Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення для касового апарату»

Виконав: студент IV курсу
групи 3ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Шинкарчук О.А
(прізвище та ініціали)
Керівник: д.т.н., проф. каф.ПЗ Ліщинська Л.Б.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ПЗ Іванчук Я.В.
(прізвище та ініціали)

Допущено до захисту

Завідувач кафедри _____Романюк О. Н.

«___»_____2022 р.

ВНТУ– 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
Романюк О. Н.
_____
"_25_" __березня__ 2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Шинкарчуку Олегу Андрійовичу

1. Тема роботи – «розробка програмного забезпечення для касового апарату».
Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., професор кафедри ПЗ, затверджені наказом вищого навчального закладу від " 24 " березня 2022 року № 66 .
2. Строк подання студентом роботи " 13 червня 2022 р.
3. Вихідні дані до роботи: модель розробки – водоспадна; архітектурний шаблон проєктування – MVC; система управління базами даних – MySQL; мова запитів – SQL; вхідні дані – база даних із товарами їх кількість, замовлення та їх статус, робітники та їх посада; вихідні дані – звіт по розрахунковим послугам; середовище розробки – IntelliJ IDEA; мова програмування – Java.
4. Зміст розрахунково–пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структури, методу та алгоритмів роботи програмного продукту; розробка програмних компонент для касового апарату; тестування програми; висновки; список використаних джерел; додатки.
5. Перелік графічного матеріалу: актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження; структура графічного інтерфейсу головної сторінки веб-додатку; блок-схема алгоритму шифрування даних; блок-схема алгоритму генерування звіту.

6. Консультанти розділів роботи

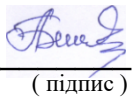
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ліщинська Л.Б., д.т.н., професор кафедри ПЗ		

7. Дата видачі завдання ___25 березня 2022 року___

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	26.03.2022 - 02.04.2022	Вик.
2	Розробка методу генерування звіту про розрахункові операції.	03.04.2022 - 07.04.2022	Вик.
3	Розробка методу шифрування і дешифрування інформації.	08.04.2022 - 12.04.2022	Вик.
4	Аналіз і вибір середовища та мови розробки	13.04.2022 - 17.04.2022	Вик.
5	Програмна реалізація вебдодатка	18.04.2022 - 25.05.2022	Вик.
6	Тестування програми	26.05.2022 - 28.05.2022	Вик.
7	Оформлення матеріалів до захисту БДР	29.05.2022 - 10.06.2022	Вик.

Студент _____ **Шинкарчук О.А.**
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи  **Ліщинська Л.Б.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 94 сторінок формату А4 містить 30 рисунків та 7 таблиць, список використаних джерел містить 24 найменуваннь.

У бакалаврській дипломній роботі проведено детальний аналіз стану програмних реалізацій для реєстрації розрахункових операцій. Визначено об’єкт, предмет, завдання та методи дослідження. Сформульовано мету дослідження – підвищення ефективності фінансових операцій які здійснюються касовим апаратом шляхом інтеграції удосконалених алгоритмів збереження інформації, надійне шифрування конфіденційної інформації та створення звітів для фінансового моніторингу. Для реалізації мети було розроблено веб-додаток касового апарату.

Запропоновано метод та алгоритм для шифрування та дешифрування даних, формування звіту на основні здійснених розрахункових операцій.

Касовий апарат розроблено з використанням мови програмування Java, контейнером Apache Tomcat для роботи з сервлетами java, фреймворком Bootstrap для графічного інтерфейсу та середовища розробки IntelliJ IDEA. Для розробки бази даних використана мова програмування SQL та система управління базами даних MySQL. В результаті виконання бакалаврської дипломної роботи розроблено програмний засіб, працездатність і правильність роботи якого перевірено також підготовлена інструкція користувача.

Отримані в бакалаврській дипломній роботі результати можна використати для заміни класичного апарату його програмним аналогом.

Ключові слова: касовий апарат; реєстратор розрахункових операцій

ANNOTATION

The bachelor's thesis consists of 94 A4 pages containing 30 figures and 7 tables, the list of used sources contains 24 titles.

In the bachelor's thesis a detailed analysis of the state of software implementations for the registration of settlement operations. The object, subject, tasks and research methods are defined. The purpose of the study is to increase the efficiency of financial transactions carried out by the cash register by integrating advanced algorithms for storing information, reliable encryption of confidential information and creating reports for financial monitoring. To achieve this goal, a software implementation of the cash register was developed.

A method and algorithm for encrypting and decrypting data, generating a report and a check for the main performed settlement operations are proposed.

The cash register is designed using the Java programming language, the Apache Tomcat container for working with java servlets, the Bootstrap framework for the graphical interface and the IntelliJ IDEA development environment. SQL programming language and MySQL database management system were used to develop the database. As a result of the bachelor's thesis, a software tool was developed, the efficiency and correctness of which was checked and a user manual was prepared.

The results obtained in the bachelor's thesis can be used to replace the classic device with its software counterpart.

Key words: cash register; registrar of settlement operations

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Впровадження новітніх програмних рішень на заміну вузькоспеціалізованим апаратам призводить до значної економії коштів і підвищення ергономічності [1]. Як прийнято, облік фінансових операцій відбувається за допомогою апаратного реєстратора розрахункових операцій (далі – РРО), який є узагальненням для касових апаратів та після внесення змін до чинного законодавства з'явився новий тип – програмний РРО [2]. Використання програмного РРО дозволяє підприємствам значною мірою пришвидшити надання своїх послуг, оскільки реєстрація в програмному варіанті займає лічені години, на відмінну від апаратного рішення процес, встановлення якого може тривати від кількох днів до місяця.

За статистикою більшість касирів не мають спеціалізованої освіти, тому використання звичайного касового апарату викликає певні труднощі як при навчанні, так і при подальшому використанні. Програмний касовий апарат являє собою спеціальну комп'ютерну програму, яку можна використовувати в ноутбуці, смартфоні чи планшеті, які мають доступ до Інтернету для зв'язку з податковою службою. Також підвищується мобільність такого рішення оплати фінансових операцій, адже класичний варіант має достатньо великі габарити та потребує постійного з'єднання з мережею. З'являється можливість використання будь-якого принтера для роздрукування чеку, також це дає змогу підключити декілька пристроїв до одного принтера, який буде друкувати чеки. Такий принтер легко замінити на інший і набагато легше чим звичайний касовий апарат [3].

Наразі ринок програмних касових апаратів налічує декілька готових до використання реалізацій. Усі вони мають певні недоліки, основними серед них є відсутність інтуїтивного класичного меню та можливості створення звітів за фіскальними чеками. Дані фактори в подальшому використані негативно вплинуть на зручність використання застосунку, що в свою чергу з часом призводить до негативної динаміки ставлення до продукту [4].

Тому розробка власного програмного забезпечення для касового апарату є достатньо актуальним завданням.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є підвищення ефективності проведення розрахункових операцій шляхом збільшення варіативності при формуванні звіту та удосконалення алгоритмів шифрування інформації.

Відповідно до поставленої мети в бакалаврській дипломній роботі потрібно вирішити такі **завдання**:

– проаналізувати сучасний стан систем реєстрації розрахункових операцій;

– проаналізувати методи шифрування даних

– розробити ефективний метод шифрування даних;

– розробити метод генерування звіту на основі розрахункових операцій;

– розробити графічний інтерфейс користувача;

– розробити програмні компоненти касового апарату;

– провести тестування програмного продукту.

Об'єкт дослідження – процеси збереження, сортування та аналізу інформації про розрахункові операції, генерування звіту на основі отриманих даних.

Предмет дослідження – методи та засоби для організації процесів збереження, сортування та аналізу інформації про розрахункові операції, генерування звіту на основі отриманих даних.

Методи дослідження. У процесі дослідження використовувались: теорії чисел і математичної статистики, теорії алгоритмів для генерування звіту; комп'ютерне шифрування для захисту від несанкціонованого доступу.

Новизна отриманих результатів.

1.Удосканалено метод генерування звіту про надані розрахункові послуги за певний період часу, які були отримані поточним програмним реєстратором

розрахункових операцій, за рахунок регулювання часових інтервалів що дозволяє покращити наочність отриманого фінансового звіту.

2.Удосконалено метод шифрування інформації, який відрізняється від відомих використанням вирівнювання та розбиття на блоки з перестановками та додаванням фальшивих блоків, що дозволило підвищити надійність шифрування.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих у бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для підвищення ефективності розрахункових операцій та фіскальної звітності.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У наукових працях, опублікованих у співавторстві, автору належать такі результати: ПРРО як заміна касовому апарату [5]; DLSS 2.0: масштабування зображень в режимі реального часу [6].

Апробація результатів роботи. Результати роботи доповідалися на LI науково-технічній конференції підрозділів Вінницького національного технічного університету (2022 р., м.Вінниця) та X Міжнародна науково-практична конференція “PRIORITY DIRECTIONS OF SCIENCE AND TECHNOLOGY DEVELOPMENT” (13-15 червня 2021 р., м.Київ).

Публікації. Основні результати досліджень опубліковано у 2 наукових працях у збірниках матеріалів конференцій.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем реєстрації розрахункових операцій

Зважаючи на досить високий темп розвитку технологій в сфері фінансів в сучасному світі, такому питанню, як опрацювання фінансових процедур та захист конфіденційної інформації він несанкціонованого доступу, критично важливим є зосередженням достатньої кількості уваги з боку розробників програмних додатків для забезпечення надійності створюваного застосунку. Перехід від касових апаратів на їх програмний аналог є лише питанням часу так само, як безготівкова оплата витісняє готівку. З боку розробника важливим аспектом у програмній реалізацій також має бути практичний та юзабельний інтерфейс застосунку для зниження порогу входження користувачів [5].

Системи реєстрації розрахункових операцій дають змогу значною мірою поліпшити процес надання фінансових послуг, адже це дає змогу завірити факт купівлі-продажі певного товару чи послуги, таким чином покупець захищає себе від будь-яких махінацій зі сторони продавця, а продавець виходить з тіньового ведення бізнесу, якщо він не використовував касові пристрої до цього часу. Проте більшість людей ставиться вороже до такого нововведення про це свідчать масові мітинги протесту [7], адже вбачають в ньому додаткове оподаткування для малого бізнесу і намагання занизити оподаткування великого, що маскуватиметься під групу малого бізнесу. Щоб досягти певного консенсусу законодавча влада має спростити процедури ведення звітності по фінансовій діяльності і запровадити певні пільгові тарифи чи знизити оподаткування для задоволення потреб малого бізнесу [7].

Програмний реєстратор розрахункових операцій виконує такі ж самі завдання, як і звичайний касовий апарат, але має значно більший функціонал. Окрім можливостей класичного РРО, в ньому є інструменти для оптимізації бізнес-процесів, пов'язаних з роботою програмних кас.

До програмного РРО отримують доступ через браузер, або встановлюють додаток на будь-який пристрій, звіт або чек видається зручним для клієнта способом: через соціальні мережі, електронну пошту, повідомлення чи сканування QR-коду [6].

Для порівння апаратного та програмного РРО складено таблицю 1.1.

Таблиця 1.1 – Порівняння апаратного та програмного касового апарату.

Операція	Апаратний	Програмний
Реєстрація в державному реєстрі (необхідність)	Так	Ні
Реєстрація системи	До 5 робочих днів	1 день
Фіскалізація	Всередині касового апарату	Транзакції з сервером ДФС
Застосовування в різних сферах	Не дозволяється. Для різних фінансових послуг різні касові апарати	Так, мультизадачний
Термін експлуатації	Максимум 9 років після виробництва	Необмежений

Також серед переваг програмного РРО:

- Відсутність необхідності обов’язкового обслуговування;
- Мобільність та зручність;
- Відсутність прив’язки до сфери його використання;
- Варіативність видачі фіскального звіту через Qrcode, e-mail чи месенджер.

Отже, програмна реалізація касового апарату має достатня вагому кількість переваг над своїм апаратним аналогом і їх популярність на ринку з часом лише зростатиме.

1.2 Порівняльний аналіз реєстраторів розрахункових операцій

Незважаючи на всі переваги програмної реалізації касового апарату, на сьогоднішній день ринок ПРРО дуже скромний і вибір у потенційних користувачів

невеликий. Серед існуючих програм найбільш близькими до створюваного застосунку по реалізації та призначенню є:

– Cashalot;

– СОТА Каса;

– ПРРО Каса.

Cashalot – це програмний реєстратор розрахункових операцій представлене розробником M.E.Doc, сучасний програмний засіб для ведення фінансової звітності, який забезпечує швидку реєстрацію розрахункових документів (чеків продажу, звітів, та інших) на фіскальному сервері державної податкової служби і має достатній для користувача набір функцій. Встановлюється на ПК, планшет, смартфон користувача, та може об’єднуватись з обліковою системою. На рис. 1.1 наведено користувацький інтерфейс програми.

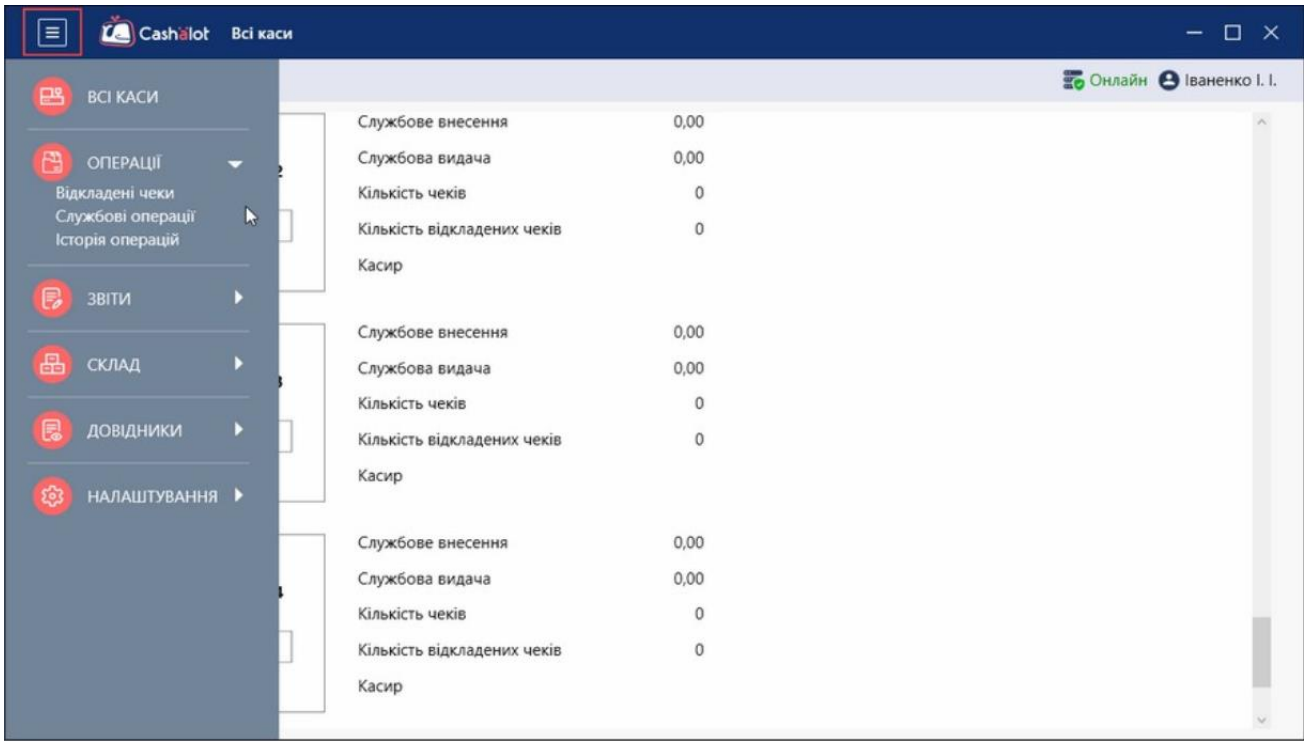


Рисунок 1.1 – Користувацький інтерфейс програми Cashalot

СОТА Каса – реєстратор розрахункових операцій і сучасна заміна звичним касовим апаратам для невеликих середніх і малих компаній. Онлайн сервіс призначений також для фіскалізації чеків в ДПС. Оскільки для доступу необхідно

мати лише браузер, отримати доступ можна на усіх сучасних пристроях. Користувацький інтерфейс додатку наведено на рисунку 1.2.

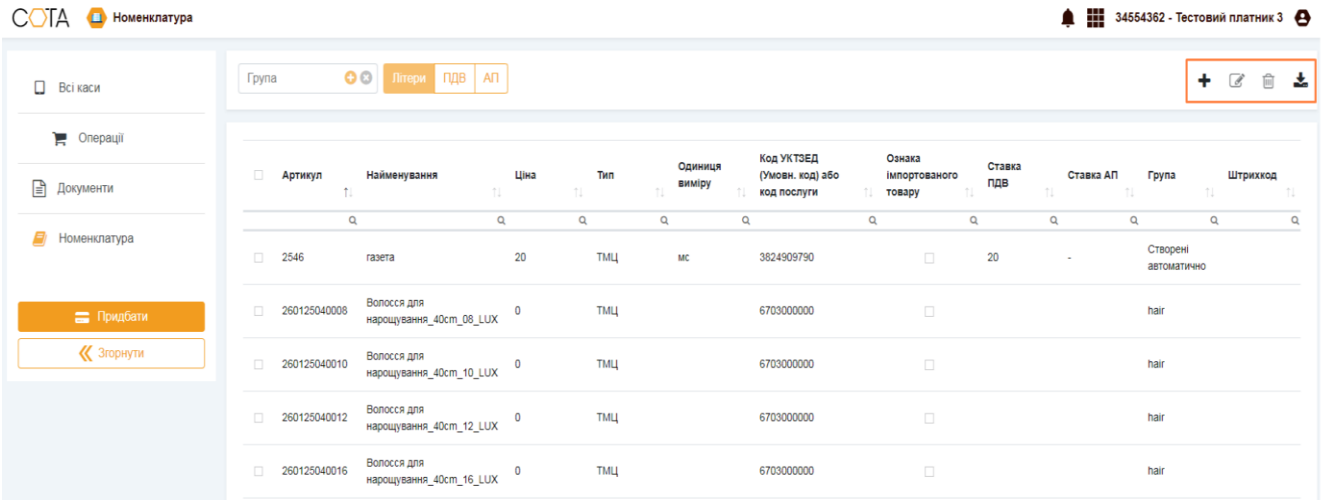


Рисунок 1.2 – Користувацький інтерфейс COTA Каса

ПРРО Каса – безкоштовне програмне забезпечення розроблене під державне замовлення, яке допомагає у введені прозорого бізнесу і без зайвого клопоту. Він значною мірою економить гроші продавця, не потребує обслуговування, забезпечує автоматичну онлайн-фіскалізацію в ДПС, спрощує надання звітності по проведеним фінансовим операціям. На рисунку 1.3 наведено користувацький інтерфейс додатку.

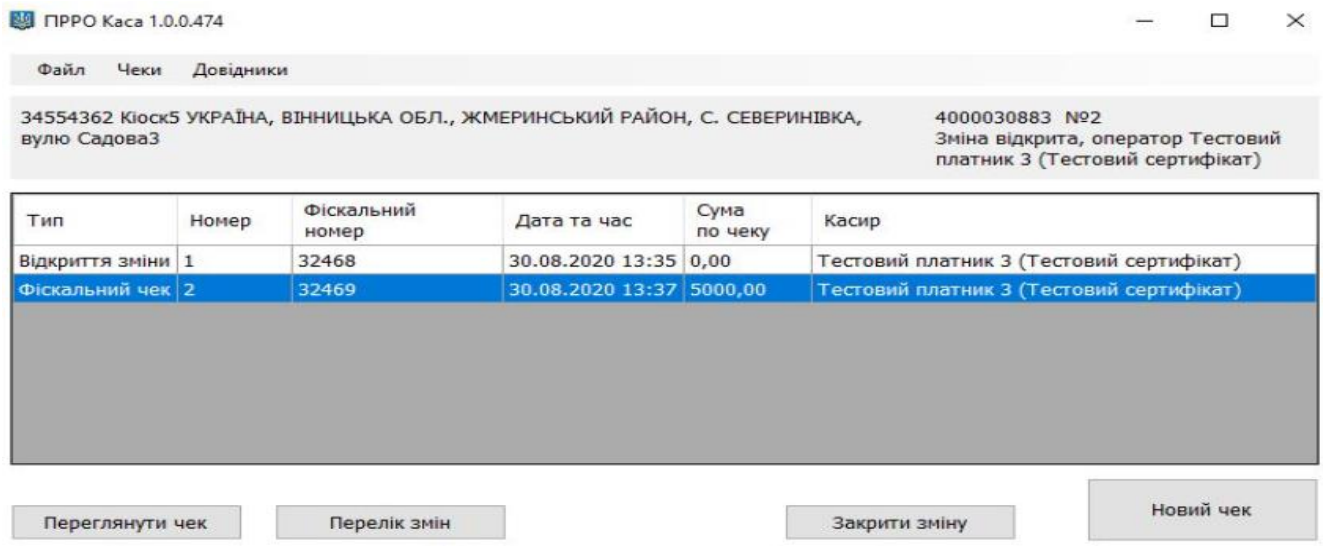


Рисунок 1.3 – Користувацький інтерфейс програми ПРРО Каса

Після аналізу усіх аналогів визначено їхні переваги та недоліки та проведено порівняння із розроблюваним веб-додатком під назвою «Cashregister».

Результат порівняння зведено в таблицю 1.2.

Таблиця 1.2 – Порівняльні характеристики програмних продуктів

Критерій	Cashalot	СОТА Каса	ПРРО Каса	Cash Register
Розмежування прав користувачів	0	0	0	1
Безкоштовність функціонала	0	0	0.5	1
Шифрування даних	1	1	1	1
Створення X та Z звітів	1	1	1	1
Кросплатформність	1	1	0	1
Підсумковий результат	3	3	2.5	5

В результаті порівняння існуючих аналогів було зроблено висновок, що розробка власного програмного додатку для реєстрації розрахункових операцій буде доцільною. В результаті розробки отримаємо продукт, який покриває недоліки існуючих аналогів та забезпечує порівняно вищу ефективність та більший обсяг функціонала.

1.3 Аналіз методів шифрування даних

Основна мета криптографічної захисту або криптографічного закриття інформації є захистом від витікання інформації, яка забезпечується шляхом звертання однозначного перетворення повідомлень або зберігаються даних у форму, невідому для сторонніх чи неавторизованих осіб. Трансформація, що забезпечує криптозахист, називається шифруванням. Захист від модифікації інформації та зв'язування хибних даних, тобто імітація захисту, забезпечується

генерацією імітоприставки. Останній представляє собою інформаційну послідовність, отриману за визначеними правилами з відкритих даних і ключа.

