунікація між клієнтом і банком здійснюється за допомогою SMS-повідомлень. Користувача інформують про операції такі як: зняття та поповнення коштів, повідомлення за здійснюваними операціями. Також клієнт завдяки відправці повідомлень на номер банку може здійснювати перекази між своїми рахунками, оплачувати комунальні послуги, та інше. Але цей спосіб також не є досконалим через обмежений функціонал, та відсутність графічного інтерфейсу.

І нарешті, інтернет-банкінг. Інтернет-банкінг – дав змогу прямого доступу до банківського рахунку через Інтернет за допомогою комп'ютера або мобільного телефону на яких є змога встановити інтернет браузер, або спеціальний додаток від банку. Це дозволило клієнтам отримувати усю необхідну інформацію за рахунками, та здійснювати за ними необхідні операції. Після впровадження інтернет-банкінгу значно підвищилась швидкість та якість обслуговування, зменшились черги у відділеннях банків. Також оскільки система інтернет банкінгу повністю автоматизована, то клієнт може отримати потрібні йому послуги цілодобово, на відміну, від банківских відділень які працюють згідно графіку.

На мій погляд, система інтернет-банкінгу є найефективнішою та найзручнішою, оскільки в порівнянні з іншими системами. По-перше це простота використання – клієнту потрібні лише його данні для входу та доступ до інтернету, завдяки зрозумілому інтерфейсу майже будь-хто зможе скористатися усіма послугами. По-друге це зручність – для користування інтернет-банкінгом потрібно мати рахунок у банку, пристрій з якого можна з'єднатись з мережою інтернет. Також зберігання фінансових документів у зручному цифровому форматі, замість купи паперових є вагомим аргументом для використання інтернет-банкінгу. По-третє – це швидкість здійснення операцій, для їх здійснення не потрібно йти до відділення, а усі операції відбуваються швидко, і за запитом, у режимі реального часу можна побачити їх статус.

Наразі у Україні стрімко розвиваються послуги інтернет-банкінгу, оскільки ця послуга має великий попит багато банків впроваджують собі таку систему, для зручнішого обслуговування клієнтів.

Розглянемо деякі популярні системи інтернет-банкінгу, їх переваги та недоліки.

Raiffeisen Online – система для інтернет-банкінгу від банку українського банку Raiffeisen Bank Aval. На сайті, після входу до своєго профілю клієнт має змогу здійснювати операції за своїм рахунком. Користувачі можуть здійснювати перекази, обмінювати валюту, сплачувати за комунальні платежі та ін. Інтерфейс головної сторінки Raiffeisen Online (рис. 1.1.) виглядає так:

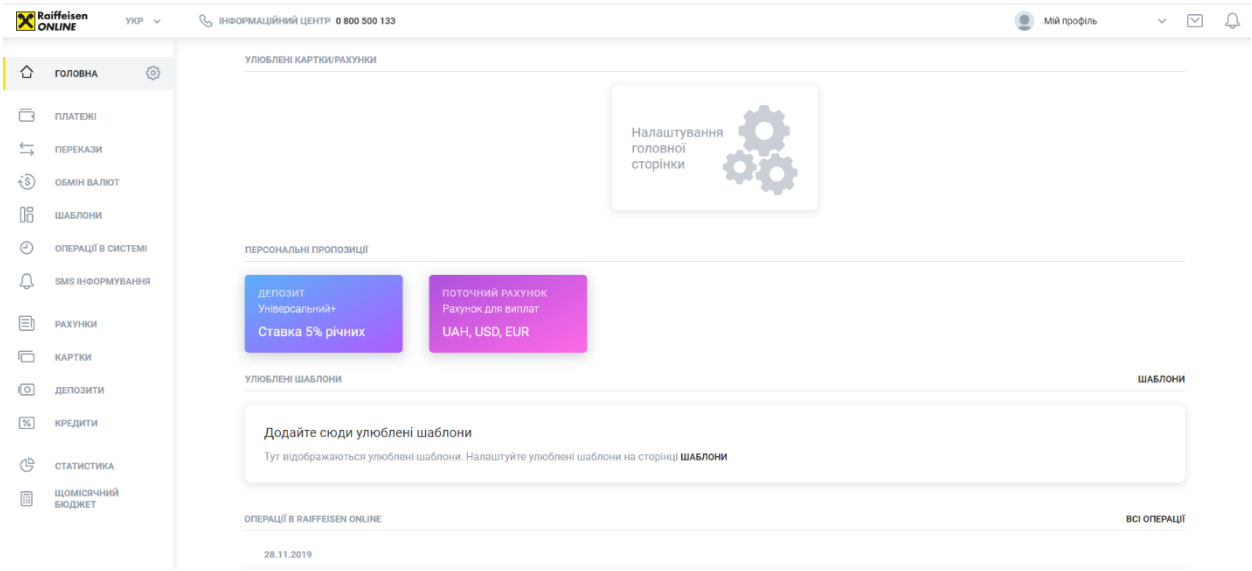

Рис. 1.1. Інтерфейс головної сторінки Raiffeisen Online

Основними перевагами Raiffeisen Online є:

- Можливість здійснювати усі основні операції за рахунком.
- Можливість керувати одразу кількома своїми рахунками.

Основними недоліками є те, що немає можливості подивитись отримувача переказу коштів, та не дуже зрозумілий інтерфейс, оскільки іноді важко знайти потрібну функцію.

Private24 – система інтернет банкінгу одного з найбільших банків України. На головній сторінці сайту є усе необхідне для користування послугами банку. Інтерфейс головної сторінки Private24 (рис. 1.2.) виглядає так:

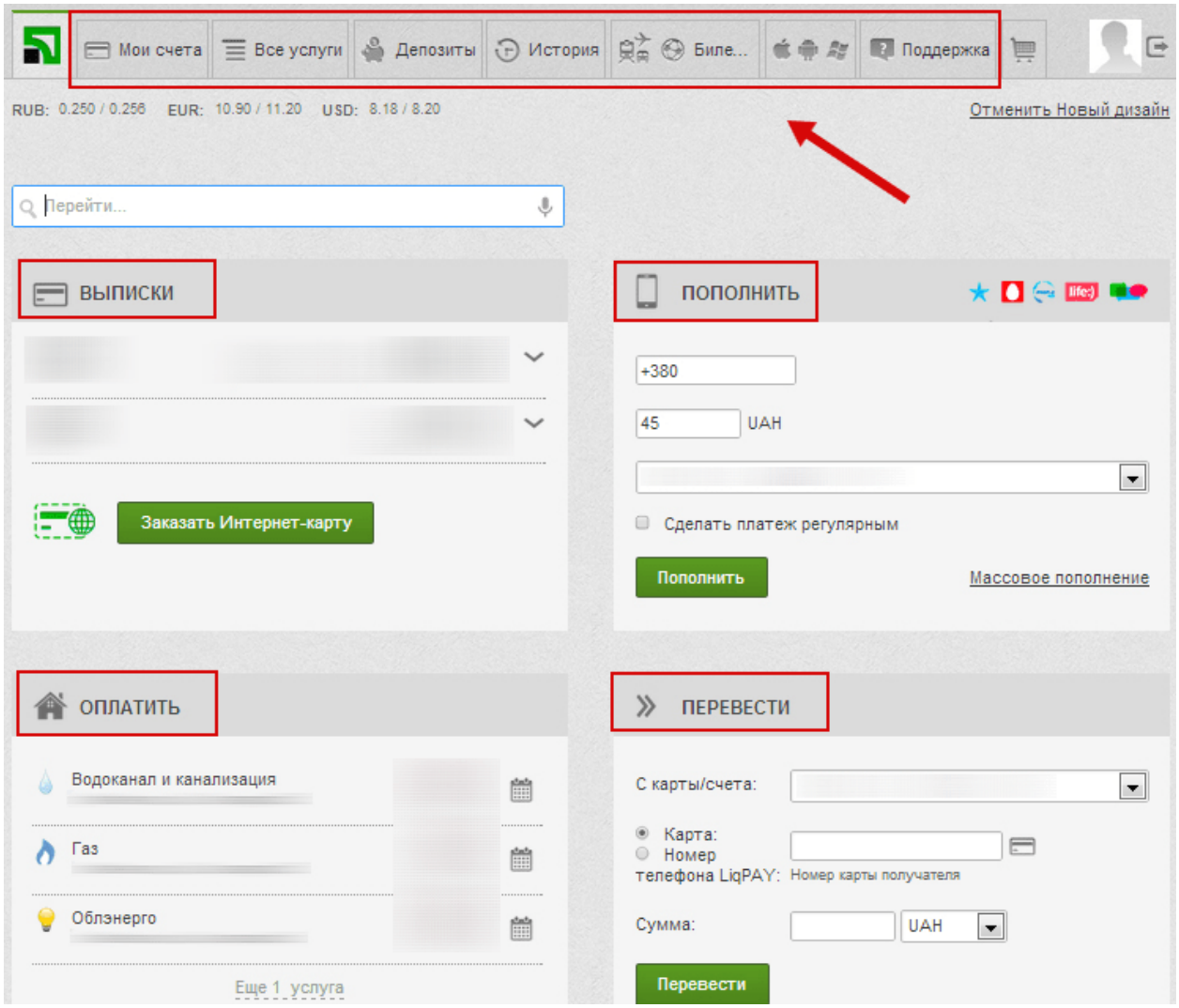

Рис. 1.2. Інтерфейс головної сторінки Private24

Головною перевагою цього сайту є те, що усі елементи інтерфейсу та базові функції знаходяться сукупно на головній сторінці сайту, тож кожен клієнт без труднощів зможе зробити усі потрібні операції.

Проаналізувавши усі дані був створений прототип бази даних розроблюваного веб-застосунку(Рис 1.3).

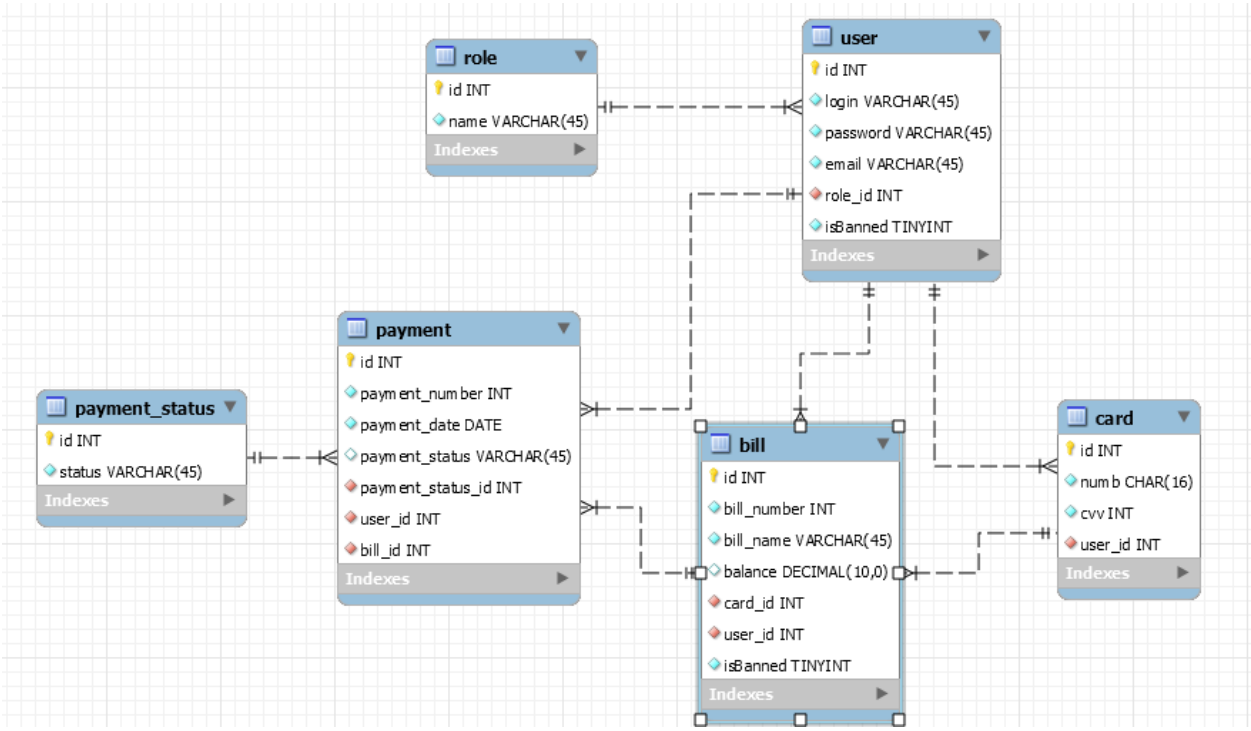

Рис. 1.3. Прототип бази даних.

1.2 Призначення розробки та область застосування.

Темою бакалаврської дипломної роботи виступає: «Розробка автоматизованої інформаційної системи для інтернет-банкінгу». Головною метою роботи є створення веб-сайту, за допомогою якого клієнт банку буде мати змогу керувати своїм рахунком. Веб-додаток має бути зручним у виконанні, та інтуітивно зрозумілим для користувача.

Система призначена для:

- Переказів коштів.
- Управління власним рахунком.
- Перегляд залишку на рахунку.

Система позиціонується як веб-додаток, який дає можливість клієнту банку використовувати особистий кабінет додатку, для здійснення платежів.

1.3 Підстава для розробки

Відповідно до ОКХ та ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОКХ та ОПП за напрямом підготовки 6.050101 «Комп'ютерні науки»;
- Графік навчального процесу та навчальний план;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № _____ від __.__.2021 р;
- завдання на дипломний проект на тему «Розробка автоматизованої інформаційної системи для інтернет-банкінгу».

1.4 Постановка завдання

Метою дипломного проекта є розробка веб-додатку, для системи інтернет-банкінгу. Система призначена для швидкого управління власним рахунком.

Веб-додаток повинен мати функції:

1. Перевірка статусу рахунку.
2. Створення платежу.
3. Перевірка статусу платежу.
4. Особистий кабінет.

Для виконання проекту необхідно:

- Розробити структуру системи.

- Розробити зручний інтерфейс додатку.
- Реалізувати front-end та back-end системи.

1.5 Вимоги до програми або програмного виробу.

1.5.1. Вимоги до функціональних характеристик.

Для досягнення поставлених цілей, розроблене програмне забезпечення повинно підтримувати виконання таких дій:

- Авто-збереження змін здійснених користувачем.
- Оновлення інформації у режимі онлайн.
- Зручна навігація за сайтом.
- Читання та оновлення таблиць СУБД з даними про рахунки.

Для повноцінного функціювання вище заявлених дій у додатку має бути реалізовано:

- Підтримка веб-браузера та доступ до програми через нього.
- Програмна та апаратна сумісності.
- Стандартна конфігурація за допомогою якої застосунок приводиться до експлуатації.

1.5.2 Вимоги до інформаційної безпеки.

Для коректної роботи програми потрібно реалізувати:

- Валідація вхідних даних.
- Можливість редагування даних.
- Збереження цілосності даних.
- Політика прав доступу.

Згідно з завданням, треба обмежити доступ клієнта до функцій і інформації, які доступні адміністратору інтернет-банкінгу.

1.5.3 Вимоги до складу та параметрів технічних засобів.

Для функціонування системи необхідно, щоб обчислювальна машина, на якій буде використовуватись додаток володіла наступними мінімальними характеристиками:

- Маніпулятор “миша”.
- Клавіатура.
- Доступ до мережі інтернет.
- 1 Гб вільного місця на жорсткому диску.
- Процессор Intel® Core™ i5-4200 2,70 GHz.
- Не менше 4 Гб операційної пам’яті.
- Монітор з діагоналлю 15".

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

1.5.4 Вимоги інформаційної та програмної сумісності.

Для корректного функціонування програми необхідне наступне програмне забезпечення, встановлене на обчислювальній машині:

- Веб браузер Google Chrome або Opera.
- Операційна система Windows 7/10.

Back end веб-додатку має бути реалізований на мові програмування Java з використанням Java EE. Для реалізації front end було використано CSS та HTML.

Висновки:

Інформаційна система інтернет банкінгу може бути корисна у роботі з клієнтами. У результаті чого буде покращено ефективність та якість наданих послуг. При розробці даної системи треба використовувати сучасні методи розробки, для покращення вражень користувача від користування веб-застосунком.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Функціональне призначення системи

Система призначена для клієнтів, які за допомогою створеного веб-додатку, зможуть передивлятися інформацію, та керувати власними рахунками.

Розроблений у ході виконання роботи веб-додаток має наступні функції:

- зручний інтерфейс для управління рахунком;
- форму логіну та реєстрації для користувачів;
- панель адміністратора сайту для перегляду усіх користувачів;
- функціональні кнопки для адміністратора сайту, з можливістю заблокувати користувача;
- можливість перегляду створених платежів;
- можливість блокування або розблокування власних рахунків;
- можливість формування платежів;

Сайт може бути використаний підприємством, якому потрібно переказувати кошти, та вести облік та статистику.

2.2 Опис застосованих математичних методів

Оскільки особливості предметної області ІС для інтернет банкінгу не передбачають застосування математичних методів при розробці, то математичні методи не були використані.

2.3 Опис використаних технологій та мов програмування

Програма була розроблена за допомогою мови програмування Java. У якості середовища розробки була використана IntelJIdea 2020 Ultimate edition. Java – це мова програмування, яка використовує об'єктно-орієнтований підхід для створення програм. Також, Java використовується для розробки як мережевних, так і мобільних та додатків для ПК.

Java була використана для написання серверної частини програми, для реалізації інтерфейсу користувача були використані HTML та CSS. Для їх інтеграції у програму була використана технологія JSP. JSP – це технологія, за допомогою якої можна створювати динамічні веб-сторінки. JSP це поєднання html-коду та Java коду. Завдяки цьому після створення запиту, сервер оброблює і генерує html-код, який відправляється користувачу.

Також був використаний плагін Lombok – це плагін для генерування коду за допомогою анотацій. Завдяки цьому плагіну, значно зростає продуктивність виробничого процессу, оскільки він допомагає економити час розробки.