Існує два типи алгоритмів шифрування, а саме симетричні та асиметричні. Симетричні алгоритми шифрування використовують один і той же ключ для шифрування інформації та для її розшифровування, а асиметричні алгоритми використовують два ключа – один для шифрування, інший для розшифровування. Якщо зашифровану інформацію необхідно передавати в інше місце, то в цьому потрібно передавати і ключ для розшифрування. Слабке місце тут канал передачі даних – якщо він не захищений або його прослуховують, то ключ для розшифрування може потрапити до зловмисника. Системи на асиметричних алгоритмах позбавлені цього недоліку. Оскільки кожен учасник такої системи має пару ключів: відкритий і закритий ключ. Відкритий ключ – представлений публічно, доступ до нього повинен бути у всіх, хто буде мати потребу зашифрувати інформацію. В той же час закритий ключ – це приватний ключ, який повинен бути доступним лише тим хто має право розшифрувати інформацію, за своїм розміром він значно більший від секретного ключа симетричного шифрування.

Програмна реалізація алгоритмів шифрування є досить громіздкою й час виконання своєю чергою теж. Наглядною демонстрацією буде йтися про алгоритми асиметричного шифрування такі як RSA, ECC, де потрібні операції з числами довжиною 256 біт і більше. Для достатнього пришвидшення операцій з такими числами застосовують апаратне втілення базових операцій з великими числами, на основі яких будують складні крипто-алгоритми [8].

У випадку алгоритму RSA, в його основі лежить підняття числа до степеню та обчислення модуля цього числа. Значення показника степеню може сягати 4096 біт. Таку операцію можливо розкласти на базові операції додавання та множення двох великих чисел. Тому в апаратному втіленні втілюють власне ці базові операції, які виконуються за лічені такти.

Високорівневий алгоритм імплементують виключно в програмному вигляді. Високорівневий алгоритм лише вказує код операції, яку потрібно здійснити (наприклад додавання, віднімання, виключне АБО тощо), вказує вказівники на

пам'ять, звідки взяти вхідні дані, та куди віднести результат. Як зазвичай пам'ять, де зберігаються операнди таких криптографічних операцій, це окремо виділена ділянка пам'яті, до якої апаратний блок шифрування має пришвидшений доступ, і для адресації використовується лише деяка частина молодших біт адреси.

Існує достатньо велика кількість алгоритмів шифрування, які мають свої переваги та недоліки. Для розроблюваного застосунку важливе значення матиме можливість асинхронного виконання запитів до його модулів для забезпечення коректної безперебійної роботи для великої кількості користувачів одночасно. Для цієї мети найкращим рішенням є використання RSA алгоритму шифрування, який складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів [9] з подальшим використання звичайного симетричного способу для режиму шифрування по блокам.

1.4 Постановка задачі розробки реєстратора розрахункових операцій

Після проведення аналізу питання розробки касового апарата для реєстрації розрахункових операцій, було визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

– проаналізувати сучасний стан систем реєстрації розрахункових операцій;

– проаналізувати методи шифрування даних

– розробити метод генерування звіту на основі розрахункових операцій;

– розробити ефективний метод шифрування даних;

– розробити графічний інтерфейс користувача;

– розробити програмні компоненти касового апарату;

– провести тестування програмного продукту.

Також було сформовано функціональні та нефункціональні вимоги, до функціональних віднесено:

– Перемикання мови інтерфейсу;

– Ведення логу помилок;

– Створення точки відновлення;

– Пошук за кодом та назвою товару;

– Шифрування даних;

– Звіт по касирах;

– Звіт по товарах;

– Звіт по часовим інтервалам;

– Розмежування прав користувачів.

До нефункціональних:

– Відновлюваність;

– Зручність у використані;

– Надійність;

– Легкість в освоєні;

– Безпека;

– Швидкодія при виконані різних запитів.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто стан питання існуючих програмних додатків для реєстрації розрахункових операцій. Також було проведено аналіз існуючих аналогів та проведено їх порівняння між собою та розроблюваним ПРРО. У результаті доведено доцільність розробки бакалаврської дипломної роботи, а також проведено аналіз існуючих підходів до вирішення поставленої задачі. Було встановлено основні завдання, які необхідно виконати для розробки веб-додатку касового апарату.

2 РОЗРОБКА МЕТОДІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Розробка методу формування звіту за проведеними розрахунковими операціями

Незамінною складовою касового апарату є введення звітності за успішно здійсненими розрахунковими операціями.

Для формування звіту по виконаним розрахунковими операціями з можливістю регулювати часовий інтервал необхідно реалізувати певний ряд вимог до веб-додатку, а саме:

1. Захист від несанкціонованого доступу тобто авторизація зареєстрованого працівника.

2. Розмежування прав доступу – не кожний працівник повинен мати доступ до формування звітів.

Початковим етапом формування звіту буде проведення нульового звіту, також його називають ранковим звітом. Його виконання є обов’язковим для впевненості у відсутності готівки в касі (перевірка, що виручки в касі на початок робочої зміни немає), але оскільки веб-додаток зберігає інформацію в електронному вигляді вимоги до нього дещо послаблені.

Далі відбувається перехід до формування загального звіту його особливістю можна зазначити великий спектр інструментів для фінансових перевірок , на цьому етапі у працівника у відповідності до деяких функціональних вимог є можливість сформувати звіт на основі кількох категорій, а саме:

– Відповідно до встановленого часового інтервалу;
– Розділяючи по касирам;
– Розділяючи по товарам.

Для успішної реалізації методу було сформовано блок-схему алгоритму його функціонування, яка описує програмну реалізацію ведення звітності за здійсненими розрахунковими операціями, тобто процес формування звітів. На рисунку 2.1 зображено блок-схему даного алгоритму.

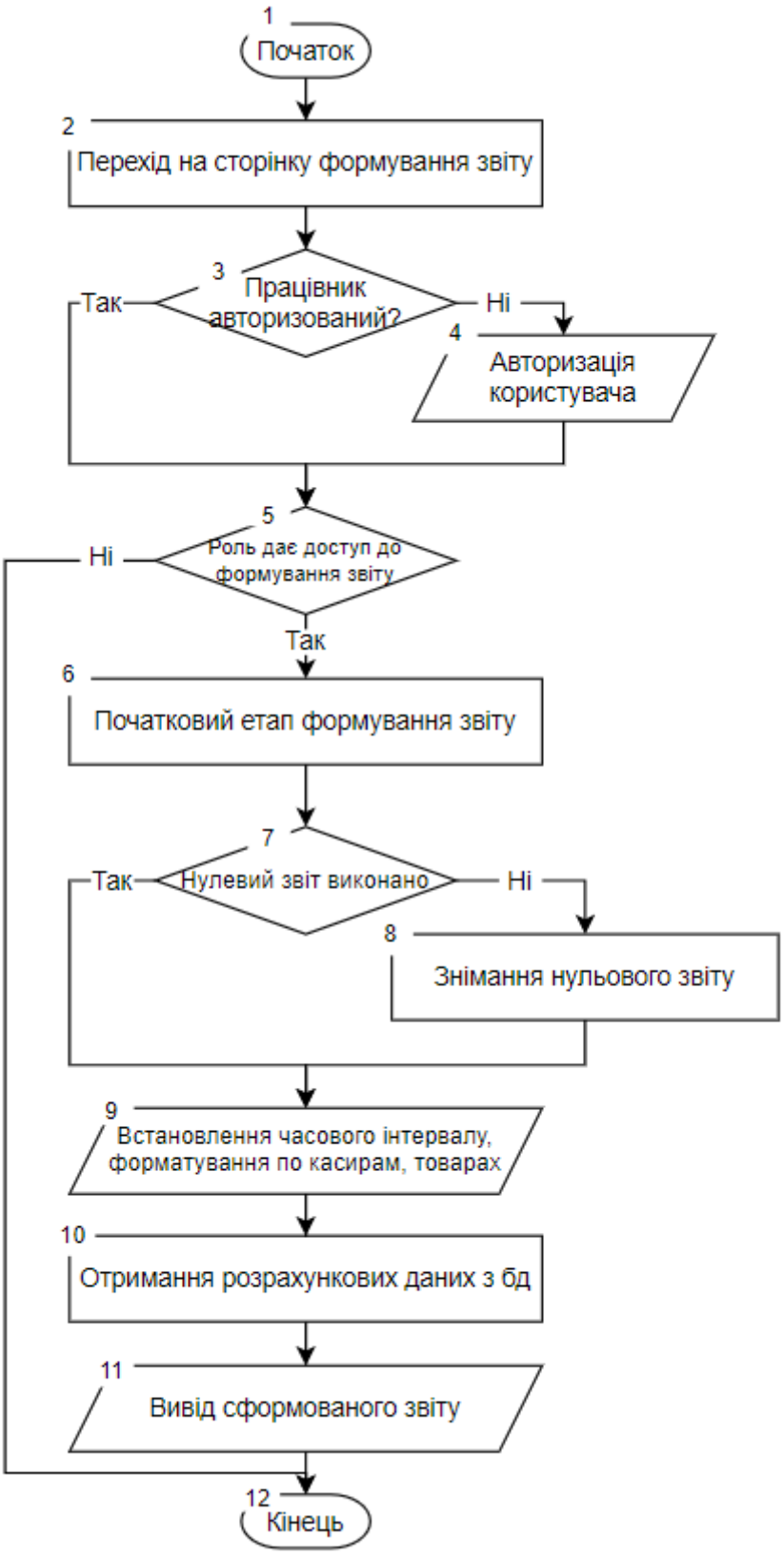


Рисунок 2.1 – Блок-схема алгоритму формування звіту

Сформований звіт можна роздрукувати за допомогою під’єднаного принтера, або відправити на електрону пошту чи за допомогою QR-коду.

2.2 Розробка методів шифрування та дешифрування даних на основі існуючих рішень

На сьогодні шифрування на основі лише одного алгоритму не практикується це пов’язано з значною легкість злому такого типу захисту, якщо взяти алгоритм RSA на перший погляд він здається надійним, адже знаходження закритого ключа по відкритому представляє надзвичайно складний процес і обрахункові можливості звичайного пристрою тут не допоможуть. Проте злам відбувається через іншу вразливості – кодування пробілів та однієї чи двох букв зловмисник легко розшифрує закодовані дані навіть не знаючи секретних ключів шифрування.

Для усунення недоліків RSA шифрування прийнято використовувати додаткові методи для посилення шифрування. В поєднанні з режимом блокового шифрування його ефективність значно зростає, але загроза злому все ще велика оскільки при отримані ключа шифрування по блокам його розшифрування не складатиме великої складності. Використання додаткових фальшивих блоків певною мірою посилить шифрування, адже лише з ключем від блокового шифрування вони лише заплутають зловмисника.

Вирівнювання даних збільшує довжину даних для отримання кратної довжини розміру даних по розбитим блокам.

Шифрування даних складається з трьох етапів:

– Вирівнювання даних.
– Розбиття на блоки, додавання фальшивих блоків та їх шифрування;
– Шифрування RSA.

На першому етапі виконується вирівнювання даних, його необхідність зумовлена подальшим розбиттям на блоки і у випадку непарної кількості байтів даних розбиття на рівні блоки буде неможливим.

Вирівнювання складається з наступних кроків:

Крок 1. Зчитуються представлені дані.

Крок 2. Перевірка на парність.

Крок 3. Якщо розмірність даних парна дані не вирівнюються.

Крок 4. Якщо розмірність даних непарна відбувається їх доповнення.

Далі виконується розбиття на блоки та додавання фальшивих блоків:

Крок 5. Вводимо ключі для фальшивих блоків та ключ для шифрування по блокам.

Крок 6. Заповнені фальшиві та справжні блоки зашифровуються за допомогою ключа та бітової функції XOR.

Далі для шифрування інформації було взято асиметричний алгоритм RSA.

Алгоритм RSA складається з наступних кроків:

Крок 7. Отримуємо текст після шифрування XOR.

Крок 8. Обираються числа p і q , для надійності значення береться велике число.

Крок 10. Обраховується добуток обраних простих чисел $n = (p \times q)$;

Крок 11. Обираємо велике число d яке мусить бути взаємно простим з виразом для успішного шифрування та дешифрування $(p - 1) \times (q - 1)$.

Крок 12. Довільне число e (e<n), обране таким чином, щоб найбільший спільний дільник $(e, (p - 1)(q - 1)) = 1$, тобто відношення e та $(p - 1) \times (q - 1)$. повинно бути простим.

Крок 13. Методом Евкліда в цілих числах знаходиться рішення рівняння $e \times d + (p - 1)(q - 1) \times y = 1$. Тут невідомими є змінні d і y - метод Евкліда саме і знаходить безліч пар (d, y), кожна з яких є рішенням рівняння в цілих числах.

Крок 14. Пара (e ; n) являє собою публічний ключ шифрування нашого алгоритму.

Крок 15. Пара (d ; n) являє собою закритий ключ шифрування нашого алгоритму.

Крок 16. Введений текст розбивається на рівні блоки, де $k=[\log_2(n)]$ біт, це довжина розбитих блоків у бітах.

Крок 17. Подібний блок може бути трактованим як число з діапазону $(0;2^k-1)$. Для кожного такого числа m_i обчислюється $c_i = ((m_i)^e) mod\, n$. Блок c_i і є зашифроване повідомлення Їх можна передавати по відкритому каналу, оскільки операція зведення в ступінь по модулі простого числа, є необоротним

математичним завданням.

Для дешифрування допоможе збережене в секреті число d. Досить давно була доведена теорема Ейлера, окремий випадок якої затверджує, що якщо число n представимо у вигляді двох простих чисел p і q, то для будь-якого x має місце рівність ($x^{(p-1)(q-1)})mod\ n\ =\ 1$. Для дешифрування порядок буде оберненим до шифрування спочатку дешифруємо RSA:

Крок 1. Отримуємо закодований текст який необхідно дешифрувати в шифр по блокам для подальшого дешифрування за допомогою іншого ключа.

Крок 2. Отримаємо пару ключів (d, n) для дешифрування, d є закритим ключем і його конфіденційність повинна бути максимальною.

Крок 3. За допомогою алгоритму Евкліда d таке, що $e \times d + (p-1)(q-1) \times y = 1$, тобто $e \times d = (-y)(p-1)(q-1) + 1$ та зведеної в ступінь $(-y)$ теореми Ейлера отримуємо $(x^{e\times d})mod\ n\ =\ x$.

Крок 4. Щоб прочитати повідомлення $c_i = ((m_i)^e)mod\ n$ досить звести його в ступінь d по модулі m: $((c_i)^d)mod\ n\ =\ ((m_i)^{e\times d})mod\ n\ = m_i$.

Крок 5. Отримуємо дешифрований RSA текст з m_i, дальше його необхідно дешифрувати за допомогою блокового дешифрування.

Далі дешифруємо по блокам:

Крок 6. Розбиваємо на блоки та дешифруємо їх за допомогою ключа.

Крок 7. Прибираємо фальшиві блоки за ключами фальшивих блоків.

І останній етап вирівнювання даних:

Крок 8. Прибираємо вирівнювання даних.

Крок 9. Отримуємо звичайний текст.

Якщо зашифрованим текстом був пароль працівника то він буде використовуватись про авторизації, в момент порівняння введено поля для пароль користувача за допомогою форми авторизації та розшифрованого пароля з бази даних.

На рисунку 2.2 та 2.3 зображено блок-схеми алгоритму шифрування та дешифрування даних відповідно.

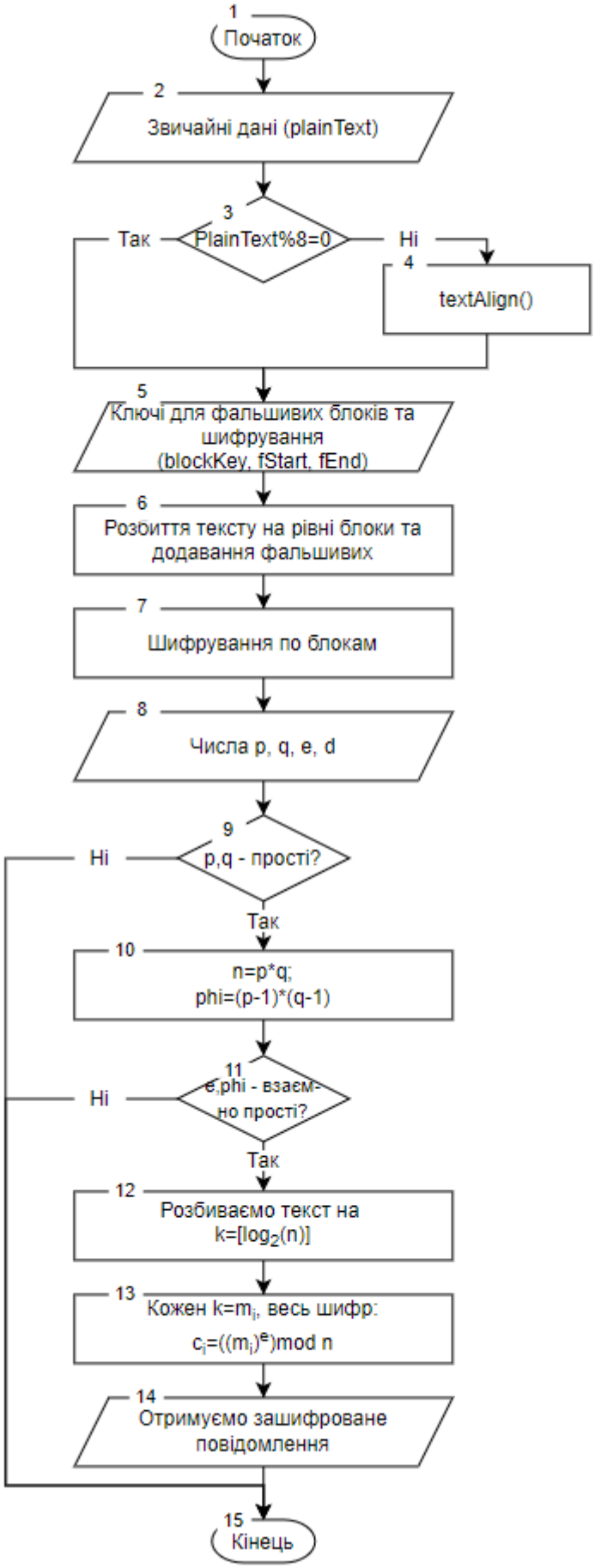


Рисунок 2.2 – Блок-схема алгоритму шифрування даних

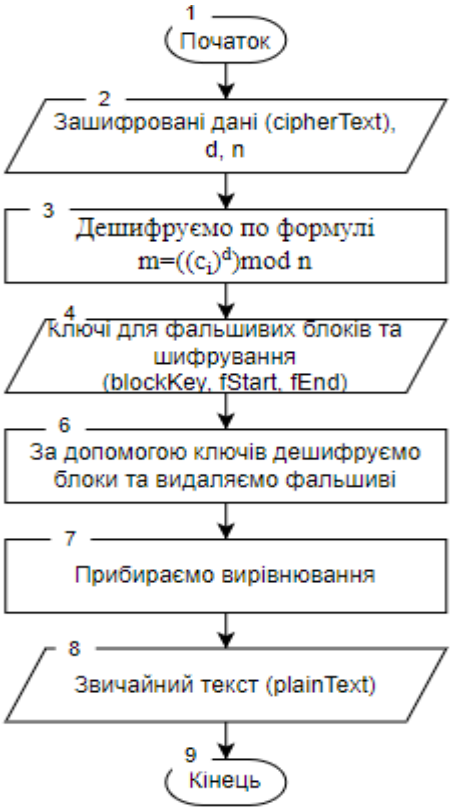


Рисунок 2.3 – Блок-схема алгоритму дешифрування даних

Операції зведення в ступінь більших чисел досить трудомісткі для сучасних процесорів, навіть якщо вони виробляються по оптимізованим за часом алгоритмам. Тому звичайно весь текст повідомлення кодується звичайним блоковим шифром (набагато більше швидким), а от сам ключ блокового шифрування шифрується асиметричним алгоритмом за допомогою відкритого ключа одержувача.

2.3 Висновки

У другому розділі було розглянуто можливі варіанти взаємодії працівника із касовим апаратом та обрано найбільш ефективні методи із них для реалізації у застосунку. На основі базового макету для веб-додатків було побудовано прототип власного графічного інтерфейсу веб-додатку, який буде зручним у використанні та зрозумілим для користувача. Було виведено метод формування розрахункового звіту, розроблено метод та алгоритм шифрування та дешифрування і побудовано їх блок-схему.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДЛЯ ПРРО

3.1 Варіантний аналіз і обґрунтування вибору мови програмування

На початковому етапі розробки будь-якого програмного забезпечення, тобто на етапі проєктування дуже важливим є вибір мови програмування, технологій та засобів розробки. Для того, щоб обрати найбільш задовольняючу мову висувається ряд вимог та робиться порівнянь між ними на основі даних порівнянь робиться певне підбиття підсумків чи дозволяє мова програмування реалізувати усі поставлені вимоги та функції у розроблюваному додатку. Під час реалізації касового апарату для реєстрації розрахункових операцій пріоритетним вибором стане та мова програмування, яка дасть змогу надійно зберігати конфіденційні дані мати повний доступ до функціонала бази даних додатку, матиме додаткові бібліотеки, потрібні для побудови графічного інтерфейсу, звіті та чеку. Розглянемо найбільш популярні мови програмування, які використовуються для розробки веб-додатку, а саме C#, Java та Python [10].

C# є однією з найпопулярніших мов програмування. Його використовують мільйони розробників по всьому світу. Оскільки C# розроблено компанією Microsoft як частина їхньої сучасної платформи для розробки та виконання додатків, .NET Framework, ця мова широко поширена серед компаній, організацій та окремих розробників, орієнтованих на Microsoft. Мова C# і платформа .NET повністю обслуговуються та керуються Microsoft і не відкриті для третіх сторін. Через це всі інші великі корпорації програмного забезпечення, такі як IBM, Oracle і SAP, базують свої рішення на платформі Java і використовують Java як свою основну мову для розробки власних програмних продуктів.

На відміну від C# та .NET Framework, мова та платформа Java є проєктами з відкритим вихідним кодом, у яких бере участь ціла спільнота програмних компаній, організацій та окремих розробників. Стандарти, специфікації та всі нові функції у світі Java є розроблено робочими групами, сформованими з усієї спільноти Java, а не окремої компанії (як у випадку C# і .NET Framework).

C# – сучасна об'єктноорієнтована мова програмування загального призначення, створена і розроблена компанією Microsoft разом із платформою .NET. Існує дуже різноманітне програмне забезпечення, розроблене на C# та на платформі .NET: офісні програми, веб-додатки, вебсайти, настільні програми, мобільні програми, ігри та багато інших.

C# – це мова високого рівня, подібна до Java і C++ і, до певної міри, до таких мов, як Delphi, VB.NET і C. Усі програми C# об'єктноорієнтовані. Вони складаються з набору визначень у класах, які містять методи, а методи містять логіку програми – інструкції, які виконує комп’ютер [11].

Мова C# не поширюється як окремий продукт – вона є частиною платформи Microsoft .NET Framework (вимовляється як «Microsoft dot net framework»). .NET Framework, як правило, складається із середовища для розробки та виконання програм, написаних на C# або іншій мові, сумісні з .NET (наприклад, VB.NET, Managed C++, J# або F#). До його складу входять:

- Мови програмування .NET (C#, VB.NET та інші); - середовище для виконання керованого коду (CLR), яке керовано виконує програми на C#;
- Набір інструментів розробки, наприклад компілятор csc, який перетворює програми C# у проміжний код (так званий MSIL), який може зрозуміти CLR;
- Набір стандартних бібліотек, таких як ADO.NET, які надають доступ до баз даних (наприклад, MS SQL Server або MySQL) і WCF, який з'єднує програми за допомогою стандартних комунікаційних фреймворків і протоколів, таких як HTTP, REST, JSON, SOAP і TCP-сокети.

Мова програмування Java спочатку була розроблена Sun Microsystems, ініційована Джеймсом Гослінгом і випущена в 1995 році як основний компонент платформи Java Sun Microsystems (Java 1.0 [J2SE]). Останнім випуском стандартної версії Java є Java SE 8. З розвитком Java та її широкою популярністю було створено безліч конфігурацій, щоб відповідати різним типам платформ. Наприклад: J2EE для корпоративних додатків, J2ME для мобільних додатків. Нові версії J2 були перейменовані на Java SE, Java EE та Java ME відповідно. Java гарантовано буде

записувати один раз, запускати будь-де. Java можна охарактеризувати такими пунктами як:

- Об’єктноорієнтованою: у Java все є об’єктом. Java можна легко розширити, оскільки вона заснована на об'єктній моделі.
- Незалежною від платформи: на відміну від багатьох інших мов програмування, включаючи C і C++, під час компіляції Java не компілюється на платформу, а в незалежний від платформи байт-код. Цей байт-код розповсюджується в Інтернеті та інтерпретується віртуальною машиною (JVM) на будь-якій платформі, на якій він виконується.
- Простою: Java розроблена так, щоб її було легко вивчати. Якщо ви розумієте основну концепцію ООП Java, її буде легко освоїти.
- Безпечною: за допомогою безпечної функції Java вона дозволяє розробляти системи без вірусів і несанкціонованого доступу. Методи аутентифікації засновані на шифруванні з публічним ключем.
- Архітектурно-нейтральною: компілятор Java генерує архітектурно-нейтральний формат об’єктних даних, який робить скомпільований код виконуваним на багатьох процесорах, з наявністю системи середовища виконання Java.
- Портативною: будучи нейтральним щодо архітектури та не має аспектів специфікації, що залежать від реалізації, робить Java портативною. Компілятор на Java написаний на ANSI C з чіткою межею переносимості, яка є підмножиною POSIX.
- Надійною: Java докладає зусиль для усунення ситуацій, схильних до помилок, наголошуючи головним чином на перевірці помилок під час компіляції та перевірці під час виконання.
- Багатопоточною: за допомогою багатопотокової функції Java можна писати програми, які можуть виконувати багато завдань одночасно. Ця особливість дизайну дозволяє розробникам створювати інтерактивні програми, які можуть працювати безперебійно.

• Інтерпретованою: байт-код Java на льоту транслюється у власні машинні інструкції і ніде не зберігається. Процес розробки є більш швидшим і аналітичним, оскільки зв’язування є поетапним і легким процесом.

• Високо продуктивною: за допомогою компіляторів Just-In-Time Java забезпечує високу продуктивність. Розповсюджений: Java розроблена для розподіленого середовища Інтернету.

• Динамічною: Java навіть вважається динамічнішою, ніж C або C++, оскільки вона розроблена для адаптації до середовища, що вдосконалюється. Програми Java можуть містити велику кількість інформації під час виконання, яку можна використовувати для перевірки та дозволу доступу до об’єктів під час виконання [12].

За спроєктованою основою, Python навмисно реалізує простий і читабельний синтаксис і дуже узгоджену модель програмування. Як свідчить гасло на конференції Python, кінцевий результат полягає в тому, що Python, здається, «відповідає вашому мозку» – тобто особливості мови взаємодіють послідовним і обмеженим чином і випливають природним чином з невеликого набору ядра. Це полегшує вивчення, розуміння та запам’ятовування мови. На практиці програмістам Python не потрібно постійно звертатися до посібників під час читання чи написання коду; це послідовно розроблена система, яка, на думку багатьох, дає напрочуд звичайний код.

За філософією Python використовує дещо мінімалістичний підхід. Це означає, що, хоча зазвичай існує кілька способів виконання завдання кодування, зазвичай є лише один очевидний спосіб, кілька менш очевидних альтернатив і невеликий набір когерентних взаємодій всюди в мові. Більше того, Python не приймає за вас довільних рішень; коли взаємодія неоднозначна, явне втручання є кращим перед «магією». У розумінні Python явне краще, ніж неявне, а просте краще, ніж складне. Python вважається однією з найспрощеніших мов програмування, а отже процес його розуміння та вивчення також значно спрощується.

Крім таких тем дизайну, Python включає такі інструменти, як модулі та ООП, які природно сприяють повторному використанню коду. І оскільки Python зосереджений на якості, то й програмісти Python, природно, теж.

Під час великого буму Інтернету середини-кінця 1990-х років було важко знайти достатньо програмістів для реалізації програмних проєктів; розробників попросили впроваджувати системи так само швидко, як розвивався Інтернет. Зараз програмістів часто просять виконувати ті самі завдання з ще меншою кількістю людей.

В обох цих сценаріях Python дозволяє програмістам виконувати більше з меншими зусиллями. Він навмисно оптимізований для швидкості розробки – його простий синтаксис, динамічний введення, відсутність кроків компіляції та вбудований набір інструментів дозволяють програмістам розробляти програми за частку часу, необхідного при використанні деяких інших інструментів. Сутній ефект полягає в тому, що Python зазвичай багаторазово підвищує продуктивність розробників за рівні, які підтримуються традиційними мовами.

Результати порівняння розглянутих мов програмування за обраними критеріями наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій	Python	C#	Java
Об’єктноорієнтованість	1	1	1
Простота синтаксису	1	0,5	1
Фреймворки з супровідними матеріалами	0	0	1
Додаткові можливості для аналізу програмного коду	0	1	1
Пул з’єднань із базою даних (Connection pool) та їх надійність	0,5	0.5	1
Надійність передачі даних у мережі пристроїв	0.5	0.5	1
Підсумковий результат	3	3,5	6

На основі таблиці 3.1, можна дійти до висновку, що мова програмування Java серед усіх розглянутих найкраще підходить для розробки веб-додатку, так як задовольняє усі поставлені вимоги та потреби, які можуть виникнути в процесі розробки касового апарату. Вона має велику кількість фреймворків для роботи із методами та графічним інтерфейсом додатку і також стандартну бібліотеку для роботи із базами даних, до того ж є простою у використанні та має зрозумілий синтаксис.

3.2 Вибір середовища розробки та СУБД

Не менш важливим, ніж питання вибору мови програмування, на якій буде реалізовано функціонал розроблюваного веб-додатку касового апарату, є питання вибору інтегрованого середовища розробки. Інтегроване середовище розробки (IDE) – це програма, яка полегшує розробку додатків. Загалом, IDE – це робоче середовище на основі графічного інтерфейсу користувача (GUI), призначене для допомоги розробнику у створенні програмних додатків з інтегрованим середовищем у поєднанні з усіма необхідними інструментами. Найбільш поширені функції, такі як налагодження, контроль версій і перегляд структури даних, допомагають розробнику швидко виконувати дії, не перемикаючись на інші програми. Таким чином, він допомагає максимізувати продуктивність, надаючи подібні інтерфейси користувача (UI) для пов’язаних компонентів і скорочує час, необхідний для вивчення мови. IDE підтримує одну або декілька мов [13]. Розглянемо найбільш популярні інтегровані середовища розробки, які використовуються для реалізації програмних продуктів на мові програмування Java, а саме Eclipce, Netbeans та Intellij IDEA.

Eclipse – це безкоштовна платформа розробки на основі Java, відома своїми плагінами, які дозволяють розробникам розробляти та тестувати код, написаний іншими мовами програмування. Eclipse випускається на умовах публічної ліцензії.

Eclipse надає ряд допоміжних засобів для розробника програмного забезпечення, а саме:

• Налагоджувач Eclipse і записник дозволяють заглянути всередину виконання коду Java. Це дозволяє вам «бачити» об’єкти та зрозуміти, як Java працює за лаштунками

• Eclipse надає повну підтримку практичних методів розробки програмного забезпечення, таких як тестова розробка та рефакторинг. Це дозволяє вам вивчати ці практики під час вивчення Java.

• Якщо ви плануєте займатися розробкою програмного забезпечення на Java, вам потрібно буде вивчити Eclipse або іншу IDE. Тому вивчення Eclipse з самого початку заощадить ваш час і зусилля.

Eclipse Foundation є незалежною некомерційною корпорацією, що базується в Канаді, яка керує спільнотою розробників програмного забезпечення Eclipse з відкритим кодом і охоплює юрисдикцію Європейського Союзу. Eclipse підтримують понад 320 учасників, 1750 комітерів і понад 332 мільйони рядків коду. Метою фонду є створення як спільноти, так і екосистеми взаємодоповнюючих продуктів і послуг [14]. Приклад інтерфейсу наведено на рисунку 3.1.

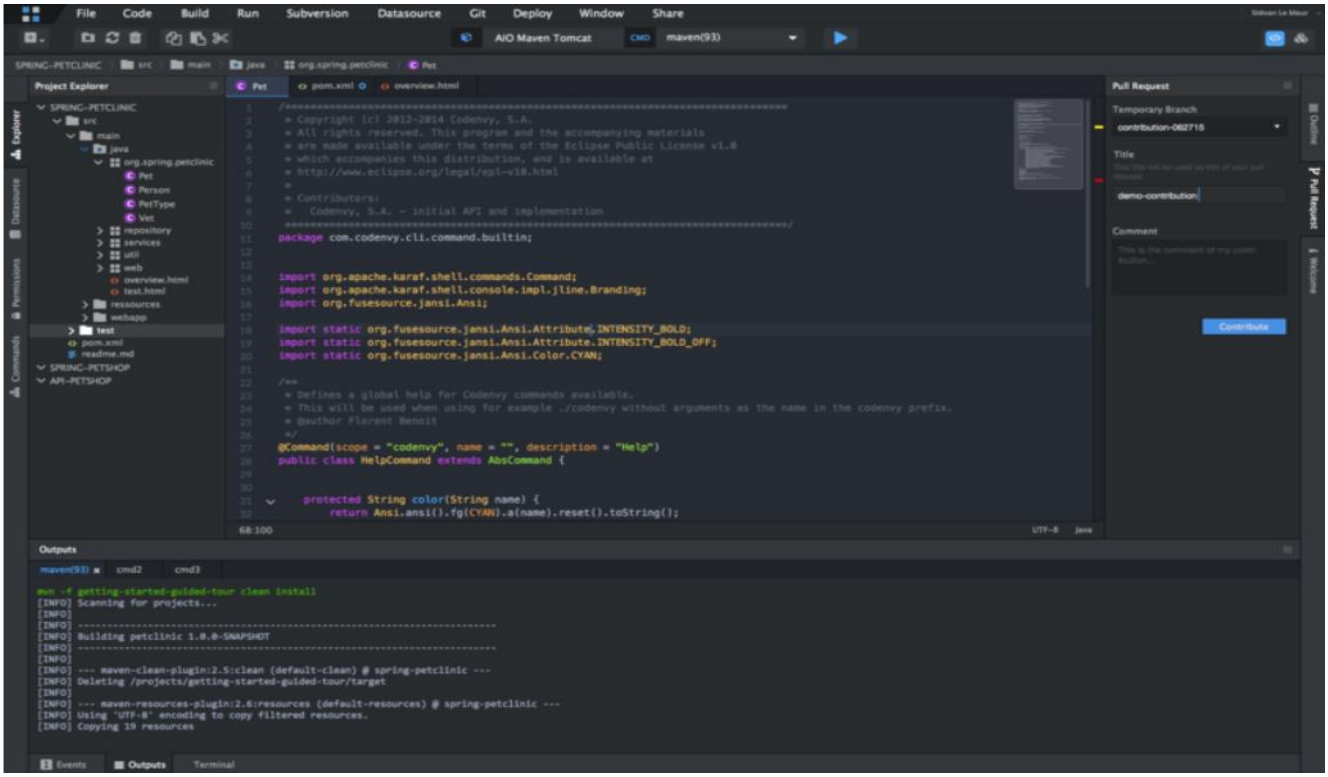


Рисунок 3.1 – Приклад інтерфейсу Eclipse

NetBeans – це безкоштовне інтегроване середовище розробки на основі Java, яке дозволяє розробникам створювати настільні, мобільні та веб-додатки. Він підтримує розробку програм різними мовами, такими як PHP, HTML, C++ і Java.

Платформа працює на основі модульної архітектури з широким набором інструментів і функцій для всього циклу розробки програмного забезпечення, від початку ідеї до розгортання програми. Через це його можна використовувати в Windows, Linux, OS X та інших операційних системах на базі UNIX. Це також офіційна IDE Java 8, яка містить потужні конвертори, редактори та аналізатори коду. Система інтегрується з широко використовуваним інструментом FindBugs, який дозволяє користувачам швидко виявляти та вирішувати типові проблеми з кодом [15]. Приклад інтерфейсу наведено на рисунку 3.2.

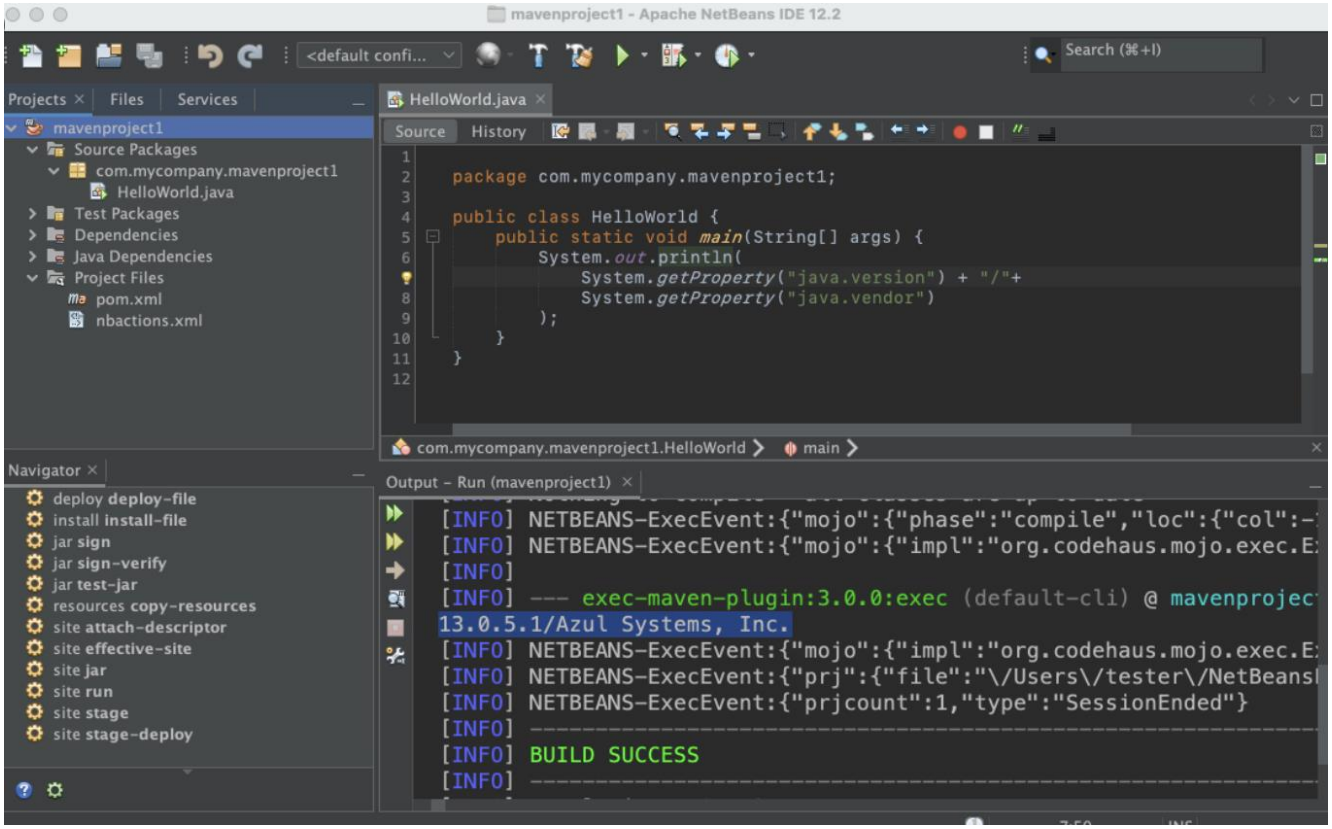


Рисунок 3.2 – Приклад інтерфейсу NetBeans

Intellij IDEA – це IDE, створена як духовний наступник широко прийнятої IDE Eclipse, що використовується для розробки Java. Eclipse, хоча і надзвичайно

потужний, часто критикують за надзвичайно незграбний і складний у використанні. Intellij IDEA намагається створити IDE з подібною потужністю до Eclipse, але з фінішним поліруванням. Розробники мали б велику перевагу, використовуючи IDEA через безліч інструментів і гачків, які вона має, щоб заощадити час на всіх проєктах. Розумне завершення коду, інтеграція вбудованого модульного тестування та власне керування Gradle – це лише деякі з основних моментів Java IDE Jetbrain [16]. Приклад інтерфейсу наведено на рисунку 3.3.

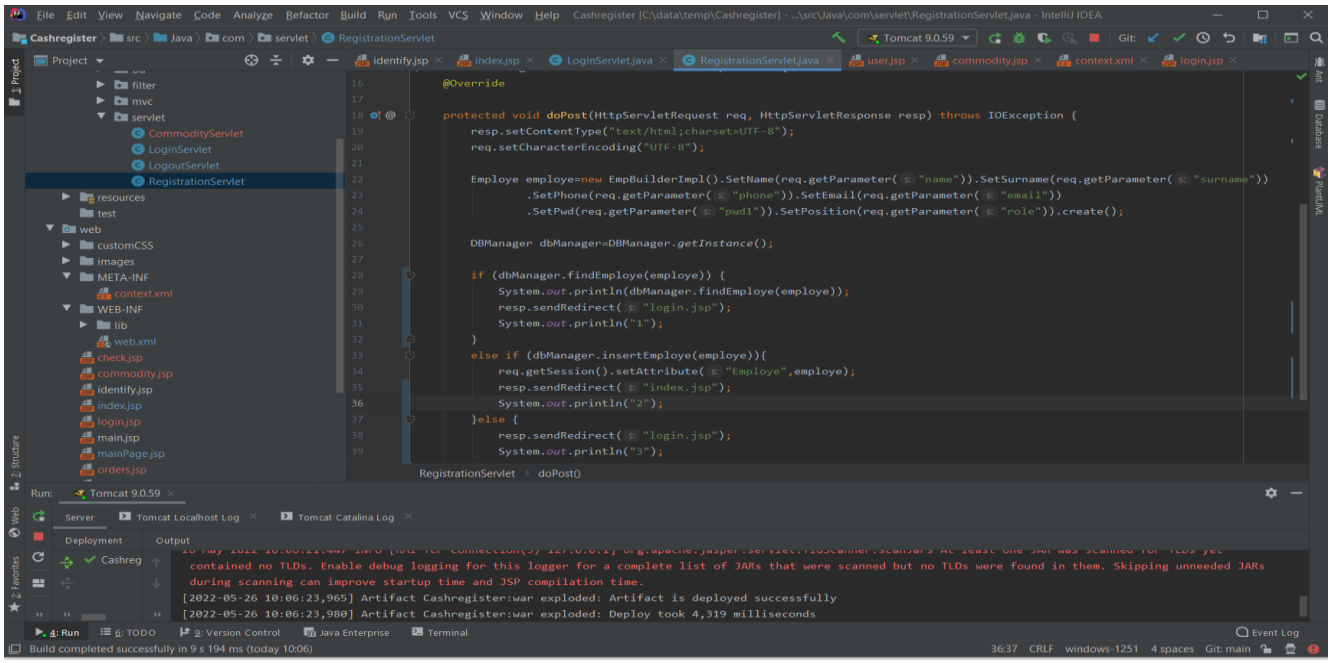


Рисунок 3.3 – Приклад інтерфейсу Intellij IDEA

Результати порівняння розглянутих інтегрованих середовищ розробки за обраними критеріями наведено в таблиці 3.2.

Таблиця 3.2 – Порівняння інтегрованих середовищ розробки

Критерій	Eclipce	NetBeans	Intellij IDEA
Безкоштовність	0,5	1	1
Розбиття коду на графічні блоки	0,5	0	1
Форматування коду	1	1	1
Швидкість компіляції	0,5	0,5	1

Продовження таблиці 3.2.

Критерій	Eclipce	NetBeans	Intellij IDEA
Спектр підтримуваних мов програмування	0	0,5	1
Магазин розширень	0	0	1
Автодоповнення коду	1	1	1
Підсумковий результат	3,5	4	7

За результатами порівняння популярних інтегрованих середовищ розробки для мови програмування Java, наведеного у таблиці 3.2, можна зробити висновок, що Intellij IDEA підходить найкраще серед усіх розглянутих аналогів для реалізації касового апарату, так як він є швидким, універсальним, зручним та має широкий спектр в магазині плагінів, які допоможуть підвищити ефективність процесу розробки вебдодатку.

Основна частина розроблюваного вебдодатку передбачає запис та зчитування інформації про витрати із бази даних. Для виконання поставлених перед програмним продуктом задач найкраще підходять реляційні бази даних. Реляційною називається база даних, у якій усі дані, доступні користувачу, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями [17]. Реляційна база даних створюється й потім управляється за допомогою спеціальних засобів – реляційних систем управління базами даних (РСУБД) [18]. Розглянемо найпопулярніші із них, а саме PostgreSQL, MySQL та SQLite.

PostgreSQL – найбільш функціональна, вільно розповсюджувана СУБД із відкритим кодом, яка максимально відповідає стандартам SQL. Основною особливістю даної системи від аналогів є те, що вона підтримує затребуваний об’єктноорінтований та реляційний підходи до баз даних [19]. Завдяки потужним технологіям PostgreSQL є дуже продуктивною СУБД. Проте її надто широкий функціонал та складність у порівнянні із MySQL та SQLite роблять її не найкращим вибором для розробки веб-додатку.