Для обробки запитів була використана технологія Java EE – сервлети. Сервлети – це спеціальний тип класу Java який здійснюється на веб-сервері. Функціональне призначення сервлетів – це обробка запитів які до нього надходять, та повертає результат їх обробки.

У для створення, та управління базою даних була використана програма MySQL. MySQL – це система для управління реляційними базами даних з моделлю клієнт-сервер. БД – це структурований набір даних. Реляційна база данних, являє собою базу данних у якій дані представлені у вигляді таблиць, які зв'язані між собою певним чином.

2.4 Опис структури системи та алгоритмів її функціонування

Структура проекту була розроблена за стандартами Java EE. Використовувалась схема для розділення даних додатку – MVC. MVC – розділяє дані додатку на три основних компоненти:

- модель;
- відображення;
- контроллер;

Модель являє собою дані, та змінює своє становище, відповідно до вказівок контролера. Відображення відповідає за показ даних моделі користувачу, та, відповідно, реагує на зміни моделі. Контролер реагує на дії користувача, та відправляє запит моделі для її оновлення, відповідно до дій користувача.

На рис. 2.1. наведена структура файлів розробленого проекту.

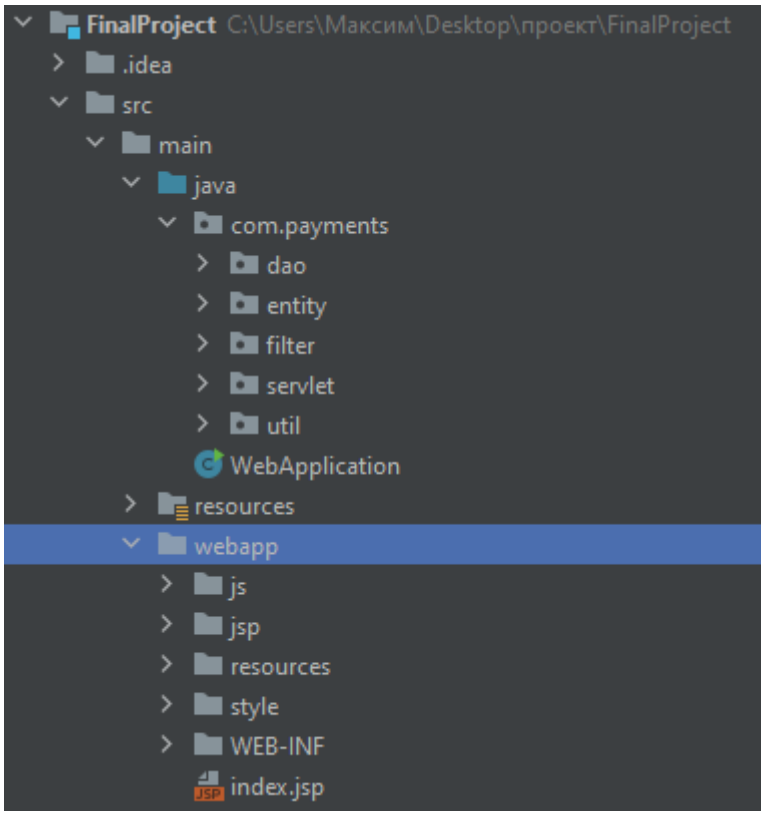

Рис. 2.1. Основна структура файлів проекту

На рис. 2.2. зображена структура файлів пакету DAO. DAO – це приклад проектування, за допомогою якого клієнт не працює напряму з сховищем даних. Це робиться для обмеження доступу до підсистем. Ще одною перевагою архітектури побудованою за принципом DAO є те, що він приховує деталі реалізації доступу до джерела даних. Архітектура патерна DAO зображена на рис. 2.3.

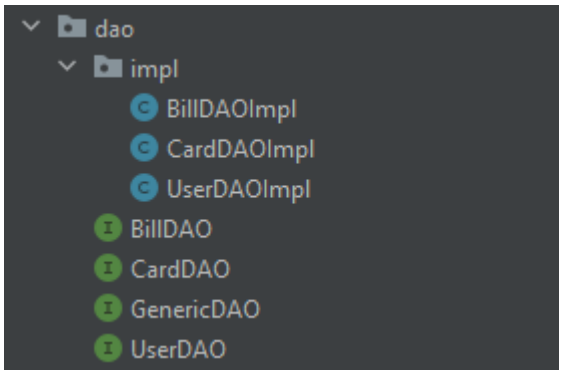

Рис. 2.2. Структура DAO

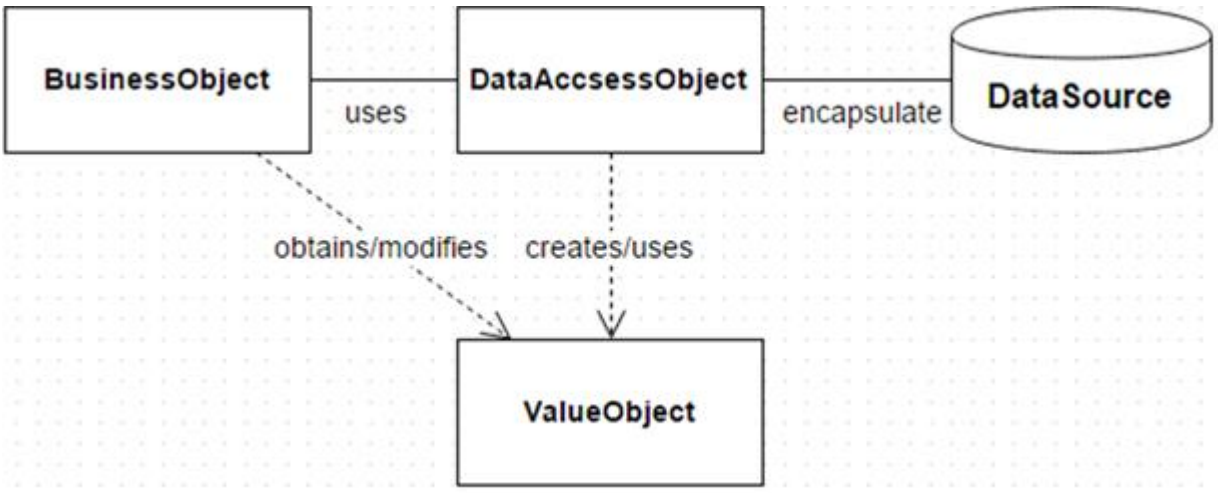

Рис. 2.3. Архітектура патерна DAO

На рис. 2.4. зображена структура пакету entity. Entity це об’єкт, який може бути ідентифікований певним способом, за допомогою якого його можна буде вирізняти серед інших об’єктів.

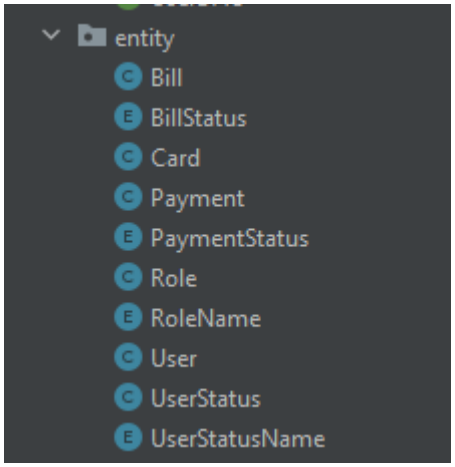

Рис. 2.4. Структура entity

На рис. 2.5. наведена структура фільтрів. Фільтри це Java клас, який оброблює відповіді сервера перед тим, як вони будуть відправлені клієнту. І також він оброблює запити від клієнта, перед тим як вони будуть надіслані до серверу.

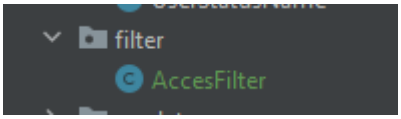

Рис. 2.5. Структура фільтрів

На рис. 2.6. зображена структура сервлетів, за допомогою яких відбувається обробка інформації. Java Servlet – це программа яка працює у додатку серверу, і працює як з'єднувач між запитом з веб-браузеру, та додатком на боці серверу. Використовуючи сервлети ви маєте можливість отримати вхідну інформацію від користувача через поля на веб-сторінках, від бази даних, або іншого ресурса, та створити динамічні веб-сторінки.

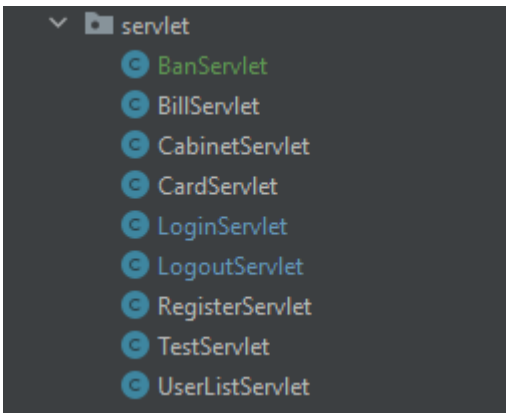

Рис. 2.6. Структура сервлетів

На рис. 2.7. зображена структура допоміжного пакету, який поєднує додаток з реляційною базою даних.