MySQL – це система управління базами даних з відкритим кодом. Це високопродуктивна и масштабована СУБД з великою кількістю програмних інтерфейсів. Вона володіє величезними функціональними можливостями та підходить для вирішення різноманітних задач [20]. Не беручи до уваги те, що у даній системі реалізовані функціональні можливості SQL не в повному обсязі, вона пропонує дуже велике різноманіття інструментів для розробки додатків, проте для реалізації функціонала неважких додатків вибір даної СУБД може бути недоцільним.

SQLite – це легка та швидка система для керування базами даних, що дає змогу легко вбудувати БД в додаток. Дана система надає представляє громіздкий набір інструментів для використання завдяки тому, що вона базується на певних файлах, тому при роботі із базами даних, побудованих на системі SQLite, відбувається звернення безпосередньо до файлів, у яких зберігається інформація, а не до портів та гнізд, як це відбувається при використанні мережевих СУБД. Саме тому SQLite є швидкою та потужною завдяки технологіям обслуговуючих бібліотек, наприклад sqlite3 для мови програмування Python [21].

Результати порівняння розглянутих систем керування базами даних за обраними критеріями наведено в таблиці 3.3.

Таблиця 3.3 – Порівняння систем управління базами даних

Критерій	PostgreSQL	SQLite	MySQL
Потужність та функціонал	1	0,5	1
Швидкість читання даних	0	0,5	1
Ступінь складності у використання	0	0,5	1
Доцільність використання для розробки неважких програмних додатків	0	0,5	1
Вміст всієї бази даних в одному файлі	0	0	1
Підсумковий результат	1	2	5

Згідно таблиці 3.3, можна зробити висновок, що система управління базами даних SQLite підходить найкраще серед усіх розглянутих СУБД, так як вона має високу швидкість обробки даних та легка у використанні, тому ідеально підійде для збереження структурованої інформації про фінансові дані швидкий доступ до даних, на основі яких будуть формуватись звіти та чек.

Отже, на основі проведеного аналізу засобів розробки було визначено, що мова програмування Java, середовище розробки Intellij IDEA та СУБД MySQL є найкращим вибором для реалізації розроблюваного веб-додатку для касового апарату під час РРО.

3.3 Розробка графічного інтерфейсу за допомогою Bootstrap framework

Графічний інтерфейс користувача (GUI) дозволяє користувачеві взаємодіяти з комп’ютерною програмою за допомогою вказівного пристрою, який маніпулює невеликими зображеннями на екрані обчислювальної техніки. Маленькі картинки називаються іконками. Можна використовувати різні типи вказівних пристроїв, наприклад, мишу, стилус або палець людини на сенсорному екрані.

Програми, які використовують графічний інтерфейс користувача називаються “програмами з графічним інтерфейсом”. Програма з графічним інтерфейсом дуже відрізняється від програми, яка використовує інтерфейс командного рядка, який отримує введення користувача від введених символів на клавіатурі. Зазвичай програми, які використовують інтерфейс командного рядка, виконують ряд завдань у заздалегідь визначеному порядку, а потім завершують роботу. Однак програма з графічним інтерфейсом створює іконки та віджети які відображаються користувачеві, а потім він просто чекає, поки користувач взаємодіє з ними. Порядок виконання завдань програмою знаходиться під контролем користувача, а не програми! Це означає, що програма з графічним інтерфейсом користувача повинна відстежувати “стан” своєї обробки та правильно реагувати на команди користувача, які надаються в будь-якому порядку, який вибирає користувач. Цей стиль програмування називається “програмування на основі

подій”. Фактично, за визначенням, усі програми з графічним інтерфейсом користувача є програмами управління яких здійснюється за допомогою івенту [22].

Bootstrap – це front-end фреймворк для створення адаптивних вебсайтів. Незалежно від того, чи це фреймворки додатків, блоги чи інші програми CMS, Bootstrap може добре підійти, оскільки він може бути таким, як вам подобається. Його комбінація HTML, CSS і JavaScript дозволяє легко створювати надійні сайти без додавання великої кількості коду. Завдяки системі сіток за замовчуванням макети легко поєднуються, а стиль кнопок, навігацій і таблиць робить базову розмітку чудовою з самого початку. Десяток плагінів JavaScript спонукають додавати інтерактивні елементи на свій сайт [22].

Спосіб взяти весь наявний вміст на сторінці та оптимізувати його для пристрою, який його переглядає називається адаптивним дизайном. Наприклад, робочий стіл не тільки отримує звичайну версію вебсайту, але також може отримати широкоекранний макет, оптимізований для більших дисплеїв, які багато людей підключили до своїх комп’ютерів. Планшети мають оптимізований макет, використовуючи переваги їх портретного або альбомного макетів. І тоді з телефонами ви можете орієнтуватися на їх набагато вужчу ширину. Щоб націлити ці різні ширини, Bootstrap використовує медіа-запити CSS для вимірювання ширини вікна перегляду браузера, а потім, використовуючи умови, змінює частини таблиць стилів, які завантажуються. Використовуючи ширину вікна перегляду браузера, Bootstrap може потім оптимізувати вміст, використовуючи комбінацію співвідношення або ширини, але в основному він покладається на властивості мінімальної та максимальної ширини. По суті, Bootstrap підтримує п’ять різних макетів, кожен з яких покладається на медіа-запити CSS. Найбільший макет має стовпці шириною 70 пікселів, що контрастує з 60 пікселями звичайного макета. Макет планшета збільшує ширину стовпців до 42 пікселів, а коли вони вужчі, кожен стовпець стає плавним, що означає, що стовпці складаються вертикально, і кожен стовпець має всю ширину пристрою.

Тому для реалізації графічного інтерфейсу було обрано фреймворк Bootstrap з його допомогою використання навігації стане найефективнішим та

найдоцільнішим у розроблюваному застосунку для реєстрації розрахункових операцій.

3.4 Розробка структури графічного інтерфейсу ПРРО

Загальна структура головної сторінки зазвичай містить стандартні типи блоків: header – задає “шапку” сторінці або розділу, в якій зазвичай знаходиться заголовок; logo – розташування логотипу застосунка ; navigation/menu – меню основної навігації по застосунку; content area – область де відображається основна інформації по певному критерію; footer – нижня область де зазвичай вказуються авторські права. На рисунку 3.4 зображено ілюстративний макет головної сторінки.

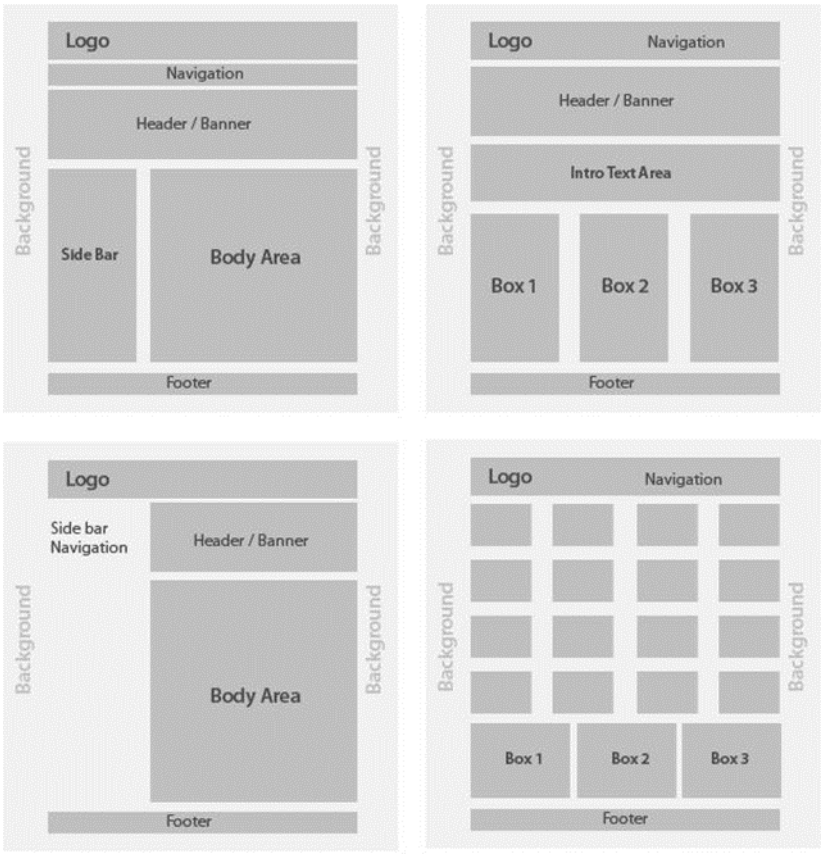


Рисунок 3.4 – Ілюстративний макет головної сторінки

У шапці касового апарату розміщено навігаційну панель. В ній присутні основні функції застосунку, а саме перехід на сторінку обслуговування замовлень, товарів чи послуг також особистий кабінет працівника і зміна мови застосунку.

Під шапкою знаходиться пошуковий рядок результат запиту якого залежить від посади яку займає працівник, для першого типу виконуються пошук по замовленням для іншого по представлених товарах чи послуг.

Далі розташовано body area для перегляду результатів пошуку.

В нижньому колонтитулі знаходяться дані про автора.

На рисунках 3.5, 3.6 та 3.7 зображені розроблені структурні схеми інтерфейсів вікон застосунка.

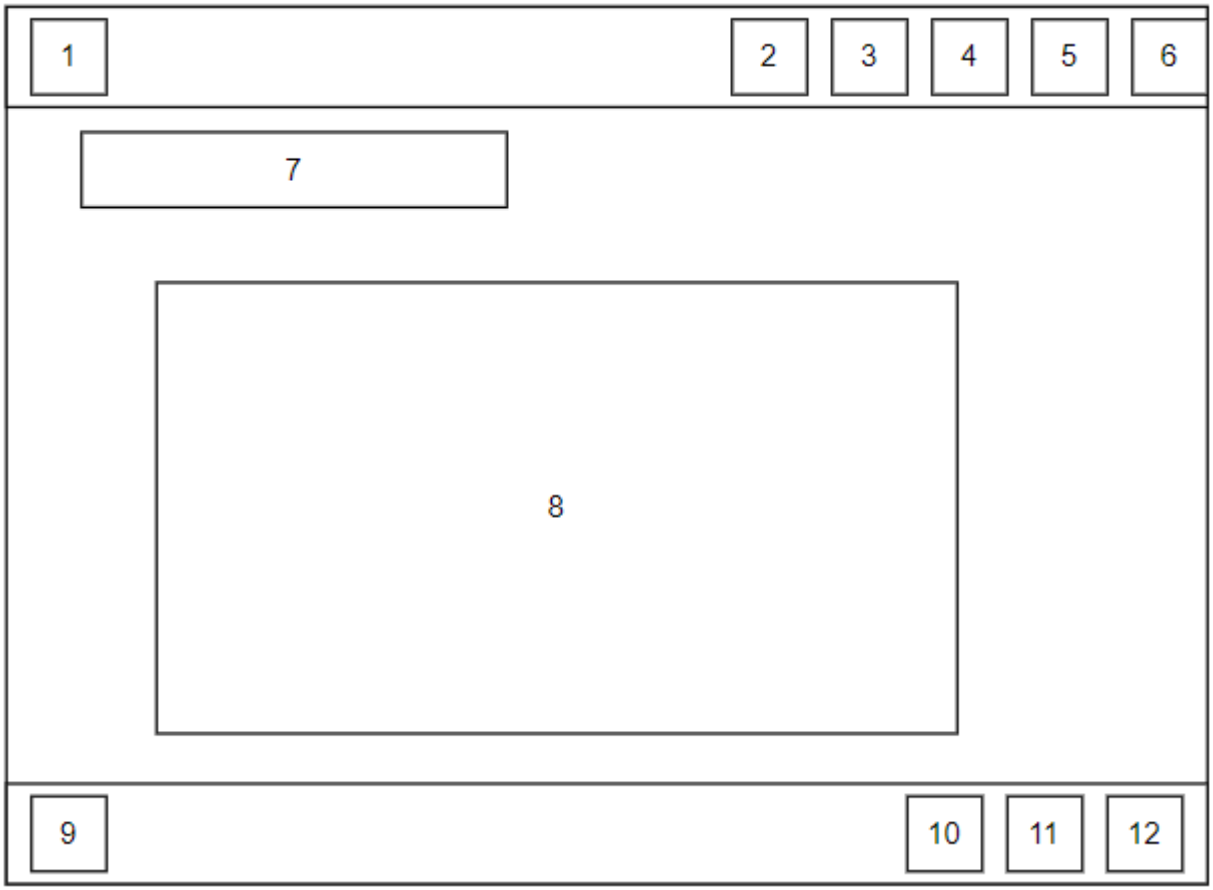


Рисунок 3.5 – Графічна схема головного вікна вебдодатка

Основні елементи інтерфейсу головного вікна касового апарату:

1. Логотип застосунку.
2. Перехід на головну сторінку (неактивний на головній сторінці).
3. Перехід на сторінку замовлень.
4. Перехід на сторінку товарів та послуг.

5. Випадаюче меню для переходу на сторінку працівника або логауту.
6. Випадаюче меню для зміни мови інтерфейсу застосунка.
7. Рядок пошуку.
8. Результуюче вікно пошуку.
9. Авторські дані.
10. FAQ.
11. Перехід на головну сторінку (неактивний на головній сторінці).
12. Інформації про застосунок.

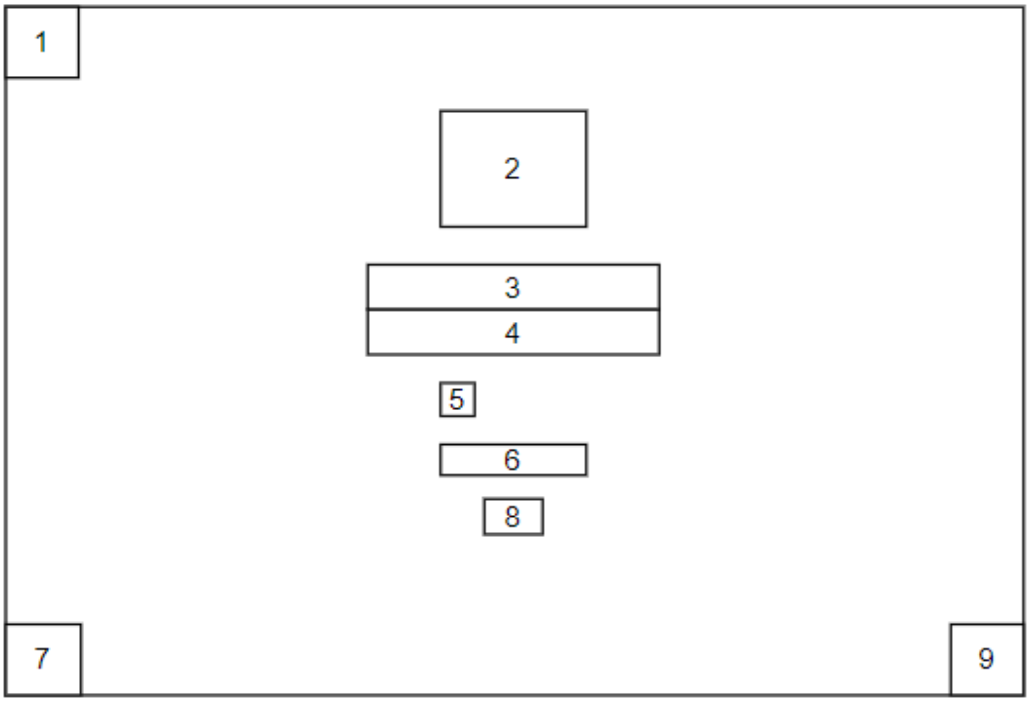


Рисунок 3.6 – Графічна схема сторінки входу у застосунок

Основні елементи інтерфейсу сторінки входу:

1. Випадаюче меню для зміни мови інтерфейсу застосунка.
2. Логотип застосунку.
3. Рядок вводу логіну для авторизації працівника.
4. Рядок вводу паролю для авторизації працівника.
5. Чекбокс для запам’ятовування даних працівника.
6. Кнопка входу.

7. Авторські дані.

8. Кнопка, яка відкриє модульне вікно для нового реєстрації працівника.

9. FAQ.

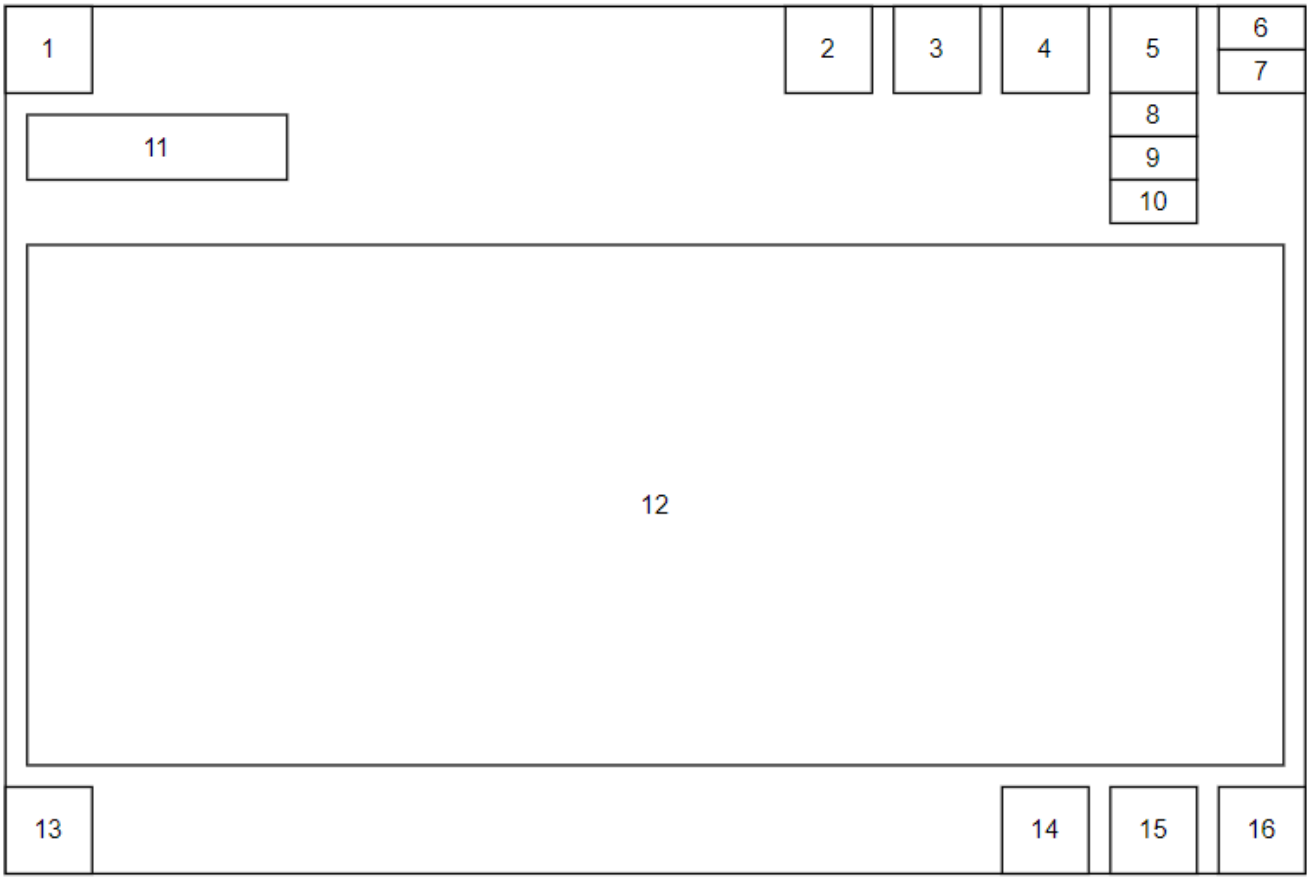


Рисунок 3.7 – Графічна схема головної сторінки вебдодатка

Основні елементи інтерфейсу головної сторінки вебдодатка:

1. Логотип застосунку який направляє на головну сторінку.

2. Елемент навігаційної панелі, відповідає за перехід на головну сторінку веб-додатка.

3. Елемент навігаційної панелі, відповідає за перехід на сторінку створення звіту, чеку та замовлення за допомогою даного веб-додатка.

4. Елемент навігаційної панелі, відповідає за перехід на сторінку створення нових товарів, створення нових, їх видалення чи додавання у замовлення за допомогою даного веб-додатка.

5. Елемент навігаційної панелі з випадаючим меню, відповідає за перехід на особисту сторінку авторизованого користувача.

6. Елемент випадаючого меню навігаційної панелі, відповідає за переведення вебдодатка на українську мову.

7. Елемент випадаючого меню навігаційної панелі, відповідає за переведення вебдодатка на англійську мову.

8. Елемент випадаючого меню навігаційної панелі, відповідає за швидкий виклик модального вікна створення замовлення.

9. Елемент випадаючого меню навігаційної панелі, відповідає за швидкий виклик модального вікна створення нового товару.

10. Елемент випадаючого меню навігаційної панелі, відповідає за логаут користувача вебдодатка.

11. Рядок вводу для здійснення пошуку по записам бази даних.

12. Область виведення інформації.

13. Авторські дані.

14. FAQ.

15. Перехід на головну сторінку (неактивний на головній сторінці).

16. Інформації про застосунок.

Розробка інтерфейсу є важливим етапом у процесі розробки касового апарату, оскільки переважна більшість користувачів в першу чергу звертають увагу на інтерфейс застосунку, яким користуються, і віддають перевагу більш зручним у юзабельності та структурно зрозумілим програмам.

3.5 Програмна реалізація РРО

Для реалізації програмного функціонала касового апарату для реєстрації розрахункових операцій було розроблено велику кількість методів. Основними серед них є наступні:

– отримання дешифрованих даних;

– реєстрації/логін;

– генерування звітів;

– пошук товарів;

– отримання з’єднання з базою даних з пулу з’єднань;

– відображення замовлень;

Першою дією, яку виконує веб-додаток, є ініціалізація контейнера сервлетів Apache Tomcat, що використовуються для реалізації сервлетів та jsp який є базовим набором функціонала веб-додатка. Дана процедура виконуються автоматично тому програмний код для даного функціонала не потрібний.