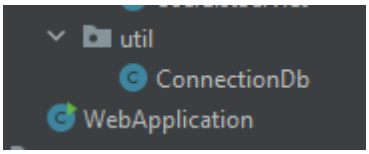

Рис. 2.7. Структура допоміжного пакету

На рис. 2.8. зображена структура пакету ресурсів. У них зберігаються специфічні ресурси, за допомогою яких можна писати програми, які можуть бути з легкістю локалізовані чи перекладені на різні мови.

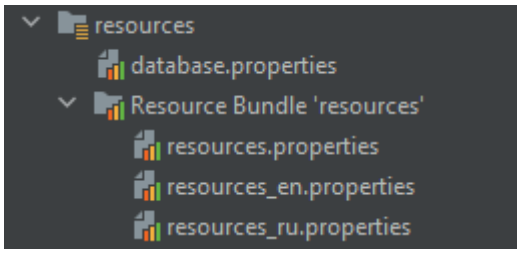

Рис. 2.8. Структура пакету ресурсів

На рис. 2.9. зображена структура пакету JSP. У цьому пакеті знаходяться файли, за допомогою яких відтворюється інтерфейс користувача.

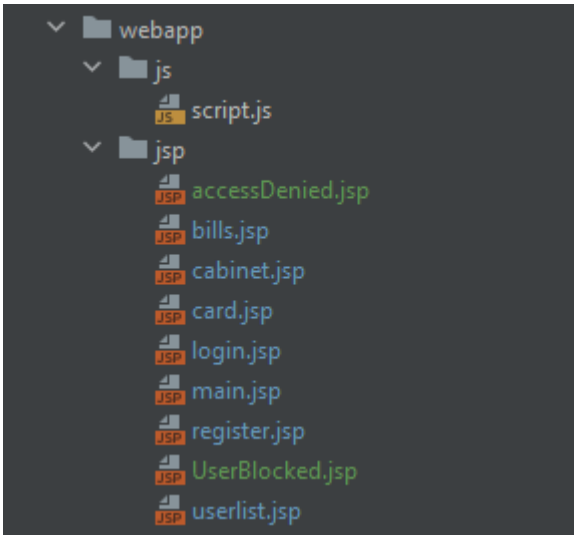

Рис. 2.9. Структура пакету JSP

На рис. 2.10. відображена структура пакету ресурсів, у якому зберігаються усі потрібні ресурси, які використовуються для створення інтерфейсу.

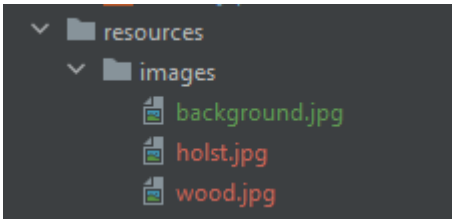

Рис. 2.10. Структура пакету ресурсів

На рис. 2.11. відображена структура пакету стилів CSS. CSS — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки від вигляду документу (що описується в CSS) [3].

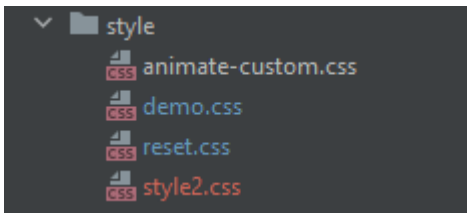

Рис. 2.11. Структура пакету стилів

На рис. 2.12. зображена структура пакету бібліотек, потрібних для коректного функціонування веб-частини додатку.

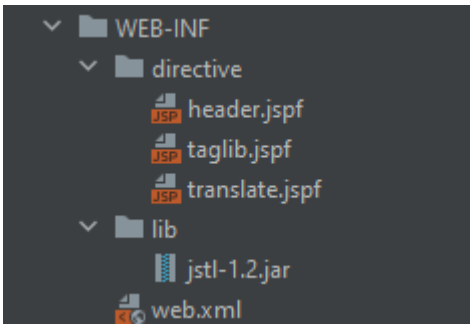

Рис. 2.12. Структура пакету бібліотек

У ході створення ІС, була спроектована та розроблена реляційна БД. Також БД була нормалізована згідно з стандартами проектування. Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей. Нормальна форма — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення [4].

На рис. 2.13. зображена спроектована БД.

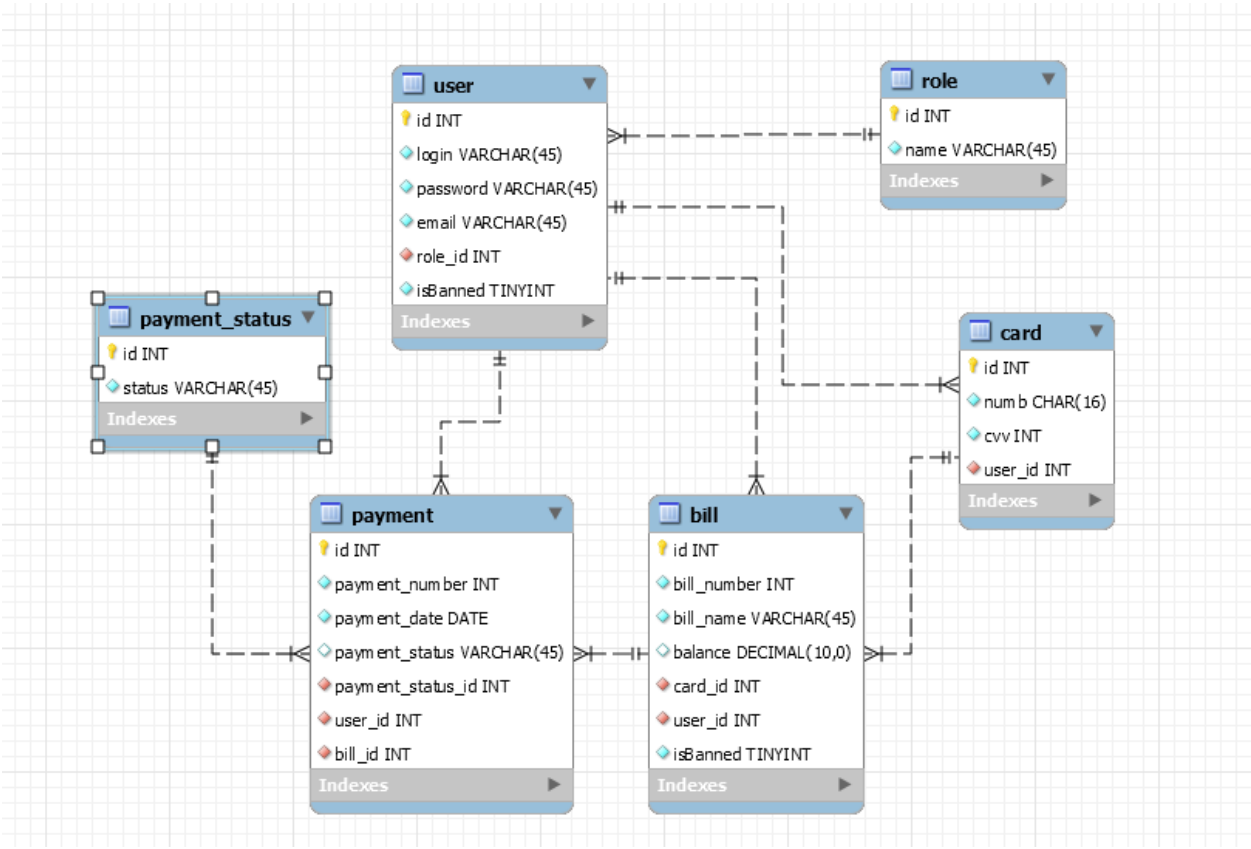

Рис. 2.13. Спроектована БД

База даних складається з 4 основних таблиць та 2 додаткових.

У таблиці user зберігається інформація про користувачів додатку, яку вони надають при реєстрації.

У таблиці payment зберігається інформація пло платежі, які були здійснені користувачами.

У таблиці bill знаходиться інформація про рахунки користувачів та їх баланс.

У таблиці card зберігається платіжна інформація користувачів.

Таблиця role слугує для того, щоб при користуванні можна було відрізняти звичайного користувача від адміністратора.

Таблиця payment_status слугує для визначення статусу платіжу користувача.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Згідно з цілями, та методами розробки ІС у якості вхідних даних, у розробленому додатку, використовуються дані, які передаються з БД. Також використовуються дані, які вводить користувач, через клавіатуру у веб-інтерфейсі.

Вихідні дані являють собою інформацію, яка відображається на веб-сторінках у браузері.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

.....

Розроблена програма може виконуватись на пристроях з операційною системою Windows, на різних типах персональних комп’ютерів чи ноутбуків.

Для запуску серверу, потрібний пристрій який зможе забезпечити належну продуктивність для стабільної роботи. У нього мають бути наступні мінімальні характеристики:

- Доступ до мережі інтернет.
- 1 Гб вільного місця на жорсткому диску.
- Процессор Intel® Core™ i5-4200 2,70 GHz.
- Не менше 4 Гб операційної пам’яті.
- Монітор з діагоналлю 15".

2.6.2. Використані програмні засоби

Для запуску та використання програми потрібні наступні програмні засоби:

- Встановлене середовище виконання IntelliJ IDEA 2020;
- Встановлене середовище для використання та зміни БД MySQL Workbanch;
- Веб-браузер.

2.6.3. Виклик та завантаження програми

Для роботи з програмою, потрібно запустити сервер за допомогою програмного забезпечення IntelliJ IDEA 2020. Потрібно запустити файл з назвою “WebApplication”. Після цього треба через браузер перейти до браузеру на адресу “localhost:8080”.

Після здійснення цих маніпуляцій додаток готовий до експлуатації користувачем.

2.6.4. Опис інтерфейсу користувача

Після потрапляння на початкову сторінку додатку, користувачу пропонується увійти у систему використовуючи свій логін та пароль (рис. 2.14.).

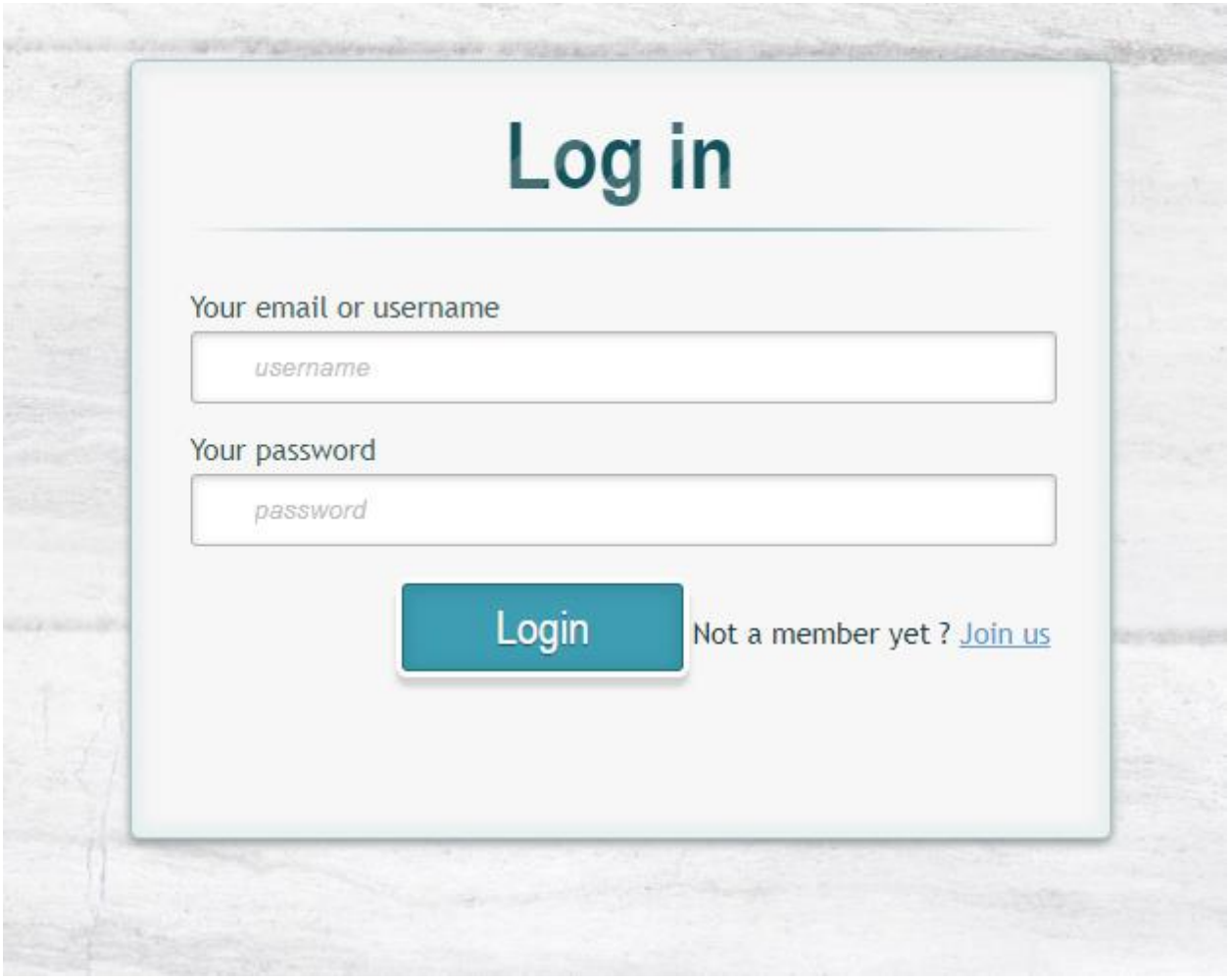

Рис. 2.14. Форма логіну

Код реалізації входу користувача до системи наведений нижче:

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        req.getRequestDispatcher("jsp/login.jsp").forward(req, resp);
    }
    private UserDAO userDAO = new UserDAOImpl();
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {

        String login = String.valueOf(req.getParameter("login"));
```

```
        String password = String.valueOf(req.getParameter("password"));
        User user;
        System.out.println(login+password);
        if (userDAO.findByLogin(login).isPresent()) {
            user = userDAO.findByLogin(login).get();
        } else {
            resp.sendError(422, "No user");
            return;
        }
        if (user.getPassword().equals(password)) {
            HttpSession session = req.getSession();
            session.setAttribute("login", login);
            session.setAttribute("userId",user.getId());
            if(user.getRole().getName().equals(RoleName.ADMIN.getRoleName()))
            {
                session.setAttribute("isAdmin",true);
            }
            if(user.getIsBanned() == true)
            {
                resp.sendRedirect("/loginCanceled");
                return;
            }
            resp.sendRedirect("/cabinet");
        }
        else {
            resp.sendError(422, "Incorrect password");
            return;
        }
    }
}
```

Якщо користувач введе некоректні дані, то веб-додаток виведе йому повідомлення про неправильність введених даних (рис. 2.15.).

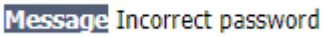

Рис. 2.15. Повідомлення про помилку

Також у разі, коли користувач буде заблокований у системі, йому надійде повідомлення про це(рис. 2.16.).

Рис. 2.16. Повідомлення про блокування

Якщо користувач ще не є зареєстрованим у системі, та не має власного логіну та паролю, він може перейти за посиланням, яке знаходиться у нижній частині форми(рис. 2.17.). Він перейде до нової сторінки на якій матиме змогу зареєструватися(рис. 2.18.).

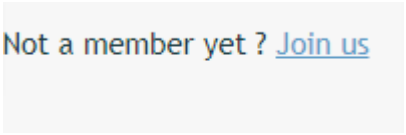

Рис. 2.17. Посилання на сторінку реєстрації

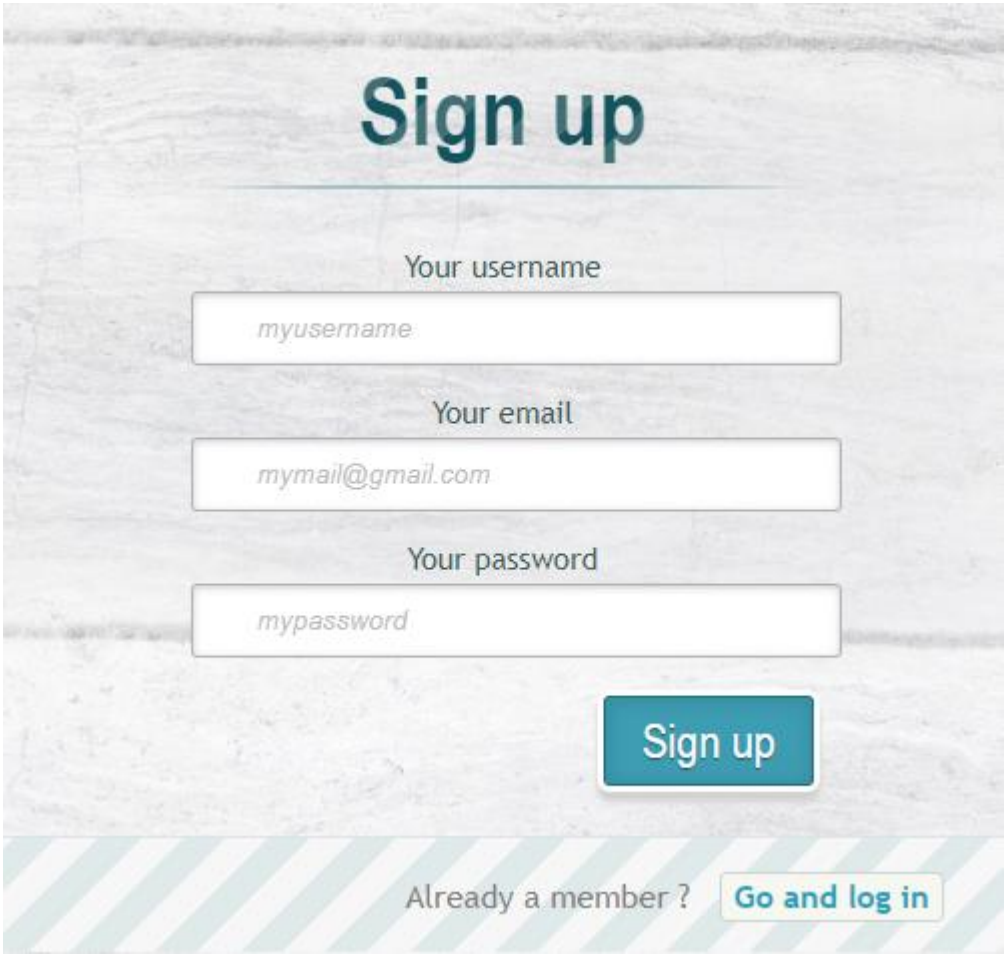

Рис. 2.18. Форма реєстрації

Програмний код, який забезпечує реєстрацію, наведений нижче:

```
@WebServlet("/registration")
public class RegisterServlet extends HttpServlet {
    UserDAO userDAO = new UserDAOImpl();
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.getRequestDispatcher("jsp/register.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
{
        String login = req.getParameter("login");
        String password = req.getParameter("password");
        String email = req.getParameter("email");
        if (login != null && password != null && email != null) {
```

```
            User user =User.builder()
                    .login(login)
                    .password(password)
                    .email(email)
                    .build();
            userDAO.create(user);
        }
        HttpSession session = req.getSession();
        session.setAttribute("login", login);
        resp.sendRedirect("/cabinet");
    }
}
```

Після реєстрації користувач потрапляє на домашню сторінку сайту, з якої здійснюється навігація за сторінками. У адміністратора сайту, та у користувача сайту будуть трохи різнитись кнопки навігації, оскільки у адміністратора сайту буде можливість переглянути усіх користувачів, та заблокувати або розблокувати їх(рис. 2.19.).

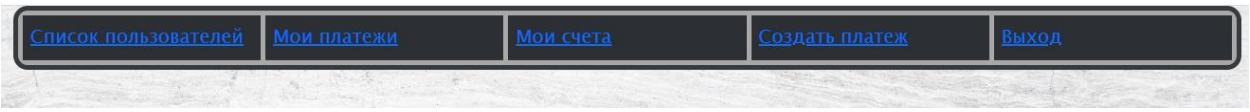

Рис. 2.19. Панель навігації для адміністратора

Рис. 2.20. Панель навігації для користувача

Для переходу на сторінку з списком усіх користувачів треба перейти за посиланням. На цій сторінці адміністратор зможе побачити список усіх користувачів, які користувались веб-додатком, і пройшли реєстрацію у ньому.

Також на сторінці відображаються статус користувача, та його роль: простий користувач або адміністратор (рис.2.21.).

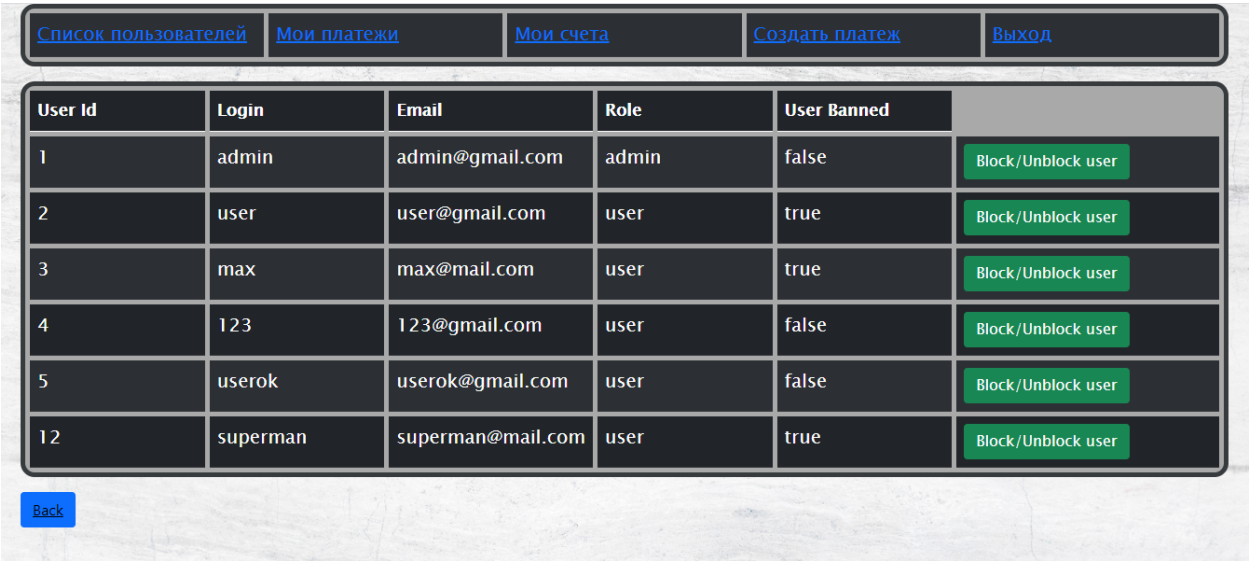

Рис. 2.21. Вигляд сторінки списку користувачів

У правій частині веб-сторінки знаходиться функціональна кнопка, за допомогою якої можна заблокувати чи розблокувати одного з користувачів. Після натиску кнопки, на сторінці змінюється статус користувача, та оновлюється інформація у БД. Зміна статусу користувача відображена на рис. 2.22.

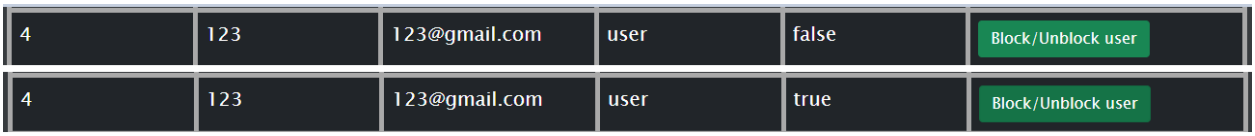

Рис. 2.22. Зміна статусу користувача

Для того щоб створити платіж, користувачу треба перейти за посиланням “Створити платіж”. У цьому розділі користувач може здійснити переказ коштів з свого особистого рахунку, на рахунок іншого користувача системи.

Після переходу за посиланням, користувач бачить меню, у яке треба ввести потрібну інформацію (рис. 2.23).

Рис. 2.23. Інтерфейс створення платіжу

Для створення платіжу користувачу потрібно ввести наступні дані:

- Номер особистого рахунку
- Рахунок на який треба перевести кошти
- Суму транзакції.

Після натиску на функціональну кнопку, формується запит та створюється платіж. Інформація оновлюється у БД та у розділі веб-інтерфейсу “Мої платежі”.

У розділі “Мої платежі” відображаються здійснені користувачем платежі.

Створюємо платіж(рис. 2.23.):

Рис. 2.23. Створення платежу

Після створення платіжу у списку, з’являється новий запис(рис. 2.24.):

Рис. 2.24. Веб-інтерфейс сторінки з платежу

І останній розділ “Мої рахунки” дозволяє користувачам переглянути інформацію за зареєстрованими ними рахунками(рис. 2.25.).

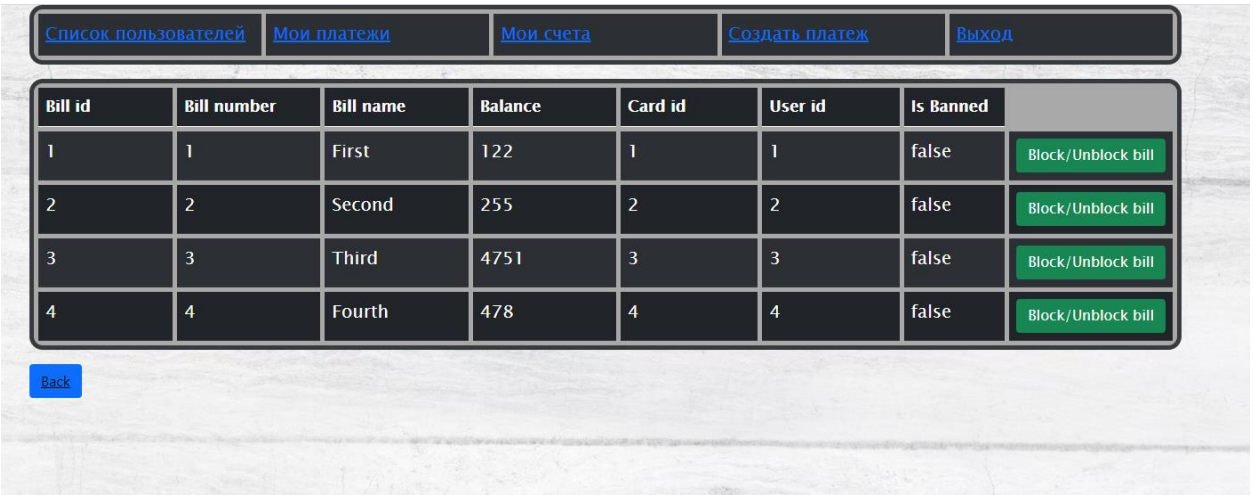

Рис. 2.25. Створені рахунки

Завдяки функціональній кнопці, яка знаходиться з правої частини форми(рис. 2.26.) можна заблокувати чи розблокувати один чи декілька своїх рахунків за потреби(рис. 2.27.).

Рис. 2.26. Кнопка блокування/розблокування рахунку

Bill id	Bill number	Bill name	Balance	Card id	User id	Is Banned	
1	1	First	122	1	1	false	Block/Unblock bill

Bill id	Bill number	Bill name	Balance	Card id	User id	Is Banned	
1	1	First	122	1	1	true	Block/Unblock bill

Рис. 2.27. Блокування рахунку

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані розробки програмного забезпечення:

а) передбачуване число операторів – 1453;

б) коефіцієнт складності програми – 1,35;

в) коефіцієнт корекції програми в ході її розробки – 0,064;

г) середня годинна заробітна плата програміста, грн/год – 107;

Середня годинна зарплата програміста була вирахувати виходячи з даних статистики зарплат професії "Junior Java developer в Україні". Середньоукраїнська заробітна плата програміста, який пише програми на мові програмування Java і з досвідом роботи близько двох місяців дорівнює 18939 грн в місяць. При восьмигодинному робочому дні (176 годин в місяць в середнбому) середня зарплата за годину буде становитися 107 грн.

д) коефіцієнт кваліфікації програміста, обумовлений від стажу – 0,8;

е) вартість машино-години ЕОМ, грн/год – 4,3.

3.1. Визначення трудомісткості розробки програмного забезпечення

Трудомісткість розробки ПЗ можна розрахувати за формулою:

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

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 60 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

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

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

$t_д$ – витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

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

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1453 \cdot 1{,}35 \cdot (1 + 0{,}064) = 2087{,}08$$

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

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

де B, яке дорівнює 1,35, – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k, яке дорівнює 0,8, – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t_u = \frac{2087{,}08 \cdot 1{,}35}{79 \cdot 0{,}8} = 44{,}58, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

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

$$t_a = \frac{2087{,}08}{22 \cdot 0{,}8} = 118{,}58, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20..25) \cdot k}, \tag{3.5}$$

$$t_n = \frac{2087{,}08}{23 \cdot 0{,}8} = 113{,}42, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}; \tag{3.6}$$

$$t_{отл} = \frac{2087,08}{4 \cdot 0,8} = 652,2, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,4 \cdot t_{отл}; \tag{3.7}$$

$$t_{отл}^{к} = 1,4 \cdot 652,2 = 913,08, \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}, \tag{3.8}$$

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

$$t_{др} = \frac{Q}{(15..20) \cdot k}; \tag{3.9}$$

$$t_{др} = \frac{2087,08}{20 \cdot 0,8} = 130,44, \text{ людино-годин.}$$

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

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

$t_{до} = 0{,}75 \cdot 130{,}44 = 97{,}83$, людино-годин.

$t_{д} = 130{,}44 + 97{,}83 = 228{,}27$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

$t_{д} = 60 + 44{,}58 + 118{,}58 + 113{,}42 + 652{,}2 + 228{,}27 = 1217{,}05$, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1217,05 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного для налагодження програми на ЕОМ.