Алгоритм дешифрування достатньо громіздкий, для демонстрації було обрано функцію отримання закодованого паролю з бази даних та його трансформація для його подальшого використання в алгоритмі дешифрування даних (рис. 3.8).

```
public String getSecurePassword(String passwordToHash, String salt) throws DBException {
    String generatedPassword;
    try {
        MessageDigest md = MessageDigest.getInstance("SHA-512");
        md.update(salt.getBytes(StandardCharsets.UTF_8));
        byte[] bytes = md.digest(passwordToHash.getBytes(StandardCharsets.UTF_8));
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < bytes.length; i++) {
            sb.append(Integer.toString( i: (bytes[i] & 0xff) + 0x100,  radix: 16).substring(1));
        }
        generatedPassword = sb.toString();
    } catch (NoSuchAlgorithmException e) {
        LOGGER.info( s: "Service exception in getSecurePassword");
        throw new DBException(e.getCause());
    }
    return generatedPassword;
}
```

Рисунок 3.8 – Лістинг функцій «getSecurePassword»

В даній функції виконується побайтове перетворення символів з одного формату в інший, увесь масив зберігається в будівельнику рядків (Stringbuilder) та

під час виконання даної функції відбувається логування на випадок утворення помилки.

Для реалізації пошукового рядка необхідного отримати інформацію з рядка вводу на стороні клієнта за допомогою сервлета пошуку і з'єднання з базою даних викликати функцію пошуку по її записам. Лістинг даної функції наведено на рисунку 3.9.

```
public ArrayList<Commodity> findCommodity(String tag){

    ArrayList<Commodity>arr=new ArrayList<>();
    Connection con = null;
    PreparedStatement stmt=null;
    try {
        con=DBConnectionPool.getInstance().getConnection();
        stmt = con.prepareStatement(DBConstants.FIND_COMMODITY);
        stmt.setString( parameterIndex: 1,tag);
        ResultSet rs=stmt.executeQuery();
        while (rs.next())
        {
            arr.add(new Commodity(rs.getString( columnLabel: "name"),rs.getString( columnLabel: "manufacturer"),
                    rs.getString( columnLabel: "description"),rs.getString( columnLabel: "country"),rs.getDouble( columnLabel: "price")));
        }

    } catch (SQLException throwables) {
        throwables.printStackTrace();
        try {
            throw new DBException(throwables.getMessage(),throwables.getCause());
        } catch (DBException e) {
            e.printStackTrace();
        }
    }finally {
        close(con);
        close(stmt);
    }
    return arr;
}
```

Рисунок 3.9 – Лістинг функції «findCommodity»

Отримана інформація повертається на сервлет де вона обробляється і виводиться касиру для подальших маніпуляцій таких як додавання знайденого товару до чеку чи можливі коригування з інформацією про товар, такі як його вартість чи кількість на складі.

Для реєстрації нового працівника повторного виклику головного меню касового апарату, виконується ряд послідовних дій, спочатку на jsp сторінці

зчитуються дані які ввів користувач, усі вони перенаправляються на сервлет де обирається зв'язок з базою даних виконується пошук по базі даних чи був раніше зареєстрований користувач, якщо був тоді перенаправляє на сторінку реєстрації пройти реєстрацію знову, якщо цей запис не знайде тоді він записується до бази даних і перенаправляється до головної сторінки. Лістинг функції запису користувача у базу даних наведено на рисунку 3.10.

```
public boolean insertEmploye(Employe employe) {
    Connection con = null;
    PreparedStatement stmt=null;
    try {
        con=DBConnectionPool.getInstance().getConnection();
        con.setAutoCommit(false);
        stmt=con.prepareStatement(DBConstants.SIGNUP_EMPLOYE);

        stmt.setString( parameterIndex: 1,employe.getName());
        stmt.setString( parameterIndex: 2,employe.getSurname());
        stmt.setString( parameterIndex: 3,employe.getPhone());
        stmt.setString( parameterIndex: 4,employe.getEmail());
        stmt.setString( parameterIndex: 5,employe.getPwd());
        stmt.setString( parameterIndex: 6,employe.getPosition());

        stmt.executeUpdate();
        con.commit();
        return true;
    } catch (SQLException throwables) {
        throwables.printStackTrace();
        rollBack(con);
        try {
            throw new DBException(throwables.getMessage(),throwables.getCause());
        } catch (DBException e) {
            e.printStackTrace();

        }
    } finally {
        close(stmt);
        close(con);
    }
    return false;
}
```

Рисунок 3.10 – Лістинг функції реєстрації користувача

Для встановлення стабільного з’єднання з базою даних використано розроблений Connection pool, який дозволяє використовувати один із заздалегідь підготовлених каналів з’єднання з базою даних. Якщо використовувати простіший тип з’єднання то при великій кількості запитів з різних пристроїв серверна частина веб-додатку буде оброблювати та відповідати на клієнтську сторону веб-додатку з помилками, пул з’єднань усуває таку проблему і також пришвидшує відповідь на запити. Лістинг функцій ініціалізації та отримання підготовленого з’єднання з базою даних зображено на рисунку 3.11.

```
public Connection getConnection(){
    Context ctx;
    Connection con = null;
    try {
        ctx = new InitialContext();
        DataSource ds = (DataSource)ctx.lookup( name: "java:comp/env/jdbc/mydatabase");
        con = ds.getConnection();
    } catch (NamingException | SQLException e) {
        e.printStackTrace();
    }
    return con;
}
```

Рисунок 3.11 – Лістинг функції «getConnection»

Для відображення усіх замовлень які були виконані за допомогою вебдодатка за допомогою сервлета замовлень буде виконано функцію отримання замовлення вони є у кількох варіаціях, за конкретним працівником, за номером замовлення, за його статусом та показ усіх замовлення.

На рисунку 3.12 зображено лістинг функції для отримання усіх замовлень.

```
public List<Order> getAllCustomerOrders(long userId) throws DBException {
    List<Order> list;
    try (Connection connection = DBConnectionPool.getInstance().getConnection();
         PreparedStatement statement = connection.prepareStatement(DBConstants.GET_ALL_ORDERS_BY_USER_ID)) {
        statement.setLong( parameterIndex: 1, userId);
        try (ResultSet rs = statement.executeQuery()) {
            list = getOrderListFromResultSet(rs);
        }
    } catch (SQLException throwables) {
        throw new DBException(throwables.getMessage(),throwables.getCause());
    }
    return list;
}
```

Рисунок 3.12 – Лістинг функції «getAllCustomerOrders»

Для отримання звіту реалізовано сервлет ReportControler (рис. 3.13.) з його допомогою викликаються функції для отримання звіту.

```
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/ReportControler")
public class ReportControler extends HttpServlet {

    private ReportRate reportRate;

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        reportRate = (ReportRate) config.getServletContext().getAttribute( s: "reportRate");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        long orderId = Long.parseLong(req.getParameter( s: "orderId"));
        Report report = new Report();
        report.setId(orderId);
        report.setComment(req.getParameter( s: "comment"));
        report.setRate(ReportRate.valueOf(req.getParameter( s: "rate")));
        HttpSession session = req.getSession();
        session.setAttribute( s: "infoMessage", reportRate.saveReport(report));
        resp.sendRedirect( s: "order?orderId=" + orderId);
    }
}
```

Рисунок 3.13 – Лістинг сервлета «ReportControler»

Даний сервлет виконує запит до бази даних по ідентифікаційному номеру замовлення, повертає його у вигляді екземпляра класу Report зберігає його у сесії сервлета для подальшого відображення на html сторінці вебдодатка.

Отримання даних звіту виконується за допомогою функції getReport за допомогою sql-запита дістаються дані про замовлення з бази даних до них додатково додаються певні поля, а саме дата створення звіту, аналіз і текст. Лістинг даної функції зображено на рисунку 3.14.

```
public Report getReport(long orderId) {
    Report report = null;
    try (Connection connection = connectionPool.getConnection();
         PreparedStatement statement = connection.prepareStatement(DBConstants.GET_REPORT)) {
        statement.setLong( parameterIndex: 1, orderId);
        try (ResultSet rs = statement.executeQuery()) {
            if (rs.next()) {
                String comment = rs.getString( columnLabel: "Report_text");
                if (comment != null) {
                    report = new Report();
                    report.setComment(comment);
                    report.setRate(ReportRate.valueOf(rs.getString( columnLabel: "rate")));
                    report.setTime(rs.getTimestamp( columnLabel: "creating_time"));
                }
            }
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
        try {
            throw new DBException(throwables.getMessage(),throwables.getCause());
        } catch (DBException e) {
            e.printStackTrace();
        }
    return report;
}
```

Рисунок 3.14 – Лістинг функції «getReport»

Отже, у підрозділі 3.3 було описано програмний код основних алгоритмів роботи розробленого касового апарату для реєстрації розрахункових операцій. Повний лістинг наведено у додатку Б.

3.6 Висновки

У даному розділі було проведено аналіз мов програмування C#, Java та Python, середовищ розробки Eclipce, Netbeans та Intellij IDEA і систем управління базами даних PostgreSQL, MySQL та SQLite. В результаті даного аналізу було прийнято рішення використовувати мову програмування JAVA, середовище розробки Intellij IDEA та СУБД MySQL для розробки веб-додатку програмного реєстратора розрахункових операцій, а саме касового апарату. Крім того, було проведено опис алгоритмів основних програмних модулів касового апарату

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення (software testing) – це процес аналізу чи експлуатації програмного забезпечення з метою виявлення дефектів. Як правило, даний етап циклу розробки програмного забезпечення є найтривалішим та найбільш ресурсозатратним. Виділяють два основних види тестування: статичне та динамічне.

Інтеграційне тестування – у ході інтеграційного тестування перевіряється, чи добре працюють разом різні модулі та сервіси, що використовуються програмою. Наприклад, можна протестувати взаємодію з базою даних або переконатися, що мікросервіси працюють разом так, як задумано. Цей вид тестування є більше витратним, оскільки проведення тестів потрібно запуск різних компонентів докладання.

Функціональне тестуваня – у функціональних тестах основна увага приділяється бізнес-вимогам до застосування. Вони перевіряють лише результат деякої дії та не перевіряють проміжні стани системи при виконанні цієї дії. Іноді виникає плутанина між поняттями інтеграційних та функціональних тестів, так як і ті та інші вимагають взаємодії кількох компонентів один з одним. Різниця в тому, що інтеграційний тест потрібний просто щоб переконатися, що ви можете надсилати запити до бази даних, тоді як функціональний тест буде очікувати на отримання з бази даних певного значення відповідно до вимог продукту [23]:

Приймальне тестування – це формальні тести, які мають перевірити, чи відповідає система вимогам бізнесу. При цьому необхідний запуск усієї програми, і основна увага приділяється відтворенню поведінки користувачів. У ході цього тестування можливе навіть замір продуктивності системи, і у разі невідповідності встановленим вимогам внесені зміни можуть бути відхилені.

– тестування «чорної скриньки» – це вид динамічного тестування, який передбачає відсутність у тестувальника знань про внутрішню будову програмного

додатку, про його алгоритми та принципи виконання функцій. При даному виді тестування відбувається перевірка виключно зовнішніх інтерфейсів додатку;

– тестування «білої скриньки» – це вид динамічного тестування, який передбачає повне розуміння тестувальником принципів роботи програмного додатку, що тестується, особливостей роботи його алгоритмів та принципів виконання його функцій;

Після визначення особливостей кожного із розглянутих методів тестування програмного забезпечення було вирішено використати методику функціонального тестування [24], оскільки використання саме цієї методики тестування касового апарату дозволить перевірити готовність функціонування додатку та можливі неполадки у роботі його основних алгоритмів, чого не завжди можна досягти при використанні інших методів тестування.

4.2 Тестування розробленого програмного продукту

Щоб перевірити основні функціональні можливості застосунку було обрано функціональне тестування, адже воно слугує для виявлення розбіжностей між поведінкою реально реалізованих функцій і очікуваною поведінкою відповідно до визначених специфікації і вимог. Функціональні тести мусять охоплювати всі реалізовані функції з урахуванням найбільш ймовірних допущених типів помилок. Сценарії тестування, що поєднують різні окремі тести, орієнтовані на перевірку якісного розв'язку функціональних задач.

Тест-кейс №1 – Тестування авторизації працівника:

1. Запускаємо сервер застосунку, та відкриваємо його в браузері.

2. На сторінці логіну вводимо дані які співпадають із раніше зареєстрованими в під’єднаній базі даних MySQL зображено на рисунку 4.1.

3. Потрапляємо на головну сторінку.

4. Для наочної демонстрації переходимо в особистий кабінет користувача зображено на рисунку 4.2.

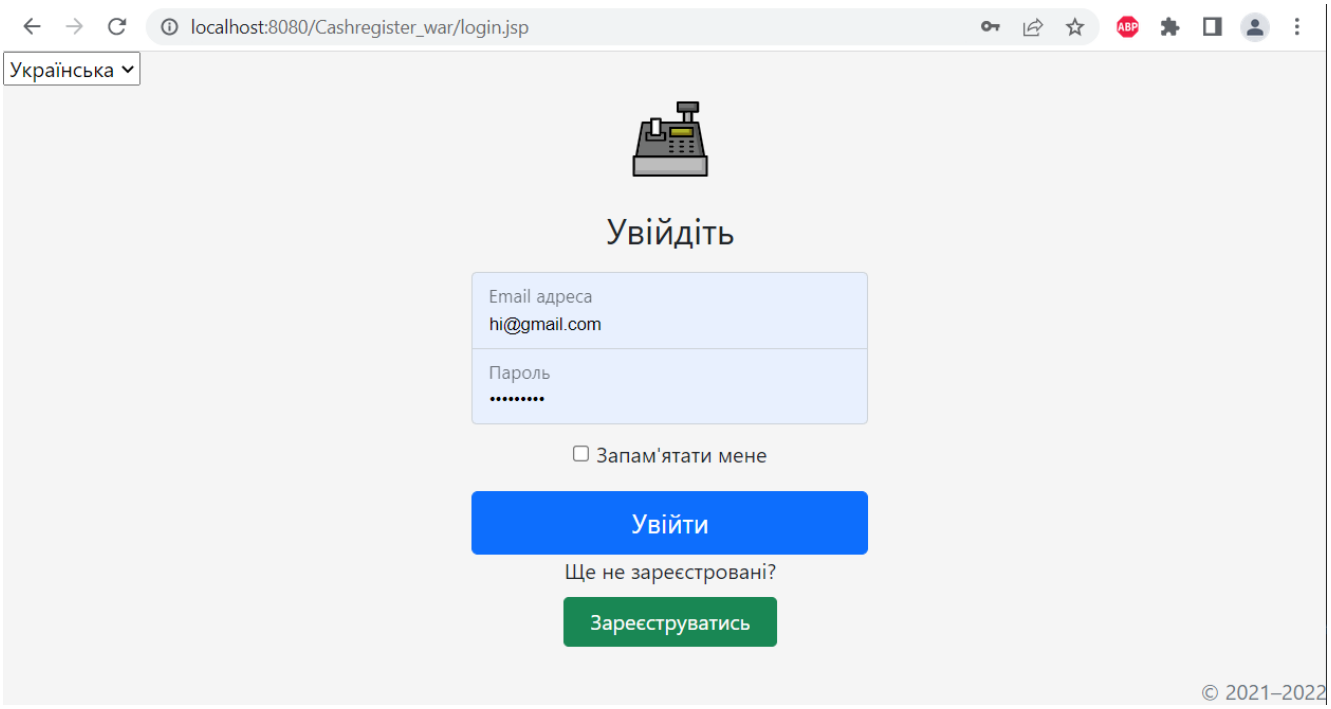


Рисунок 4.1 – Сторінка логіну користувача

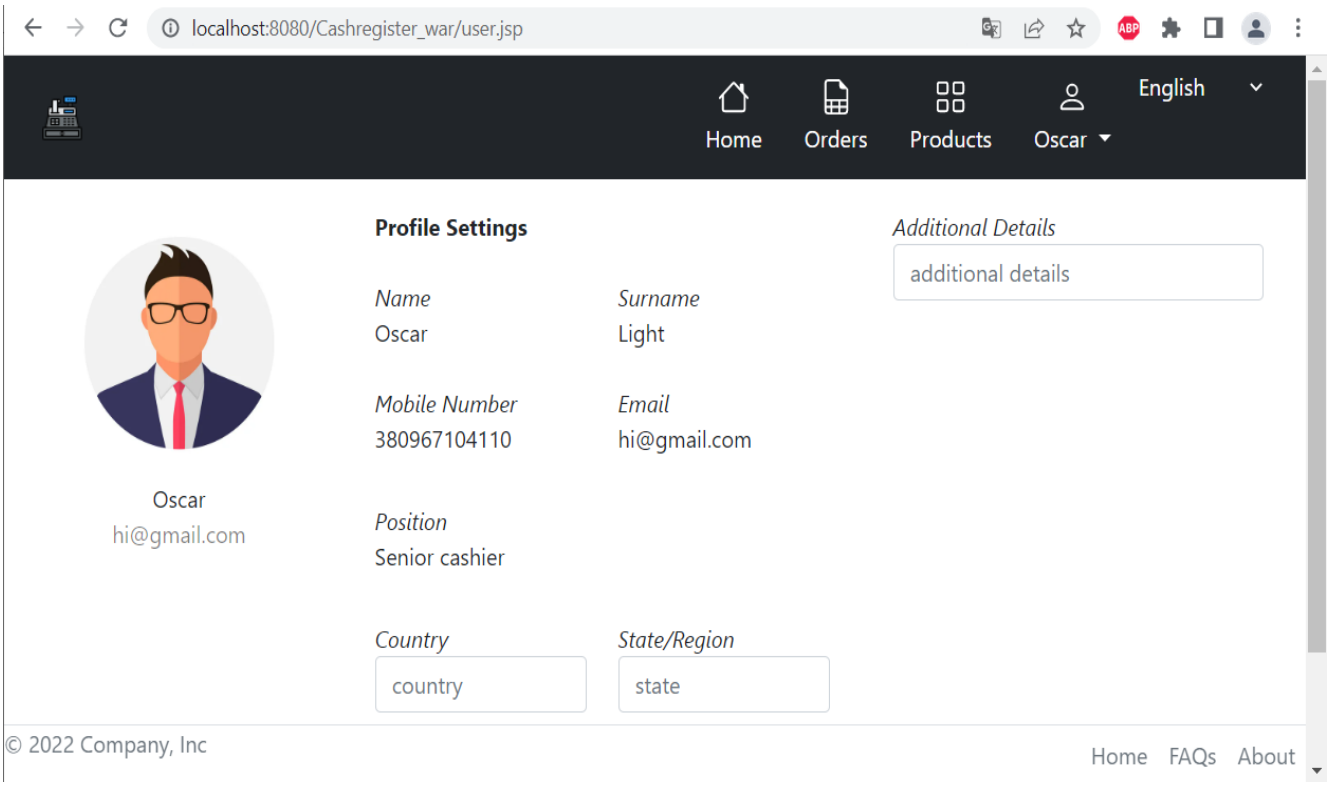


Рисунок 4.2 – Особиста сторінка користувача

Можна констатувати факт того, що результат виконання відповідає очікуваному, отже тест-кейс №1 успішно пройдено.

Для тестування шифрування перевіримо процес реєстрації під час якого в базі даних пароль користувача повинен зберігатись в зашифрованому виді.

Тест-кейс №2 – Шифрування конфіденційної інформації:

1. Перейти на сторінку логіну.
2. Натиснути зелену кнопку “Зареєструватись”.
3. В модальному вікні (рис. 4.3.) заповнити реєстраційну форму.
4. Погодиться з правилами та натиснути синю кнопку “Зареєструватись”.
5. Переглянути запис базі даних чи був він зашифрований (рис. 4.4.).

Очікуваним результатом у даному випадку є перехід на головну сторінку додатку, а у базі даних зявиться запис із щойно створеним акаунтом користувача веб-додатка.

Перед виконанням реєстрації вібувається перевірка в базі даних чи не було раніше зареєстровано схожого запису працівника за полями логіну та мобільного телефону і якщо буде знайденно аналогічне поле то реєстрація не відбудеться і користувача веб-додатку перенаправить на сторінку відновлення доступу . Результат виконання тест-кейсу наведено на рисунку 4.4.