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

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

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

де t – загальна трудомісткість, людино-годин,

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

$$З_{ЗП} = 1217{,}05 \cdot 107 = 13022{,}35, \text{ грн.}$$

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

$$З_{МВ} = t_{отл} \cdot C_{МЧ}, \text{ грн}, \quad (3.13)$$

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

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 652{,}2 \cdot 4{,}3 = 2804{,}46, \text{ грн},$$

$$К_{ПО} = 13022{,}35 + 2804{,}46 = 13302{,}81, \text{ грн.}$$

Очікуваний період створення ПЗ:

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

де B_k – число виконавців,

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

$$T = \frac{1217{,}05}{1 \cdot 176} \approx 6{,}91 \text{ міс.}$$

Висновки: для розробки даного програмного забезпечення потребується 1217,05 людино-годин. Таким чином, очікувана тривалість розробки складе 6,91 місяця при 40 годинному робочому тижні і 176-годинному робочому місяці, а витрати на створення програмного забезпечення складатимуть 133028,81 грн.

ВИСНОВКИ

Метою дипломного проекту є створення веб-додатку, за допомогою якого клієнт зміг би керувати своїми рахунками, переглядати за ними інформацією та здійснювати платежі. Основною мовою розробки було використано Java, оскільки ця мова розробки є зручною, та досить популярна на ринку технологій.

Програма може бути корисна для людей, яким потрібно вести лік своїм фінансовим операціям, створювати платежі та переглядати інформацію за своїми рахунками.

Для досягнення мети були вирішені наступні завдання:

- аналіз предметної області;
- уточнення вимог до розроблюваного веб-застосунку;
- уточнення вимог до форматів програми;
- розробка алгоритмів аналізу та обробки даних.

Відповідно до проведеного аналізу та завдання в роботі поставлено та вирішено всі функціональні задачі та вимоги до розробленого веб-додатку, а саме:

- зручний і інформативний графічний інтерфейс;
- можливість формування платежу;
- можливість перегляду особистих платіжних даних;
- можливість управління власним рахунком;
- можливість реєстрації та входу до особистого кабінету;

Визначено трудомісткість розробленої інформаційної системи (1217,05 людино-годин), проведений підрахунок вартості роботи по створенню програми (133028,81 грн) та розраховано час на його створення (6,91 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Середня заробітна плата Junior Java developer в Украине [Електронний ресурс]. URL: https://ua.trud.com/salary/2/3334.html (дата звернення 09.06.2021).
2. Розрахунок вартості машино-години ЕОМ [Електронний ресурс]. URL: https://studbooks.net/1786806/geografiya/raschet_stoimosti_mashino_chasa (дата звернення 09.06.2021).
3. CSS каскадні таблиці стилів [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення 30.05.2021).
4. Інформація про нормалізацію БД [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85 (дата звернення 18.05.2021).
5. Паттерн DAO у мові Java [Електронний ресурс]. URL: https://ua-blog.com/%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80-%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8-%D0%BF%D0%B0%D1%82%D0%B5%D1%80%D0%BD%D0%B0-dao-%D0%B8-vo-%D0%B2-java (дата звернення 15.05.2021).
6. Герберт Ш. Java. Полное руководство / Шилдт Герберт. – Київ: Вільямс, 2015. – 1377 с.
7. Блох Д. Java эффективное программирование / Джошуа Блох. – Москва: Диалектика, 2019. – 466 с. – (Третье издание).
8. JDBC. Засіб для роботи з БД [Електронний ресурс] – URL: https://proselyte.net/tutorials/jdbc/introduction/ (дата звернення 02.05.2021).

9. JSP. Засіб створення веб-сторінок. [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/jsp/index.htm (дата звернення 07.05.2021).
10. Розробка веб-додатків, за допомогою мови Java. [Електронний ресурс] – Режим доступу до ресурсу: https://tproger.ru/translations/building-a-web-app-with-java-servlets/ (дата звернення 27.04.2021).
11. Створення та проектування БД. [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/62415/menedzhment/proektuvannya_bazi_danih (дата звернення 20.04.2021).
12. Створення веб-сторінки з використанням каскадних аркушів стилю [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/krasnopolska68/rabota/distancijne-navcanna/k-191-informatika/pz-no-28-stvorenna-veb-storinki-z-vikoristannam-kaskadnih-arkusiv-stilu (дата звернення 22.04.2021).
13. Підключення додатку на Java до БД. [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/database/2.2.php (дата звернення 06.05.2021).
14. Інтеграція робочої середи з Tomcat [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/javaee/2.3.php (дата звернення 18.05.2021).
15. Створення Git-репозиторія. [Електронний ресурс] – Режим доступу до ресурсу: https://git-scm.com/book/uk/v2/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-Git-%D0%A1%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-Git-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D1%96%D1%8F (дата звернення 10.04.2021).

16. Мова SQL. [Електронний ресурс] – Режим доступу до ресурсу: https://www.ukraine.com.ua/uk/blog/programming/sql-baza-dannih-dlya-chego-prednaznachena-baza-dannih.html (дата звернення 25.04.2021).
17. Патерни проектування у Java. [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/ru/design-patterns/java (дата звернення 07.04.2021).
18. Обробка помилок. [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/javaee/4.7.php (дата звернення 07.05.2021).
19. Мова програмування JAVA та платформа JAVAFX [Електронний ресурс] – Режим доступу до ресурсу: http://www.dut.edu.ua/ua/news-1-626-6031-mova-programuvannya-java-ta-platforma-javafx-prikladi-zastosuvannya_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення 01.04.2021).
20. Додання бібліотек у проект Intellij IDEA. [Електронний ресурс] – Режим доступу до ресурсу: https://javadevblog.com/kak-dobavit-biblioteku-jar-fajl-v-proekt-intellij-idea.html (дата звернення 04.05.2021).

ДОДАТОК А

КОД ПРОГРАМИ

BillDAOImpl.java

```
public class BillDAOImpl implements BillDAO {

//    private static final String SQL ="SELECT b.id, b.bill_number,b.bill_name, b.isBanned,
card.numb,user.login FROM bill as b\n" +

//            "JOIN card On b.card_id=card.numb\n" +

//            "JOIN user On b.user_id=user.login";

private static final String SQL ="SELECT b.id, b.bill_number,b.bill_name,b.balance, b.card_id,b.user_id,
b.isBanned FROM bill as b\n";

    private static final String BILL_STATUS_UPDATE = "UPDATE bill SET isBanned=? WHERE id=?";



    private final Connection connection = ConnectionDb.getConnection();


    @Override
    public Optional<Bill> findById(Long aLong) {
        return Optional.empty();
    }


    @Override
    public List<Bill> findALL() {
        String sql = SQL;
        List<Bill> bills = new ArrayList<>();
        try (Statement statement = connection.createStatement()) {
            ResultSet resultSet = statement.executeQuery(sql);
            while (resultSet.next()) {
                bills.add(
                        new Bill(resultSet.getLong("id"),
                        resultSet.getInt("bill_number"),
                        resultSet.getString("bill_name"),
                        resultSet.getBigDecimal("balance"),
                        resultSet.getLong("card_id"),
                        resultSet.getLong("user_id"),
                        resultSet.getBoolean("isBanned")));
            }
            return bills;
```

```
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

@Override
public Bill create(Bill entity) {
    return null;
}

@Override
public Bill update(Bill entity, Long aLong) {
    return null;
}

@Override
public void deleteById(Long aLong) {

}

@Override
public void delete(Bill entity) {

}

@Override
public void updateBillStatus(Long id, boolean isBanned) {
    String sql = BILL_STATUS_UPDATE;
    try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
        preparedStatement.setLong(2,id);
        preparedStatement.setBoolean(1,isBanned);
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

```
}
```

CardDAOImpl.java

```
package com.payments.dao.impl;

public class CardDAOImpl implements CardDAO {
    private final Connection connection = ConnectionDb.getConnection();
    private static final String SELECT_FROM_CARD_WHERE_USER_LOGIN="SELECT c.id,
c.numb,c.cvv,c.user_id FROM card as c WHERE c.user_id=?";
    @Override
    public Optional<Card> findById(Long aLong) {
        return Optional.empty();
    }
    @Override
    public List<Card> findALL() {
        return null;
    }
    @Override
    public Card create(Card entity) {
        return null;
    }
    @Override
    public Card update(Card entity, Long aLong) {
        return null;
    }
    @Override
    public void deleteById(Long aLong) {
    }
    @Override
    public void delete(Card entity) {
    }
    @Override
    public Optional<Card> findByLogin(Long id) {
        String sql = SELECT_FROM_CARD_WHERE_USER_LOGIN;
        try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setLong(1,id);
            ResultSet resultSet = preparedStatement.executeQuery();
            Optional<Card> card = Optional.empty();
            while (resultSet.next()) {
                card = Optional.ofNullable(Card.builder()
```

```
                .id(resultSet.getLong("id"))
                .numb(resultSet.getString("numb"))
                .cvv(resultSet.getString("cvv"))
                .userId(resultSet.getLong("user_id"))
                .build());
        }
        return card;
      } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
      }
    }
  }
```

UserDAOImpl.java

```
package com.payments.dao.impl;
public class UserDAOImpl implements UserDAO {

    private static final String SELECT_FROM_USER_WHERE_USER_LOGIN = "SELECT u.id, u.login,
u.password, u.email,role.name, u.isBanned FROM user as u\n" +
            " join role On u.role_id=role.id  WHERE u.login=?\n";
    //join user_status On u.user_status_id=user_status.id
    private static final String SQL = "SELECT u.id, u.login, u.password, u.email,role.name, u.isBanned
FROM user as u \n" +
            "join role On u.role_id=role.id\n";
    private static final String USER_STATUS_UPDATE = "UPDATE user SET isBanned=? WHERE id=?";
    private final Connection connection = ConnectionDb.getConnection();

    @Override
    public Optional<User> findByLogin(String login) {
      String sql = SELECT_FROM_USER_WHERE_USER_LOGIN;
      try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
        preparedStatement.setString(1, login);
        ResultSet resultSet = preparedStatement.executeQuery();
        Optional<User> user = Optional.empty();
        while (resultSet.next()) {
          user = Optional.ofNullable(User.builder()
                .id(resultSet.getLong("id"))
                .login(resultSet.getString("login"))
```

```
                .password(resultSet.getString("password"))
                .email(resultSet.getString("email"))
                .role(new Role(null, resultSet.getString("name")))
                .isBanned(resultSet.getBoolean("isBanned"))
                .build());
        }
        return user;
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
@Override
public Optional<User> findById(Long aLong) {
    return Optional.empty();
}
@Override
public List<User> findALL() {
    String sql = SQL;
    List<User> users = new ArrayList<>();
    try (Statement statement = connection.createStatement()) {
        ResultSet resultSet = statement.executeQuery(sql);
        while (resultSet.next()) {
            users.add(
                    new User(resultSet.getLong("id"),
                            resultSet.getString("login"),
                            resultSet.getString("password"),
                            resultSet.getString("email"),
                            new Role(resultSet.getLong("id"),
                                    resultSet.getString("name")),
                            resultSet.getBoolean("isBanned")));
        }
        return users;
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

```
@Override
public User create(User entity) {
    String sql = "insert into user (login, password, email, role_id, isBanned) value(?, ?, ?, ?,?)";
    try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
        preparedStatement.setString(1, entity.getLogin());
        preparedStatement.setString(2, entity.getPassword());
        preparedStatement.setString(3, entity.getEmail());
        preparedStatement.setLong(4, 2);
        preparedStatement.setLong(5, 1);
        preparedStatement.executeUpdate();
        return entity;
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
@Override
public User update(User entity, Long aLong) {
    return null;
}
@Override
public void deleteById(Long aLong) {
}
@Override
public void delete(User entity) {
}
@Override
public void updateUserStatus(Long id, boolean isBanned) {
    String sql = USER_STATUS_UPDATE;
    try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
        preparedStatement.setLong(2,id);
        preparedStatement.setBoolean(1,isBanned);
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
```

```
        }
    }
}
```

BillDAO.java

```
package com.payments.dao;
import com.payments.entity.Bill;
public interface BillDAO extends GenericDAO<Bill,Long>{
    void updateBillStatus(Long id,boolean isBanned);
}
```

CardDAO.java

```
import java.util.Optional;
public interface CardDAO extends GenericDAO<Card,Long>{
    Optional<Card> findByLogin(Long id);
}
```

GenericDAO.java

```
public interface GenericDAO<T, ID> {
    Optional<T> findById(ID id);
    List<T> findALL();
    T create(T entity);
    T update(T entity, ID id);
    void deleteById(ID id);
    void delete(T entity);
}
```

UserDAO.java

```
public interface UserDAO extends GenericDAO<User, Long> {
    Optional<User> findByLogin(String login);
    void updateUserStatus(Long id,boolean isBanned);
}
```

AddPayment.java

```
@WebServlet("/addBill")
public class AddPayment extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        req.getRequestDispatcher("jsp/addBill.jsp").forward(req, resp);
    }
    @Override
```

```
        protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
            super.doPost(req, resp);
        }
    }
```

BanServlet.java

```
@WebServlet("/loginCanceled")
public class BanServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        req.getRequestDispatcher("jsp/UserBlocked.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        super.doPost(req, resp);
    }
}
```

BillServlet.java

```
@WebServlet("/bills")
public class BillServlet extends HttpServlet {
   private final BillDAO billDAO=new BillDAOImpl();
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
      List<Bill> bills=billDAO.findALL();
      req.setAttribute("bills",bills);
      req.getRequestDispatcher("jsp/bills.jsp").forward(req, resp);
   }
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
{
      Long id=Long.valueOf(req.getParameter("id"));
      Boolean isBanned= Boolean.valueOf(req.getParameter("isBanned"));
      isBanned=!isBanned;
      billDAO.updateBillStatus(id,isBanned);
      resp.sendRedirect("/bills");
   }
}
```

CardServlet.java

```
@WebServlet("/card")
public class CardServlet extends HttpServlet {
    private final CardDAO cardDAO=new CardDAOImpl();
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        Long userId = Long.valueOf(String.valueOf((req.getSession().getAttribute("userId"))));
        Card card;
        card=cardDAO.findByLogin(userId).orElseThrow(RuntimeException::new);
        req.setAttribute("card",card);
        req.getRequestDispatcher("jsp/card.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
{
        super.doPost(req, resp);
    }
}
```

LogutServlet.java

```
@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        HttpSession session = req.getSession();
        if(session.getAttribute("login")!=null){
            session.invalidate();
        }
        req.getRequestDispatcher("jsp/login.jsp").include(req, resp);
        out.close();
    }
    }
```

UserListServlet.java

```
@WebServlet("/userlist")
public class UserListServlet extends HttpServlet {
    private final UserDAO userDAO=new UserDAOImpl();
```

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        List<User> userList=userDAO.findALL();
        req.setAttribute("userlist",userList);
        req.getRequestDispatcher("jsp/userlist.jsp").forward(req, resp);


    }


    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
{
        Long id=Long.valueOf(req.getParameter("id"));
        Boolean isBanned= Boolean.valueOf(req.getParameter("isBanned"));
        isBanned=!isBanned;
        userDAO.updateUserStatus(id,isBanned);
        resp.sendRedirect("/userlist");
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Появнювальні документи	
Kapusta_Diplom.docx	Пояснювальна записка кваліфікаційної роботи. Документ Word.
Kapusta_Diplom.pdf	Пояснювальна записка кваліфікаційної роботи в форматі PDF.
Програма	
JavaWeb.zip	Архів. Містить коди програми.
Презентація	
Kapusta_Diplom.pptx	Презентація кваліфікаційної роботи.