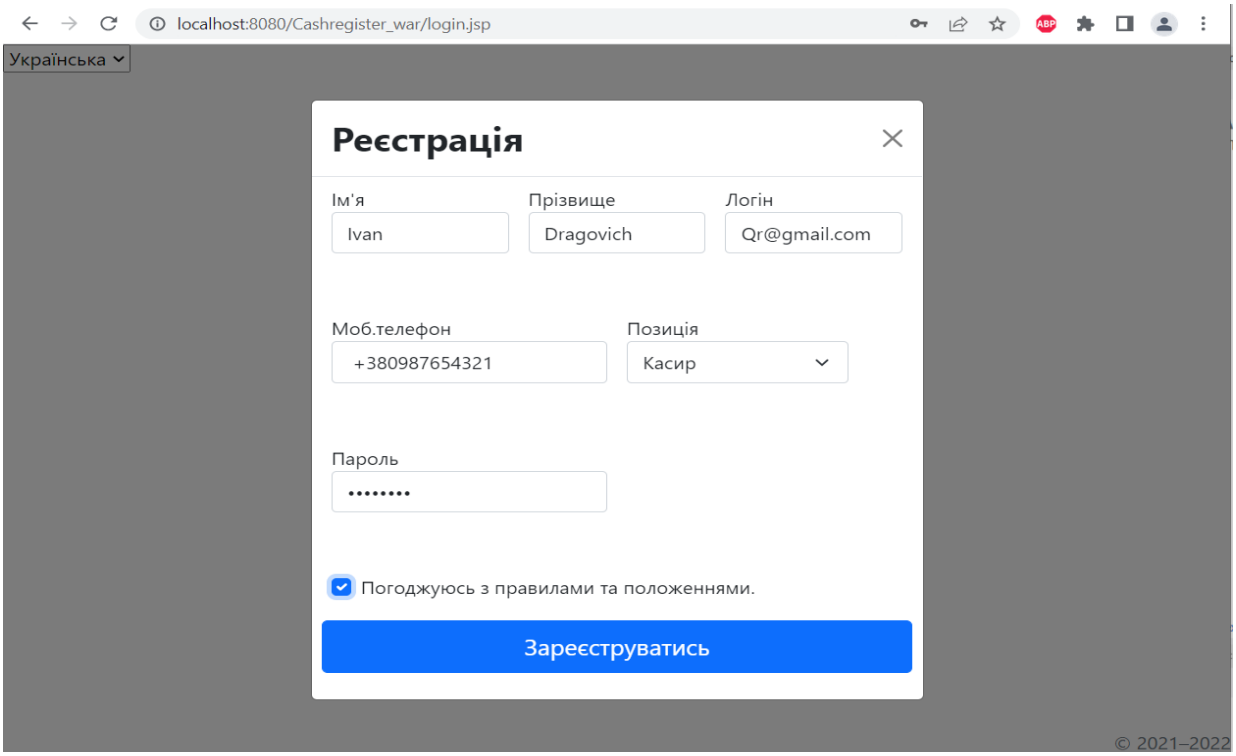


Рисунок 4.3 – Вікно реєстрації

```
1 •  SELECT * FROM cashregister.employe;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce

id	name	surname	phone	email	position	pwd
1	Oscar	Light	380967104110	hi@gmail.com	Senior c...	ZXCVBN11q
2	Ivan	Dragovich	+380987654321	Qr@gmail.com	Cashier	DJcdOIG0L1l22S6aYmJglLsWb8xh
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 4.4 – База даних з даними працівників

В результаті виконання тест кейсу пароль працівника з ID – 2 було зашифровано для наглядного прикладу попередній працівник був зареєстрований без використання шифрування. Тобто, фактичний результат відповідає очікуваному, тому тест-кейс №2 успішно пройдено.

Тест-кейс №3 – Формування графічного звіту:

1. Після авторизації переходимо на сторінку заказів (рис. 4.5.).

2. Натиснути кнопку X-звіт.

3. Отримуємо звіт (рис. 4.6).

Звіт формується на основі виконаних заказів за один день в ньому представлена інформація про надані фінансові послуги.

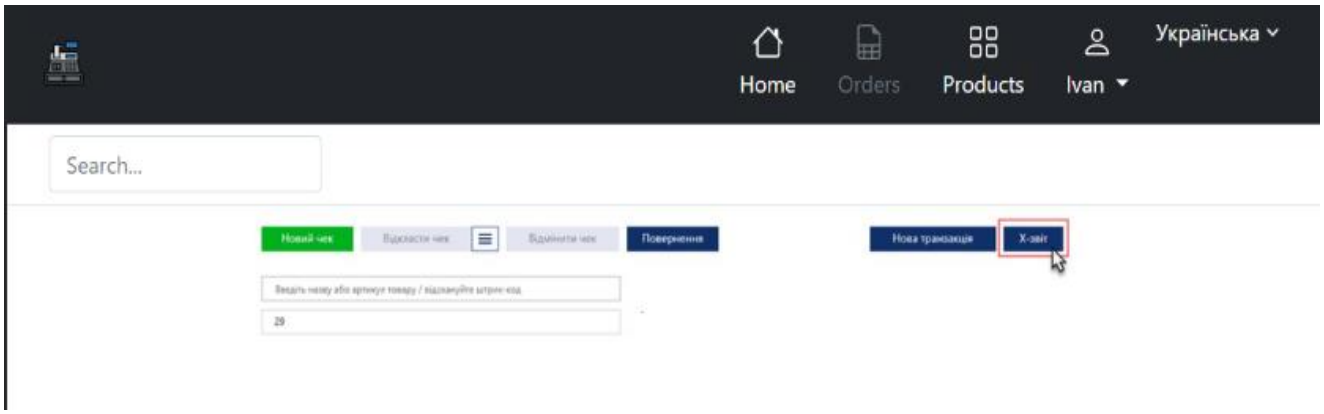


Рисунок 4.5 – Сторінка заказів для формування чеків та звітів

Cash register ТОВ

Звіт

З: 28/05/2022 12:33 по: 28/05/2022 15:33
Дата друку: 28/05/2022 17:58
PPO ID: 01
Номер звіту: 94
Працівник: 08
Товар: Весь

Аналіз

Оплата готівкою	1290.20
Знижка	4.56
Загалом готівкою	1285.64
Оплата чеком	122.40
Знижка	0.00
Загалом чеком	122.40
Оплата карткою	2865.40
Знижка	31.20
Загалом карткою	2834.20
Безготівкова оплата	16.20
Знижка	0.00
Загалом безготівки	16.20
Загальна сума усіх видів	4258.44
Кількість платежів	13

Аналіз типів оплати

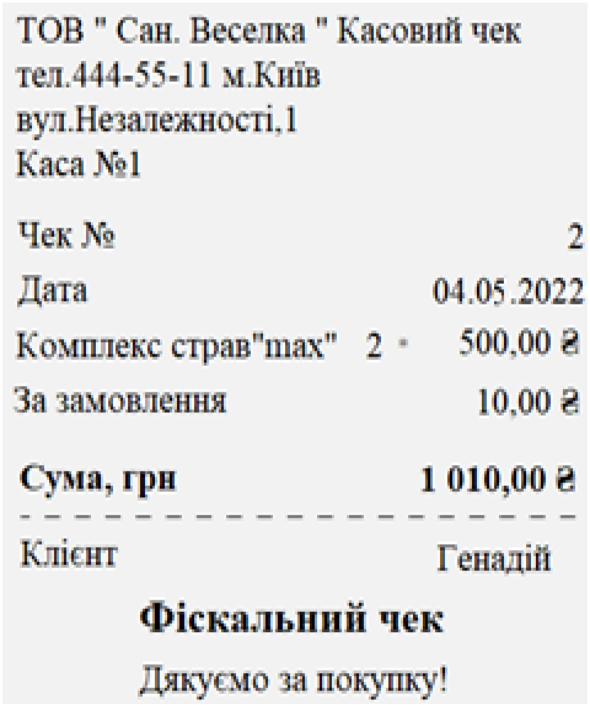
Готівкою	1285.64
Чеком	122.40
Карткою	2834.20
Безготівкою	16.20

Рисунок 4.6 – Результат виконання тест-кейсу №3

В результаті виконання тест кейсу було сформовано Х-звіт. Тобто, фактичний результат відповідає очікованому, тому тест-кейс №3 успішно завершений.

Тест-кейс №4 – Формування чеку:

1. Переходимо на сторінку заказів (рис. 4.5.).
2. Натиснути кнопку новий чек.
3. Заповнюємо форму.
4. Отримуємо чек (рис. 4.7).


ТОВ " Сан. Веселка " Касовий чек
тел.444-55-11 м.Київ
вул.Незалежності,1
Каса №1

Чек № 2
Дата 04.05.2022
Комплекс страв"max" 2 * 500,00 ₴
За замовлення 10,00 ₴

Сума, грн **1 010,00 ₴**

Клієнт Генадій

Фіскальний чек

Дякуємо за покупку!

Рисунок 4.7 – Результат виконання тест-кейсу.

Фактичний результат виконання даного тест-кейсу повністю відповідає очікуваному, отже тест-кейс №4 успішно пройдено.

Для наступного тест кейсу переходимо на сторінку Products вебдодатку.

Тест-кейс №5 – Створення товару товару:

1. На сторінці товарів (рис. 4.8.), для виклику модального вікна натискаємо кнопку створення товару.

2. У появившсяшому модальному вікні заповнюємо всі поля, заповнене модальне вікно зображено на рисунку 4.9.

3. Підтверджуємо та натискаємо створити.

4. Шукаємо щойно сторений запис товару у базі даних вебдодатка, результат пошуку запису у базі даних зображено на рисунку 4.10.

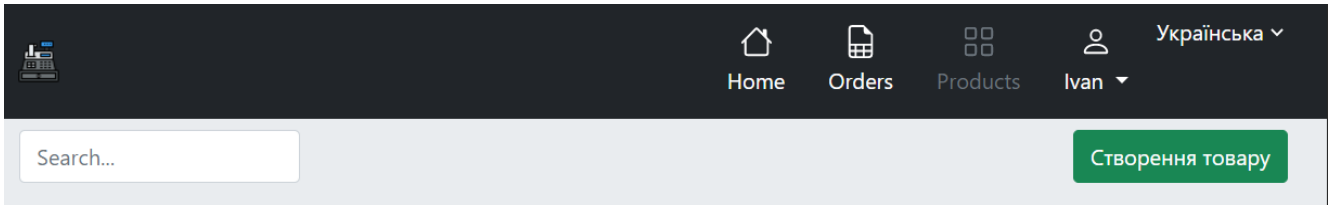


Рисунок 4.8 – Сторінка створення товару.

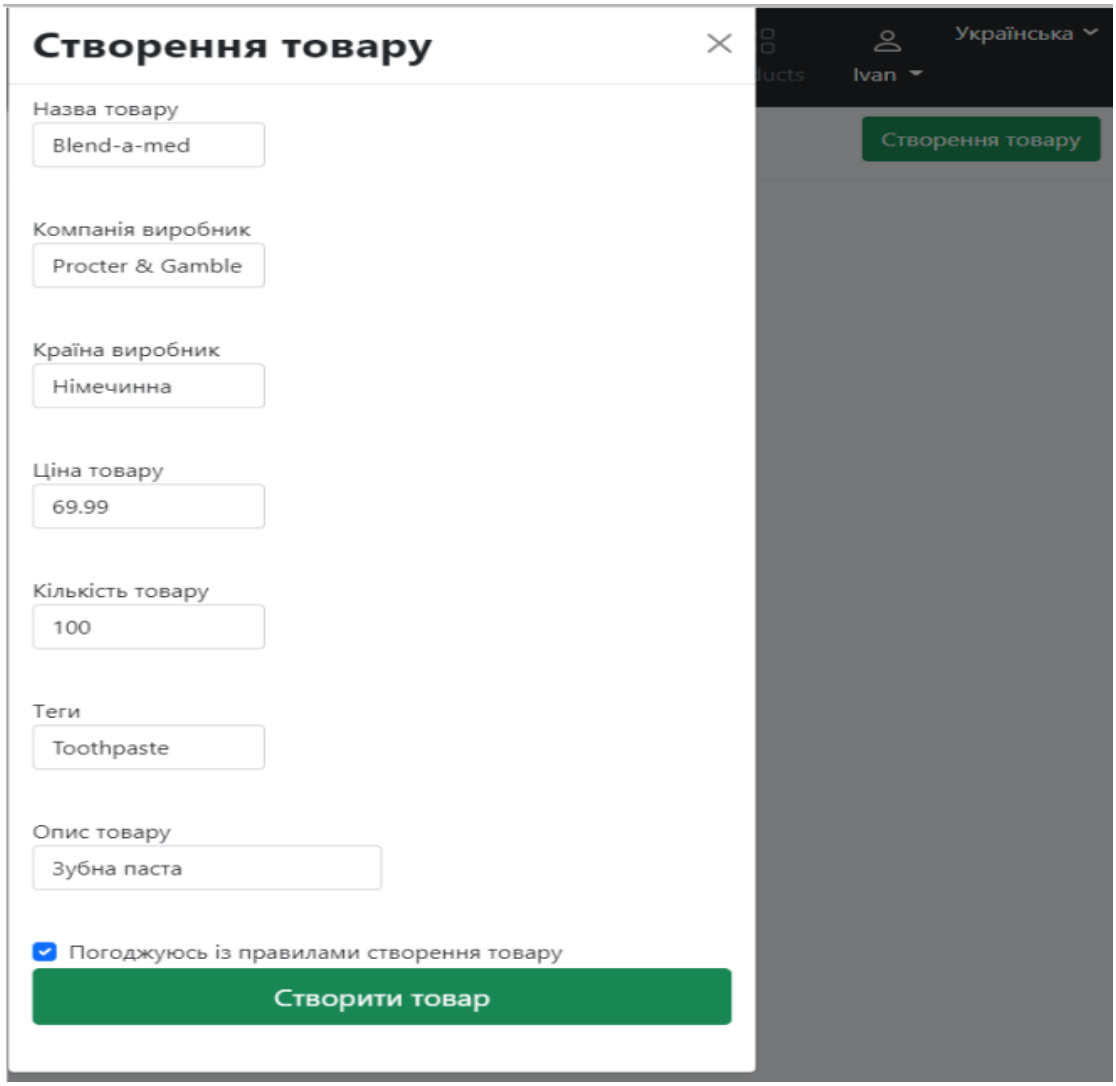


Рисунок 4.9 – Модальне вікно для створення товару

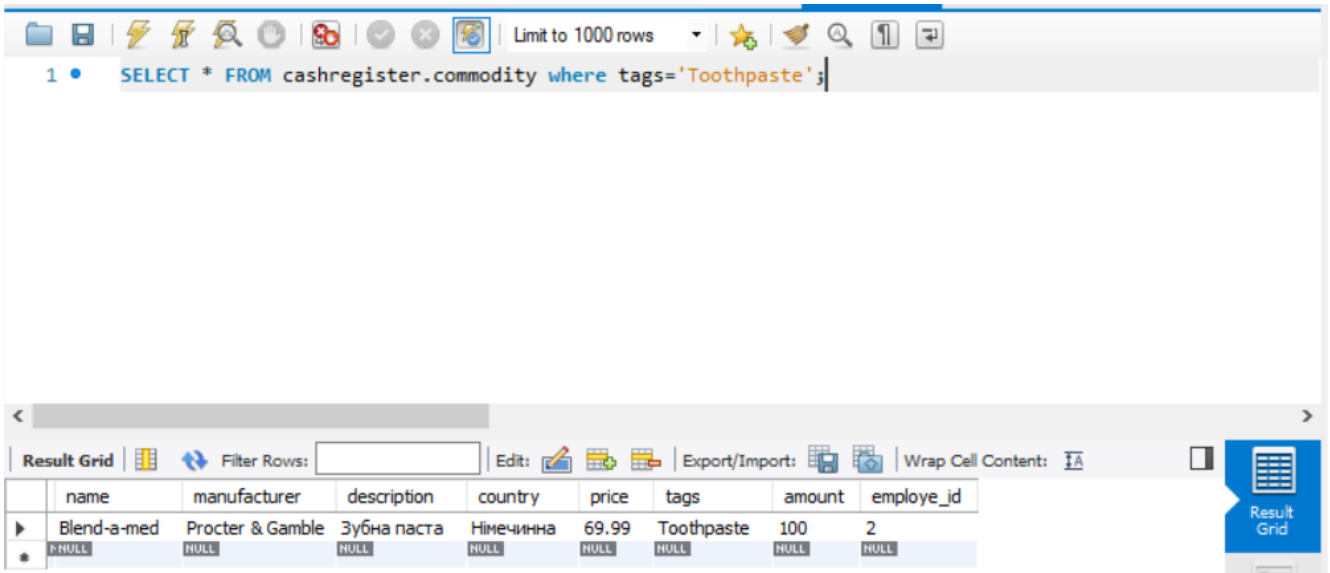


Рисунок 4.10 – Результат виконання тест кейсу

Фактичний результат виконання даного тест-кейсу повністю відповідає очікуваному, про це також свідчить поле employe_id яке вказує на працівника що створив даний товар, отже тест-кейс №5 успішно пройдено.

При проведенні тестування програмного застосунку було отримано повну відповідність фактичних результатів очікуваним. Помилок чи збоїв при роботі застосунку виявлено не було, застосунок працював в повні відповідності до поставлених вимог, як було прогнозовано відповідно до описаного функціонала. У результаті доведено повну функціональність програмного застосунку та його відповідність поставленому технічному завданню.

4.3 Розробка інструкції користувача

Оскільки “Cash register” вебдодаток, то для його використання достатньо браузера з підтримкою сучасних стандартів HTML та CSS, додаткових програм встановлювати не потрібно.

Доступ до функціоналу вебдодатка надається після авторизації, сам процес був описаний в попередніх розділах.

Якщо у авторизованого працівника роль касир, то у його можливості входить пошук товарів за допомогою поля пошуку, формування замовлення за допомогою форми чи графічно, для цього працівник переходить у вкладку Products та за допомогою каталогу шукає необхідний для внесення у замовлення наявний на складі товар.

Якщо у авторизованого працівника роль старший касир, то до можливостей звичайного касира додається можливость редагувати вже наявні замовлення у яких статус поки що не завершений, він має змогу як скасувати так і внести правки до замовлення.

Коли у авторизованого працівника роль товарознавець, тоді у нього немає доступу до формування замовлень, але з’являються власні можливості до них відносять, створення нових товарів у вкладці Products, редагування наявних товарів

їх назву, ціну, кількість на складі, опис, а також має змогу видалити застарілі позиції товарів, якщо вони довго не перебували у наявності.

Для обмеження доступу певному працівнику необхідного авторизуватись з ролю адміністратора тоді в особистому кабінеті зявиться список з наявними працівниками та можливість обмежити їх функціонал у випадку коли необхідно.

Для зміни мови інтерфейсу на будь-якій сторінці вебдодатку у правому верхньому куті

Отже, було розроблено інструкцію користувача. В ній описані дії, які потрібно виконати для початку використання повного функціонала касового апарату та можливі варіанти взаємодії користувача із ним.

4.4 Вимоги до пристрою

Так як розроблений веб-додаток працює на базі браузера, то для його використання користувачу потрібний встановлений на його пристрої будь-який із існуючих на даний момент браузерів оновлених до останньої версії. Веб-браузер можна встановити для усіх смартфонів на базі операційних систем Android та iOS, а також на десктопні операційні системи Windows, Mac OS X та Linux. Мінімальна та рекомендована конфігурації персонального комп'ютера на базі операційної системи Windows наведено у таблицях 4.1 та 4.2. Єдиним виключенням з веб браузерів є Safari на базі Windows через його непідтримку фреймворком Bootstrap.

Таблиця 4.1 – Мінімальна конфігурація

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1,6 ГГц
Об'єм оперативної пам'яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-розрядної системи
Місце на жорсткому диску	1 ГБ
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX9
Операційна система	Windows 7

Таблиця 4.2 – Рекомендована конфігурація

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 2,4 ГГц
Об’єм оперативної пам’яті	4 ГБ для 32-розрядної системи і 8 ГБ для 64-розрядної системи
Місце на жорсткому диску	1 ГБ
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX10
Операційна система	Windows 10

Веб-додаток не вимагає інсталяції, так як доступ до нього можна отримати із браузера, адже він працює на віддаленому сервері. Для того, щоб почати використовувати функціонал касового апарату достатньо авторизуватись під своїм обліковим записом

4.5 Висновки

Для проведення тестування веб-додатка касового апарату для реєстрації розрахункових операцій, яка передбачає перевірку функціональності додатку за допомогою тест-кейсів. Результат тестування програмного продукту довів його повну працездатність та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача, яка допоможе користувачу розпочати використання касового апарату та ознайомить його із способами використання його функціонала. Крім того було визначено мінімальну та рекомендовану конфігурації персональних комп’ютерів, які необхідні для правильної та ефективної роботи веб-додатку.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено веб-додаток касовий апарат, який виконує функціонал програмного реєстратора розрахункових операцій та має назву «Cash register». Розроблений веб-додаток призначений для підвищення ефективності виконання розрахункових операцій та збереження інформації про них, ведучи фінансовий моніторинг за допомогою звітів.

Виконано аналіз існуючих систем для реєстрації розрахункових операцій. В результаті аналізу аналогів було висвітлено їхні основні недоліки. Проведено їхнє порівняння із власним програмним продуктом, в результаті було визначено доцільність розробки нового програмного додатку для контролю особистих витрат. Досліджено причини відмови користувачів від використання програмних аналогів класичного касового апарату. Виконано постановку задач розробки програмного продукту.

Здійснено аналіз варіантів керування графічним інтерфейсом касового апарату, та обрано найефективніші та найзручніші для користувача. Перед написанням програмного веб-додатку було розроблено блок-схеми алгоритмів шифрування та дешифрування в базі даних і генерування текстових та графічних звітів та чеку. Розроблено основні модулі програми.

Проведено варіантний аналіз та обґрунтування вибору програмних засобів при розробці програми, обрано мову програмування Java, середовище розробки Intellij IDEA та СУБД MySQL. В процесі розробки було описано програмну реалізацію основних модулів програми.

Розглянуто основні види та методи тестування, обрано методику функціонального тестування. Проведено тестування за допомогою ряду тест-кейсів, яке довело повну працездатність програмного додатку та його відповідність поставленому технічному завданню. Розроблено інструкцію користувача та вимоги до персонального комп’ютера користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Diamandis P. H., Kotler S. Future is faster than you think: how converging technologies are transforming business, industries, and our lives. Simon & Schuster, 2020. 384 p.
2. Що таке РРО та касовий апарат? Програмне РРО: альтернатива касовому апарату в 2022 році – Law firm "Viktor Reshetov & Partners". Адвокат | Юридическая фирма "Виктор Решетов и Партнеры" - Адвокат Решетов Виктор Викторович. URL: https://vreshetov.com/ua/obrazcy-dokymentov/ekspertni-statti/ppo-ta-kasovi-aparaty-2020-novi-pravyla-roboty-z-hotivkoyu/rro/scho-take-rro-ta-kasovyi-aparat (дата звернення: 10.06.2022).
3. Максимюк О. Особливості професії: продавець-касир супермаркету. Погляд. URL: https://pogliad.ua/news/ukraine/osoblivosti-profesiyi-prodavec-kasir-supermarketu-141456 (дата звернення: 10.06.2022).
4. Переконання, ставлення. Pidru4niki. URL: https://pidru4niki.com/1135071763214/marketing/perekonannya_stavlennya (дата звернення: 13.06.2022).
5. Шинкарчук О.А., Ліщинська Л.Б. Програмний реєстратор розрахункових операцій як заміна класичному касовому апарату. LI науково-технічній конференції підрозділів Вінницького національного технічного університету, м.Вінниця, 31 трав. 2022 р. Вінниця, 2022. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15644.
6. Шинкарчук О. А., Романюк О. В., Романюк О. Н. Можливості технології DLSS 2.0: масштабування зображень в режимі реального часу. X Міжнародна науково-практична конференція “PRIORITY DIRECTIONS OF SCIENCE AND TECHNOLOGY DEVELOPMENT” : Міжнар. науково-практ. конф., м. Київ, 13–15 черв. 2021 р.
7. Slovo і Dilo. ФОПи проти РРО: що саме не влаштовує підприємців та чи є підстави для занепокоєння. Слово і Діло. URL: https://www.slovoidilo.ua/2021/12/17/stattja/finansy/fopy-proty-rro-same-ne-vlashtovuye-pidpryyemcziv-ta-chy-ye-pidstavy-zanepokoyennya (дата звернення: 10.06.2022).

8. Порівняння симетричних з асиметричними криптосистемами. Сучасні криптосистеми. URL: https://sites.google.com/site/sucasnikriptosistemik/home/porivnanna-simetricnih-z-asimetricnimi-kriptosistemami (дата звернення: 10.06.2022).
9. Mollin R. A. RSA and Public-Key Cryptography. Chapman & Hall/CRC, 2002. 304 p.
10. Мови програмування що для чого. Вибір мови програмування. Портал про операційні системи. URL: https://crashbox.ua/tools-to-help/programming-languages-for-what-choice-of-programming-language/ (дата звернення: 10.06.2022)
11. Stellman A. Head first C# (head first). O'Reilly Media, 2010.
12. Eckel B. Thinking in java. 4th ed. Upper saddle River, NJ : Prentice Hall, 2006.
13. Techopedia. What is an integrated development environment (IDE)? - definition from techopedia. Techopedia.com. URL: https://www.techopedia.com/definition/26860/integrated-development-environment-ide (date of access: 10.06.2022).
14. Techopedia. What is Eclipse and the foundation behind it? - Definition from Techopedia. Techopedia.com. URL: https://www.techopedia.com/definition/7755/intellij-idea (date of access: 10.06.2022)
15. McKenzie C. What is netbeans? - definition from whatis.com. TheServerSide.com. URL: https://www.theserverside.com/definition/NetBeans (date of access: 10.06.2022).
16. Techopedia. What is IntelliJ IDEA? - Definition from Techopedia. Techopedia.com. URL: https://www.techopedia.com/definition/7755/intellij-idea (date of access: 10.06.2022).
17. Основні поняття реляційних БД: нормалізація, зв'язок та ключі ⋆ ДПА и ЗНО онлайн. ДПА и ЗНО онлайн. URL: https://bondarenko.dn.ua/osnovni-ponyattya-relyatsijnih-bd-normalizatsiya-zv-yazok-ta-klyuchi/ (дата звернення: 10.06.2022).
18. What is a relational database? – amazon web services (AWS). Amazon Web Services, Inc. URL: https://aws.amazon.com/relational-database/?nc1=h_ls (date of access: 10.06.2022).
19. Database management software, data access components, developer tools. *Devart Software*. URL: https://www.devart.com/dbforge/postgresql (date of access: 10.06.2022).

20. What Is MySQL? A Beginner-Friendly Explanation. Kinsta®. URL: https://kinsta.com/knowledgebase/what-is-mysql/ (date of access: 10.06.2022).
21. Rush A. SQLite with swift tutorial: getting started. *raywenderlich.com*. URL: https://www.raywenderlich.com/6620276-sqlite-with-swift-tutorial-getting-started (date of access: 10.06.2022).
22. Get started with Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/docs/5.2/getting-started/introduction/ (date of access: 10.06.2022).
23. Романюк О., Савчук Т. Організація баз даних і знань : навч. посіб. Вінниця : УНІВЕРСУМ-Вінниця, 2003. 123 с.
24. Функціональне тестування проти нефункціонального тестування - Інший. Огляди, Ігри, Розваги, Червень 2022. URL: https://uk.myservername.com/functional-testing-vs-non-functional-testing (дата звернення: 10.06.2022).

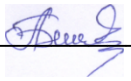
ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедрою ПЗ
д.т.н., проф. О. Н. Романюк
_____
31 березня 2022 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного забезпечення для касового апарату» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
 д.т.н., проф. каф. ПЗ Л.Б. Ліщинська
березня 2022 р.
Виконав:
студент гр. 3ПІ-18б О.А. Шинкарчук
березня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка касового апарату».

Галузь застосування – фінанси.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №66 від «24» березня 2022 р. ректора по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою бакалаврської роботи є підвищення ефективності проведення розрахункових операцій шляхом збільшення варіативності при формуванні звіту та удосконалення алгоритмів шифрування інформації.

Призначення розробки – розробка та програмна реалізація програмного касового апарату для реєстрації розрахункових операцій.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Eckel B. Thinking in java. 4th ed. Upper saddle River, NJ : Prentice Hall, 2006.

2. Get started with Bootstrap. *Bootstrap · The most popular HTML, CSS, and JS library in the world.* URL: https://getbootstrap.com/docs/5.2/getting-started/introduction/ (date of access: 10.06.2022).

3. Порівняння симетричних з асиметричними криптосистемами. *Сучасні криптосистеми*. URL: https://sites.google.com/site/sucasnikriptosistemik/home/porivnanna-simetricnih-z-asimetricnimi-kriptosistemami (дата звернення: 10.06.2022).

4. Романюк О., Савчук Т. Організація баз даних і знань : навч. посіб. Вінниця : УНІВЕРСУМ-Вінниця, 2003. 123 с.

5. Технічні вимоги

Модель розробки – водоспадна; архітектурний шаблон проєктування – MVC; система управління базами даних – MySQL; мова запитів – SQL; вхідні дані – база даних із товарами їх кількість, замовлення та їх статус, робітники та їх посада; вихідні дані – звіт по розрахунковим послугам; середовище розробки – IntelliJ IDEA; мова програмування – Java.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б. Протокол перевірки

кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного забезпечення для касового апарату

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ліщинська Л.Б.

Оригінальність	84,9
Схожість	15,1

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Шинкарчук О. А.

Керівник роботи _____ Ліщинська Л.Б.

Додаток В. Лістинг додатка

```
public class DBConnectionPool {

    private DBConnectionPool(){
        //private constructor
    }

    private static DBConnectionPool instance = null;

    public static synchronized DBConnectionPool getInstance(){
        if (instance==null)
            instance = new DBConnectionPool();
        return instance;
    }

    public Connection getConnection(){
        Context ctx;
        Connection con = null;
        try {
            ctx = new InitialContext();
            DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/mydatabase");
            con = ds.getConnection();
        } catch (NamingException | SQLException e) {
            e.printStackTrace();
        }
        return con;
    }
}
package com.bd;

import com.bd.builder.EmpBuilderImpl;
import com.bd.entity.Commodity;
import com.bd.entity.Employe;
import java.sql.*;
import java.util.ArrayList;


public class DBManager {
    private static DBManager instance;

    public static synchronized DBManager getInstance() {
        if (instance == null) {
            instance = new DBManager();
        }
        return instance;
    }

    private DBManager() {
    }


    public boolean findEmploye(Employe employe)  {
        Connection con = null;
        PreparedStatement stmt=null;
        try {
            con=DBConnectionPool.getInstance().getConnection();
            stmt = con.prepareStatement(DBConstants.FIND_EMPLOYE);
            stmt.setString(1,employe.getName());
```

```
            stmt.setString(2,employe.getSurname());
            stmt.setString(3,employe.getPhone());
            stmt.setString(4,employe.getEmail());
            ResultSet rs=stmt.executeQuery();
            return rs.next();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            try {
                throw new DBException(throwables.getMessage(),throwables.getCause());
            } catch (DBException e) {
                e.printStackTrace();
            }
        }finally {
            close(con);
            close(stmt);
        }
        return false;
    }

    public boolean insertEmploye(Employe employe) {
        Connection con = null;
        PreparedStatement stmt=null;
        try {
            con=DBConnectionPool.getInstance().getConnection();
            con.setAutoCommit(false);
            stmt=con.prepareStatement(DBConstants.SIGNUP_EMPLOYE);

            stmt.setString(1,employe.getName());
            stmt.setString(2,employe.getSurname());
            stmt.setString(3,employe.getPhone());
            stmt.setString(4,employe.getEmail());
            stmt.setString(5,employe.getPwd());
            stmt.setString(6,employe.getPosition());

            stmt.executeUpdate();
            con.commit();
            return true;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            rollBack(con);
            try {
                throw new DBException(throwables.getMessage(),throwables.getCause());
            } catch (DBException e) {
                e.printStackTrace();

            }
        } finally {
            close(stmt);
            close(con);
        }
        return false;
    }

    public boolean authEmploye(Employe employe){
        Connection con = null;
        PreparedStatement stmt=null;
        try {
            con=DBConnectionPool.getInstance().getConnection();
            stmt = con.prepareStatement(DBConstants.SIGNIN_EMPLOYE);
            stmt.setString(1,employe.getEmail());
            stmt.setString(2,employe.getPwd());
```

```
            ResultSet rs=stmt.executeQuery();
            return rs.next();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            try {
                throw new DBException(throwables.getMessage(),throwables.getCause());
            } catch (DBException e) {
                e.printStackTrace();
            }
        }finally {
            close(con);
            close(stmt);
        }
        return false;
    }

    private void rollBack(Connection con) {
        try {
            con.rollback();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void close(AutoCloseable stmt) {
        if (stmt != null){
            try {
                stmt.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public Employe getAuthEloye(Employe employe) {
        Connection con = null;
        PreparedStatement stmt=null;
        try {
            con=DBConnectionPool.getInstance().getConnection();
            stmt = con.prepareStatement(DBConstants.SIGNIN_EMPLOYE);
            stmt.setString(1,employe.getEmail());
            stmt.setString(2,employe.getPwd());
            try(ResultSet rs=stmt.executeQuery()) {
                if (rs.next()) return new
EmpBuilderImpl().SetName(rs.getString("name")).SetSurname(rs.getString("surname")).SetPhon
e(rs.getString("phone")).SetEmail(rs.getString("email")).SetPwd(rs.getString("pwd")).SetPo
sition(rs.getString("position")).create();
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            try {
                throw new DBException(throwables.getMessage(),throwables.getCause());
            } catch (DBException e) {
                e.printStackTrace();
            }
        }finally {
            close(con);
            close(stmt);
        }
        return null;
    }
```

```
    public ArrayList<Commodity> findCommodity(String tag){

        ArrayList<Commodity>arr=new ArrayList<>();
        Connection con = null;
        PreparedStatement stmt=null;
        try {
            con=DBConnectionPool.getInstance().getConnection();
            stmt = con.prepareStatement(DBConstants.FIND_COMMODITY);
            stmt.setString(1,tag);
            ResultSet rs=stmt.executeQuery();
            while (rs.next())
            {
                arr.add(new Commodity(rs.getString("name"),rs.getString("manufacturer"),

rs.getString("description"),rs.getString("country"),rs.getDouble("price")));
            }

        } catch (SQLException throwables) {
            throwables.printStackTrace();
            try {
                throw new DBException(throwables.getMessage(),throwables.getCause());
            } catch (DBException e) {
                e.printStackTrace();
            }
        }finally {
            close(con);
            close(stmt);
        }
        return arr;
    }
}
package com.filter;

import java.io.IOException;
import java.util.Objects;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter("/*")
public class LoginFilter implements Filter {
    @Override
    public void init(FilterConfig config)
            throws ServletException {
        System.out.println("init filter");
        // If you have any <init-param> in web.xml, then you could get them
        // here by config.getInitParameter("name") and assign it as field.
    }

    @Override
```

```
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
            throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        HttpSession session = request.getSession();

//        String requestPath = request.getRequestURI();
        if (needsAuthentication(request.getServletPath()) &&
session.getAttribute("Employe") == null) { // change "user" for the session attribute you
have defined
                    response.sendRedirect("/Cashregister_war/login.jsp"); // No logged-in
user found, so redirect to login page.
        } else {
            chain.doFilter(req, res); // Logged-in user found, so just continue request.
        }
    }

    @Override
    public void destroy() {
        // If you have assigned any expensive resources as field of
        // this Filter class, then you could clean/close them here.
    }

    //basic validation of pages that do not require authentication
    private boolean needsAuthentication(String url) {
        String[] validNonAuthenticationUrls =
                { "/login.jsp","/index.jsp", "/registration","/LoginServlet" };
        for(String validUrl : validNonAuthenticationUrls) {
            if (url.endsWith(validUrl)) {
                return false;
            }
        }
        return true;
    }
}

package com.servlet;

        import com.bd.DBManager;
        import com.bd.entity.Commodity;

        import javax.servlet.ServletException;
        import javax.servlet.annotation.WebServlet;
        import javax.servlet.http.HttpServlet;
        import javax.servlet.http.HttpServletRequest;
        import javax.servlet.http.HttpServletResponse;
        import javax.servlet.http.HttpSession;
        import java.io.IOException;
        import java.sql.*;
        import java.util.ArrayList;

@WebServlet("/CommodityServlet")
public class CommodityServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        HttpSession session = req.getSession();

        resp.setContentType("text/html;charset=UTF-8");
        req.setCharacterEncoding("UTF-8");
```

```
        System.out.println("1");
        DBManager dbManager=DBManager.getInstance();
        if (dbManager.findCommodity(req.getParameter("search")).isEmpty()) {
            System.out.println("2");
            req.setAttribute("Commodity",
dbManager.findCommodity(req.getParameter("search")));
            getServletContext().getRequestDispatcher("/commodity.jsp").forward(req, resp);
        }else {
            System.out.println("3");
            req.setAttribute("Commodity",
dbManager.findCommodity(req.getParameter("search")));
            getServletContext().getRequestDispatcher("/commodity.jsp").forward(req, resp);
        }
    }
}
package com.servlet;

import com.bd.DBManager;
import com.bd.builder.EmpBuilderImpl;
import com.bd.entity.Employe;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.sql.*;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html;charset=UTF-8");
        req.setCharacterEncoding("UTF-8");
        HttpSession session = req.getSession();
        Integer count = (Integer) session.getAttribute("count");
        if (count == null) {
            session.setAttribute("count", 1);
            count = 1;
        } else {
            session.setAttribute("count", count + 1);
        }
        DBManager dbManager=DBManager.getInstance();
        if (dbManager.authEmploye(new
EmpBuilderImpl().SetEmail(req.getParameter("login")).SetPwd(req.getParameter("pwd")).creat
e())){
            req.getSession().setAttribute("Employe",dbManager.getAuthEloye(new
EmpBuilderImpl().SetEmail(req.getParameter("login")).SetPwd(req.getParameter("pwd")).creat
e()));
            resp.sendRedirect("index.jsp");
        }else if (count<3){
            resp.sendRedirect("login.jsp");
        }else  resp.sendRedirect("identify");

    }
}
```

```java
package com.servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        HttpSession session = req.getSession();
        session.removeAttribute("Employe");
        req.getSession().invalidate();
        resp.sendRedirect("login.jsp");
    }
}
package com.servlet;


import com.bd.DBManager;
import com.bd.builder.EmpBuilderImpl;
import com.bd.entity.Employe;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/registration")
public class RegistrationServlet extends HttpServlet {
    @Override

    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        resp.setContentType("text/html;charset=UTF-8");
        req.setCharacterEncoding("UTF-8");

        Employe employe=new
EmpBuilderImpl().SetName(req.getParameter("name")).SetSurname(req.getParameter("surname"))
                .SetPhone(req.getParameter("phone")).SetEmail(req.getParameter("email"))

.SetPwd(req.getParameter("pwd1")).SetPosition(req.getParameter("role")).create();

        DBManager dbManager=DBManager.getInstance();

        if (dbManager.findEmploye(employe)) {
            System.out.println(dbManager.findEmploye(employe));
            resp.sendRedirect("login.jsp");
        }
        else if (dbManager.insertEmploye(employe)){
            req.getSession().setAttribute("Employe",employe);
            resp.sendRedirect("index.jsp");
        }else {
            resp.sendRedirect("login.jsp");
```

```
        }
    }
}
package com.bd.builder;

import com.bd.entity.Employe;

public class EmpBuilderImpl implements EmpBuilder {
    Employe employe=new Employe();
    @Override
    public EmpBuilder SetId(int id) {
        employe.setId(id);
        return this;
    }

    @Override
    public EmpBuilder SetName(String name) {
        employe.setName(name);
        return this;
    }

    @Override
    public EmpBuilder SetSurname(String surname) {
        employe.setSurname(surname);
        return this;
    }

    @Override
    public EmpBuilder SetPhone(String phone) {
        employe.setPhone(phone);
        return this;
    }

    @Override
    public EmpBuilder SetEmail(String email) {
        employe.setEmail(email);
        return this;
    }

    @Override
    public EmpBuilder SetPwd(String pwd) {
        employe.setPwd(pwd);
        return this;
    }

    @Override
    public EmpBuilder SetPosition(String position) {
        employe.setPosition(position);
        return this;
    }

    @Override
    public Employe create() {
        return employe;
    }
}


package com.bd.builder;

import com.bd.entity.Employe;
```

```
public interface EmpBuilder {
    EmpBuilder SetId(int id);
    EmpBuilder SetName(String name);
    EmpBuilder SetSurname(String surname);
    EmpBuilder SetPhone(String phone);
    EmpBuilder SetEmail(String email);
    EmpBuilder SetPwd(String pwd);
    EmpBuilder SetPosition(String position);
    Employe create();
}
package com.bd.entity;

public class Commodity {
    private int id;
    private String name;
    private String manufacturer;
    private String description;
    private String country;
    private double price;
    private String tags;
    private int amount;
    private int employe_id;

    public Commodity() {
    }

    public Commodity(String name, String manufacturer, String description, String country,
double price) {

        this.name = name;
        this.manufacturer = manufacturer;
        this.description = description;
        this.country = country;
        this.price = price;
    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }


    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }


    public String getManufacturer() {
        return manufacturer;
    }
    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }
```

```
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }


    public String getCountry() {
        return country;
    }
    public void setCountry(String country) {
        this.country = country;
    }


    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }


    public String getTags() {
        return tags;
    }
    public void setTags(String tags) {
        this.tags = tags;
    }


    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }


    public int getEmploye_id() {
        return employe_id;
    }
    public void setEmploye_id(int employe_id) {
        this.employe_id = employe_id;
    }

}
package com.bd.entity;

public class Employe {

    private int id;
    private String name;
    private String surname;
    private String phone;
    private String email;
    private String position;
    private String pwd;
    private String country;
    private String state;
```

```
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}


public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}


public String getSurname() {
    return surname;
}
public void setSurname(String surname) {
    this.surname = surname;
}


public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}


public String getPhone() {
    return phone;
}
public void setPhone(String phone) {
    this.phone = phone;
}


public String getPosition() {
    return position;
}
public void setPosition(String position) {
    this.position = position;
}


public String getPwd() {
    return pwd;
}
public void setPwd(String pwd) {
    this.pwd = pwd;
}


public String getCountry() {
    return country;
}
public void setCountry(String country) {
```

```
        this.country = country;
    }


    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
}
```

```
@WebServlet("/ReportControler")
public class ReportControler extends HttpServlet {

    private ReportRate reportRate;

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        reportRate = (ReportRate) config.getServletContext().getAttribute("reportRate");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        long orderId = Long.parseLong(req.getParameter("orderId"));
        Report report = new Report();
        report.setId(orderId);
        report.setComment(req.getParameter("comment"));
        report.setRate(ReportRate.valueOf(req.getParameter("rate")));
        HttpSession session = req.getSession();
        session.setAttribute("infoMessage", reportRate.saveReport(report));
        resp.sendRedirect("order?orderId=" + orderId);
    }
}
```

```
public class OrderServiceImpl implements OrderService {

    private static final Logger LOGGER = Logger.getLogger(OrderServiceImpl.class);
    private OrderDao orderDao;
    private UserService userService;
    private Validator validator;
    private OrdersExelWriter ordersExelWriter;

    public OrderServiceImpl(OrderDao orderDao, UserService userService, Validator
validator) {
        this.orderDao = orderDao;
        this.userService = userService;
        this.validator = validator;
        ordersExelWriter = new OrdersExelWriter();
    }

    @Override
    public Order getOrderById(long orderId) {
        if (orderId > 0) {
            return orderDao.getOrderById(orderId);
        }
        LOGGER.error("Service Exception in getOrderById, wrong orderId " + orderId);
```

```
        throw new ServiceException();
    }

    @Override
    public List<Order> getAllCustomerOrders(long userId) {
        if (userId > 0) {
            return orderDao.getAllCustomerOrders(userId);
        }
        LOGGER.error("Service Exception in getAllCustomerOrders, wrong userId " + userId);
        throw new ServiceException();
    }

    @Override
    public InfoMessage insertOrder(Order order) {
        if (validator.validateOrder(order)) {
            orderDao.insertOrder(order);
            return InfoMessage.CREATING_ORDER_SUCCES;
        }
        LOGGER.error("Service Exception in Order " + order);
        return InfoMessage.WRONG_FIELDS;
    }

    @Override
    public List<Order> getAllOrders() {
        return orderDao.getAllOrders();
    }

    @Override
    public InfoMessage updateOrder(Order order) {
        if (validator.validateOrder(order) && order.updateOrder(order)) {
            return InfoMessage.UPDATING_ORDER_SUCCESS;
        }
        LOGGER.error("Service Exception in updateOrder " + order);
        return InfoMessage.WRONG_FIELDS;
    }


    @Override
    public List<Order> findOrders(HttpServletRequest req) {
        OrderPaginationObject pagObject = getPgObjectFromRequest(req);
        List<Order> orders = orderDao.getOrderRows(pagObject);
        setPaginationAttributes(req, pagObject);
        return orders;
    }

    private OrderPaginationObject getPgObjectFromRequest(HttpServletRequest req) {

        OrderPaginationObject pagObj;
        try {
            int currentPage = Integer.parseInt(req.getParameter("currentPage"));
            int recordsPerPage = Integer.parseInt(req.getParameter("recordsPerPage"));
            String filter = req.getParameter("filter");
            String filterParam = req.getParameter("filterParam");
            boolean reverse = Boolean.parseBoolean(req.getParameter("reverse"));
            String orderBy = req.getParameter("orderBy");

            pagObj = new OrderPaginationObject.Builder(currentPage,
recordsPerPage).addFilter(filter, filterParam)
                    .addSorting(orderBy, reverse).addLimit().build();

            orderDao.setRowsAmount(pagObj);
```

```
            int nOfPages = pagObj.getAmountOfRows() / recordsPerPage;
            if (pagObj.getAmountOfRows() % recordsPerPage > 0) {
                nOfPages++;
            }
            pagObj.setNumberOfPages(nOfPages);
        } catch (IllegalArgumentException | NullPointerException exception) {
            LOGGER.error("Exception in getOrderPaginationParams. Wrong Arguments",
exception);
            throw new ServiceException();
        }
        return pagObj;
    }

    private HttpServletRequest setPaginationAttributes(HttpServletRequest req,
OrderPaginationObject pagObject) {
        if (pagObject.getFilter() != null) {
            req.setAttribute("filter", pagObject.getFilter());
            if (pagObject.getFilterParam() != null) {
                req.setAttribute("filterParam", pagObject.getFilterParam());
            }
        }
        if (PaginationField.STATUS.equals(pagObject.getFilter())) {
            Set orderStatuses = EnumSet.allOf(OrderStatus.class);
            req.setAttribute("statuses", orderStatuses);
        }
        if (PaginationField.MASTER.equals(pagObject.getFilter())) {
            List<User> listOfMasters = userService.getAllUsersByRole(UserRole.MASTER);
            req.setAttribute("masters", listOfMasters);
        }
        req.setAttribute("reverse", pagObject.isReverse());
        req.setAttribute("noOfPages", pagObject.getNumberOfPages());
        req.setAttribute("currentPage", pagObject.getCurrentPage());
        req.setAttribute("recordsPerPage", pagObject.getRecordsPerPage());
        req.setAttribute("orderBy", pagObject.getOrderBy());
        return req;
    }

    @Override
    public void updateNewOrderCount(HttpServletRequest req) {
        ServletContext context = req.getServletContext();
        long countOfNewOrders = orderDao.getNewOrdersAmount();
        context.setAttribute("countOfNewOrders", countOfNewOrders);
    }

    @Override
    public void uploadOrdersToExel(String path) {
        try {
            List<Order> orderList = orderDao.getAllOrders();
            ordersExelWriter.writeIntoExcel(path, orderList);
        } catch (IOException e) {
            LOGGER.error("Exception in uploadOrdersToExel. Can't upload orders to file",
e);
            throw new ServiceException();
        }
    }
}
public class Validator {

    private static final String EMAIL_PATTERN = "[a-zA-Z]+@{1}[a-zA-Z]+\\.{1}[a-zA-
Z]{2,4}";
    private static final String PASSWORD_PATTERN = "^(?=.*\\d)(?=.*[a-z])(?=.)(?=.*[a-zA-
```

```
Z]).{5,}$";
    private static final String PHONE_PATTERN = "(^[0-9]{6,12})[^a-zA-Za-яА-Я]|^$";

    public boolean validateEmail(String email) {
        if (email == null || email.length() < 5 || email.length() > 25) {
            return false;
        }
        Pattern patternMail = Pattern.compile(EMAIL_PATTERN);
        Matcher matcher = patternMail.matcher(email);
        return matcher.find();
    }

    public boolean validatePassword(String password) {
        if (password == null) {
            return false;
        }
        if (password.length() < 5 || password.length() > 128) {
            return false;
        }
        Pattern patternMail = Pattern.compile(PASSWORD_PATTERN);
        Matcher matcher = patternMail.matcher(password);
        return matcher.find();
    }

    public boolean validatePhone(String phone) {
        if (phone == null) {
            return false;
        }
        if (phone.length() < 6 || phone.length() > 12) {
            return false;
        }
        Pattern patternMail = Pattern.compile(PHONE_PATTERN);
        Matcher matcher = patternMail.matcher(phone);
        return matcher.find();
    }

    public boolean validateUser(User user) {
        if (user == null) {
            return false;
        }
        if (user.getName() == null || user.getSurName() == null || user.getEmail() == null
                || user.getPassword() == null || user.getPhone() == null ||
user.getLocale() == null) {
            return false;
        }
        if (user.getName().length() < 2 || user.getName().length() > 30) {
            return false;
        }
        if (user.getSurName().length() < 2 || user.getSurName().length() > 35) {
            return false;
        }
        return validateEmail(user.getEmail()) && validatePassword(user.getPassword()) &&
validatePhone(user.getPhone())
                && user.getName().length() >= 2 && user.getSurName().length() >= 2;
    }

    public boolean validateOrder(Order order) {
        if (order == null) {
            return false;
        }
        if (order.getDevice() == null || order.getDevice().length() < 5
```

```
                || order.getDevice().length() > 60) {
            return false;
        }
        if (order.getComment() == null || order.getComment().length() < 10
                || order.getComment().length() > 200) {
            return false;
        }
        return true;
    }

    public boolean validateFeedback(FeedBack feedBack) {
        if (feedBack == null) {
            return false;
        }
        if (feedBack.getComment() == null || feedBack.getComment().length() > 300) {
            return false;
        }
        return feedBack.getRate() != null;
    }

    public boolean validateService(Service service) {
        if (service == null) {
            return false;
        }
        if (service.getNameEn() == null || service.getNameRu() == null) {
            return false;
        }
        return service.getNameEn().length() > 4 && service.getNameRu().length() > 4;
    }
}
public class UserServiceImpl implements UserService {


    private static final Logger LOGGER = Logger.getLogger(UserServiceImpl.class);
    private UserDao userDao;
    private Validator validator;

    public UserServiceImpl(UserDao userDao, Validator validator) {
        this.userDao = userDao;
        this.validator = validator;
    }

    @Override
    public User getUserByEmailPass(String userEmail, String userPass){
        if (!validator.validateEmail(userEmail) || !validator.validatePassword(userPass))
{
            throw new AuthenticationException();
        }
        User user = userDao.getUserByEmail(userEmail);
        if (user != null && checkPass(user, userPass)) {
            return user;
        } else {
            LOGGER.info("Wrong authorization " + userEmail);
            throw new AuthenticationException();
        }
    }

    @Override
    public User getUserById(long id) {
        if (id > 0) {
            return userDao.getUserById(id);
```

```
        } else {
            LOGGER.info("Service exception in get user by Id  " + id);
            throw new ServiceException();
        }
    }

    @Override
    public User saveUser(User user){
        if (!validator.validateUser(user)) {
            LOGGER.info("Wrong registration " + user.getEmail());
            throw new RegistrationException();
        }
        if (userDao.getUserByEmail(user.getEmail()) != null) {
            LOGGER.info("Wrong registration " + user.getEmail());
            throw new RegistrationException();
        }
        user.setPassword(getSecurePassword(user.getPassword(),
getSalt(user.getPassword())));
        return userDao.saveUser(user);
    }

    @Override
    public User updateUser(User user){
        if (!validator.validateUser(user)) {
            LOGGER.info("Can't update account");
            throw new RegistrationException();
        }
        User currentUser = userDao.getUserById(user.getId());
        if (!currentUser.getPassword().equals(user.getPassword())) {
            user.setPassword(getSecurePassword(user.getPassword(),
getSalt(user.getPassword())));
        }
        return userDao.saveUser(user);
    }

    @Override
    public List<User> findUsers(long currentPage, long recordsPerPage) {
        long startPosition = currentPage * recordsPerPage - recordsPerPage;
        return userDao.findUsers(startPosition, recordsPerPage);
    }

    @Override
    public List<User> getAllUsersByRole(UserRole role) {
        return userDao.getAllUsersByRole(role);
    }

    @Override
    public long getAmountOfUsers() {
        return userDao.amountOfUsers();
    }

    private boolean checkPass(User user, String userPass) {
        if (user != null && userPass != null) {
            String encodedPass = getSecurePassword(userPass, getSalt(userPass));
            return user.getPassword().equals(encodedPass);
        } else {
            LOGGER.info("Service exception in checkPass. Input user = " + user + " pass "
+ userPass);
            throw new AuthenticationException();
        }
    }
```

```
    private String getSecurePassword(String passwordToHash, String salt) {
        String generatedPassword;
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-512");
            md.update(salt.getBytes(StandardCharsets.UTF_8));
            byte[] bytes = md.digest(passwordToHash.getBytes(StandardCharsets.UTF_8));
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < bytes.length; i++) {
                sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
            }
            generatedPassword = sb.toString();
        } catch (NoSuchAlgorithmException e) {
            LOGGER.info("Service exception in getSecurePassword");
            throw new ServiceException();
        }
        return generatedPassword;
    }

    private String getSalt(String pass) {
        return Base64.encodeBase64String(pass.getBytes());
    }
}
```

Додаток Г. Графічна частина до бакалаврської дипломної роботи

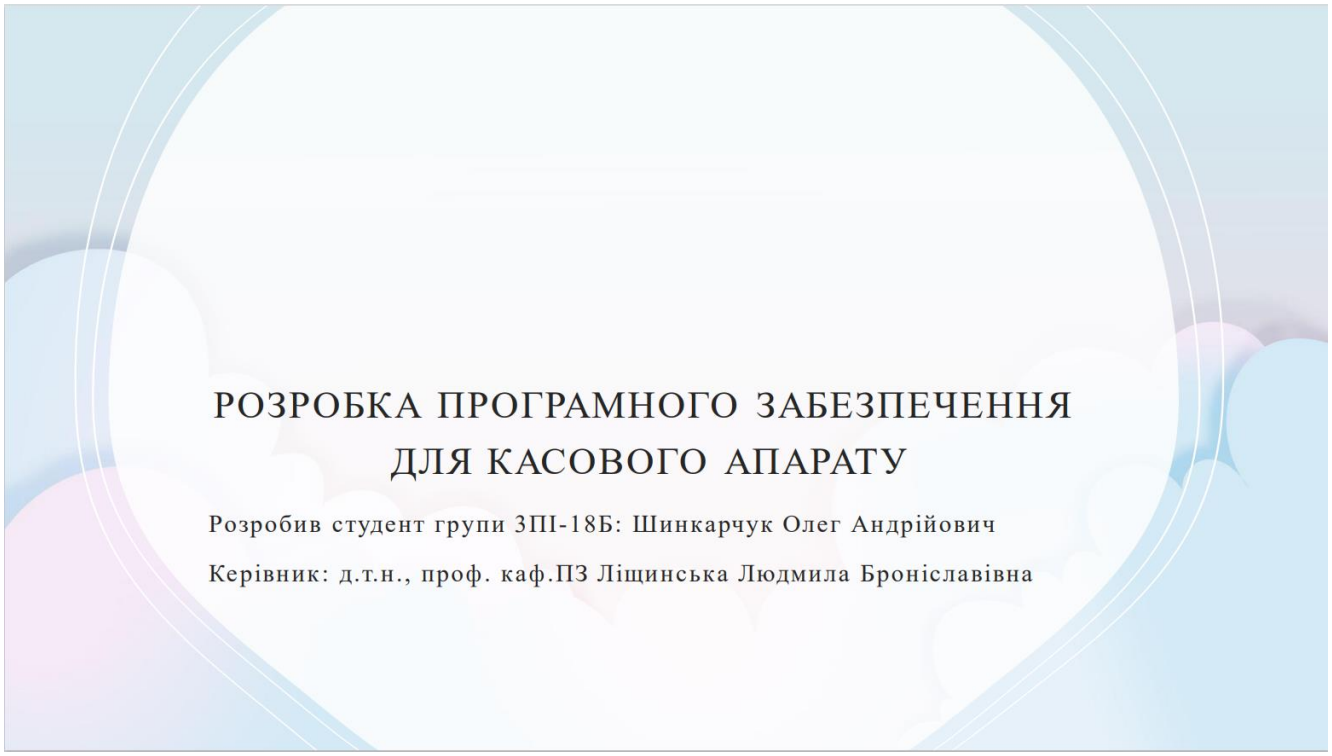


Рисунок Г.1 – Титульний слайд презентації

Рисунок Г.2 – Актуальність теми

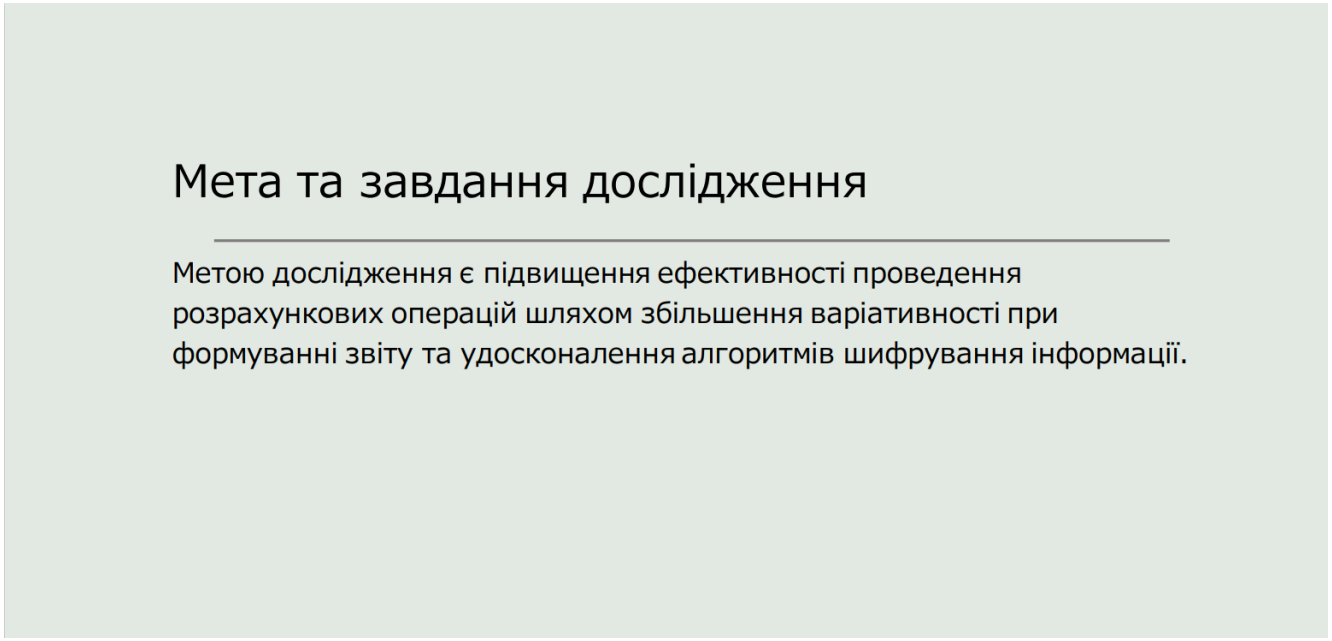


Рисунок Г.3 – Мета та завдання дослідження

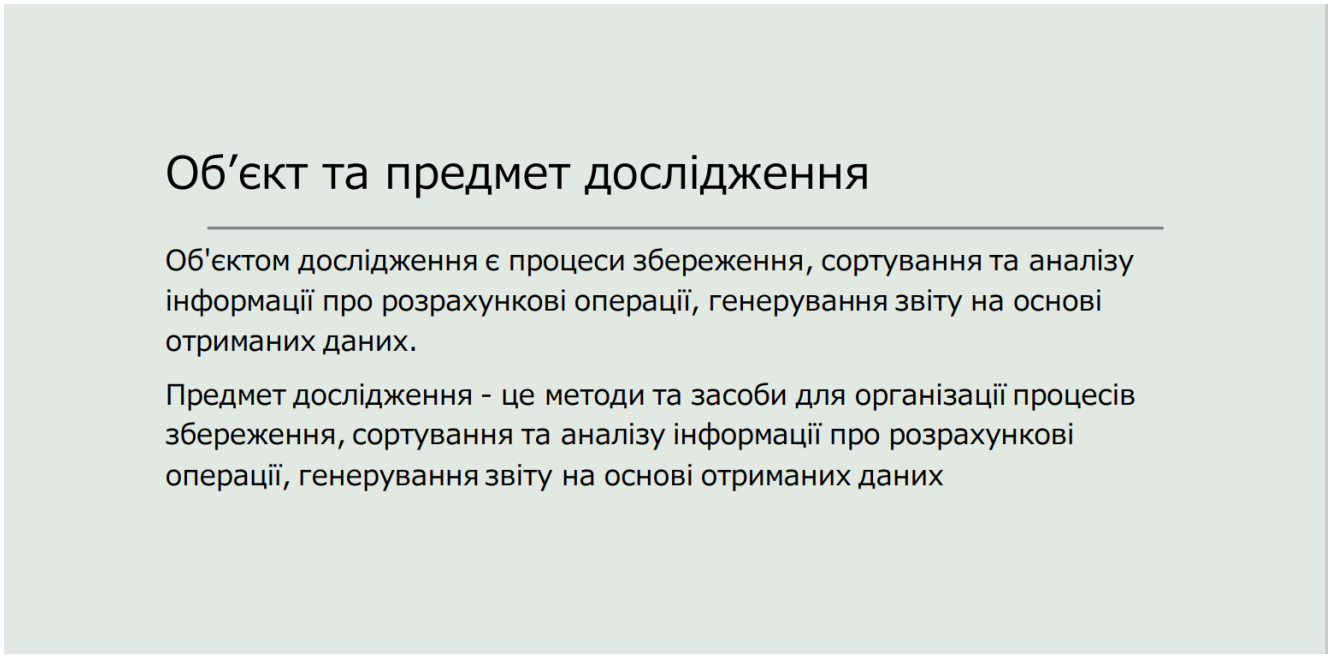
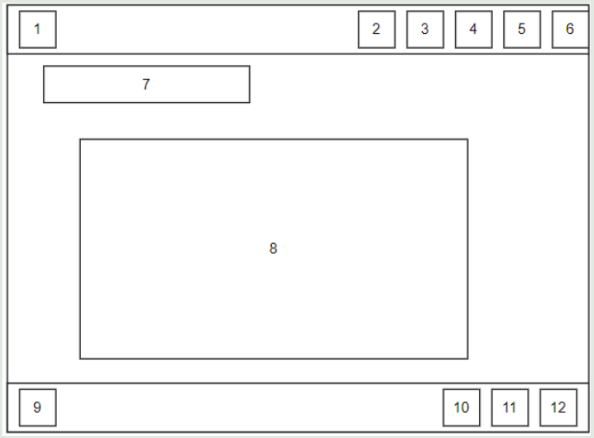


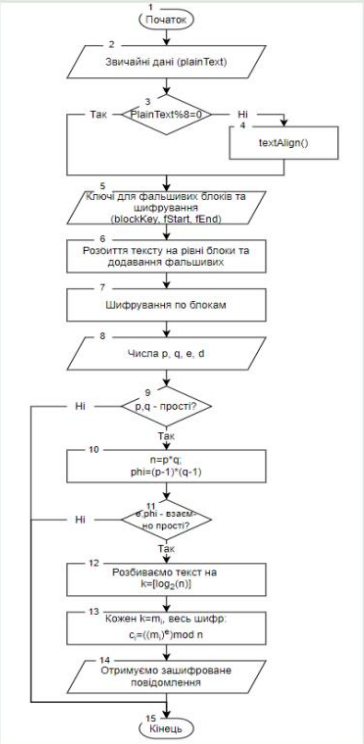
Рисунок Г.4 – Об’єкт та предмет дослідження

СТРУКТУРА ГРАФІЧНОГО ІНТЕРФЕЙСУ ГОЛОВНОЇ СТОРІНКИ ВЕБДОДАТКА



1. Логотип застосунку.
2. Перехід на головну сторінку (неактивний на головній сторінці).
3. Перехід на сторінку замовлень.
4. Перехід на сторінку товарів та послуг.
5. Випадаюче меню для переходу на сторінку працівника або логауту.
6. Випадаюче меню для зміни мови інтерфейсу застосунка.
7. Рядок пошуку.
8. Результуюче вікно пошуку.
9. Авторські дані.
10. FAQ.
11. Перехід на головну сторінку (неактивний на головній сторінці).
12. Інформації про застосунок.

Рисунок Г.5 – Структура графічного інтерфейсу



Блок-схема алгоритму шифрування даних

Шифрування даних відбувається в 3 етапи:
1) 1-4 Вирівнювання даних збільшує їх довжину для отримання кратної довжини розміру даних
2) 5-7 Шифрування блоками проведення - проведення операції виключної диз'юнкції.
3) 8-14 Асиметричне шифрування даних.

Рисунок Г.6 – Блок-схема алгоритму шифрування даних

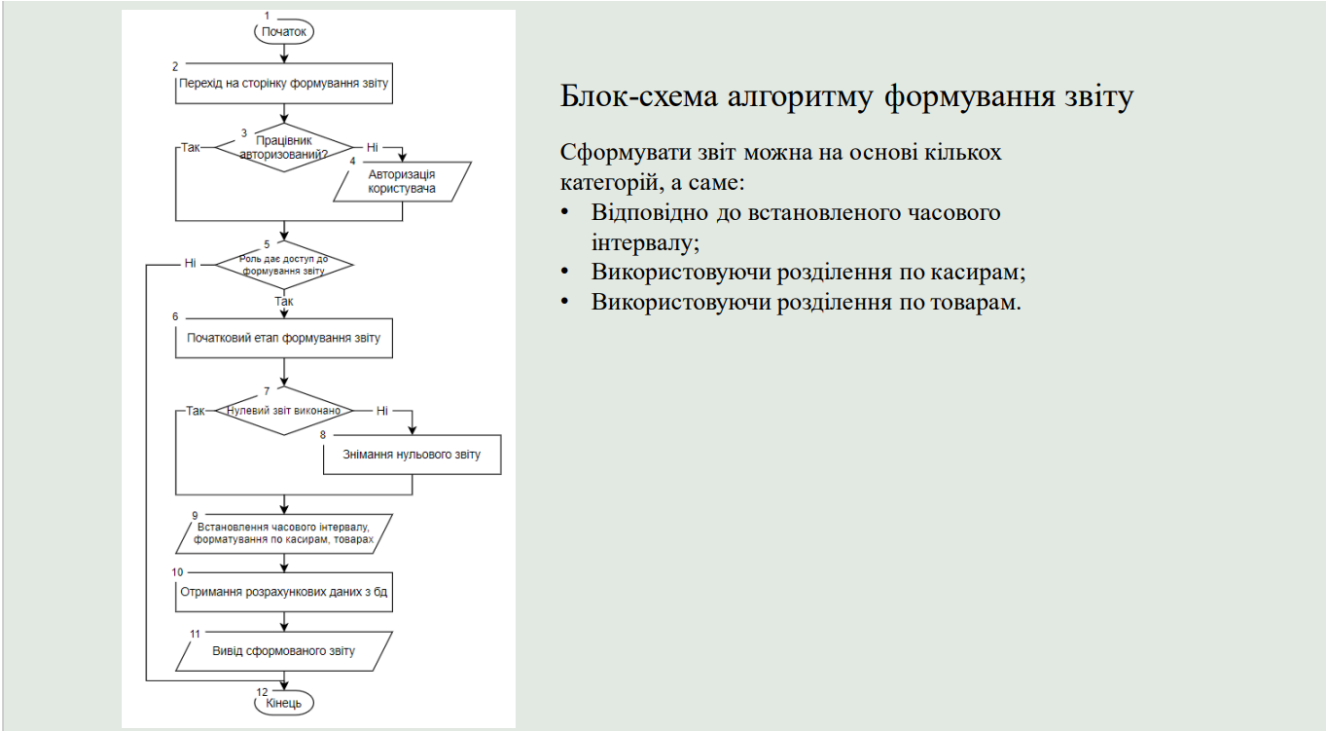


Рисунок Г.7 – Блок-схема алгоритму формування звіту

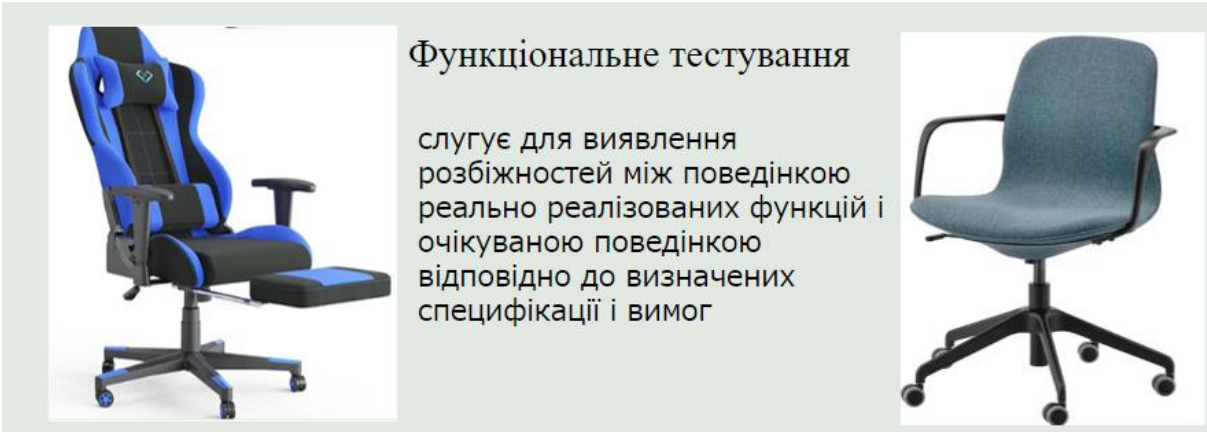


Рисунок Г.8 – Тестування додатку

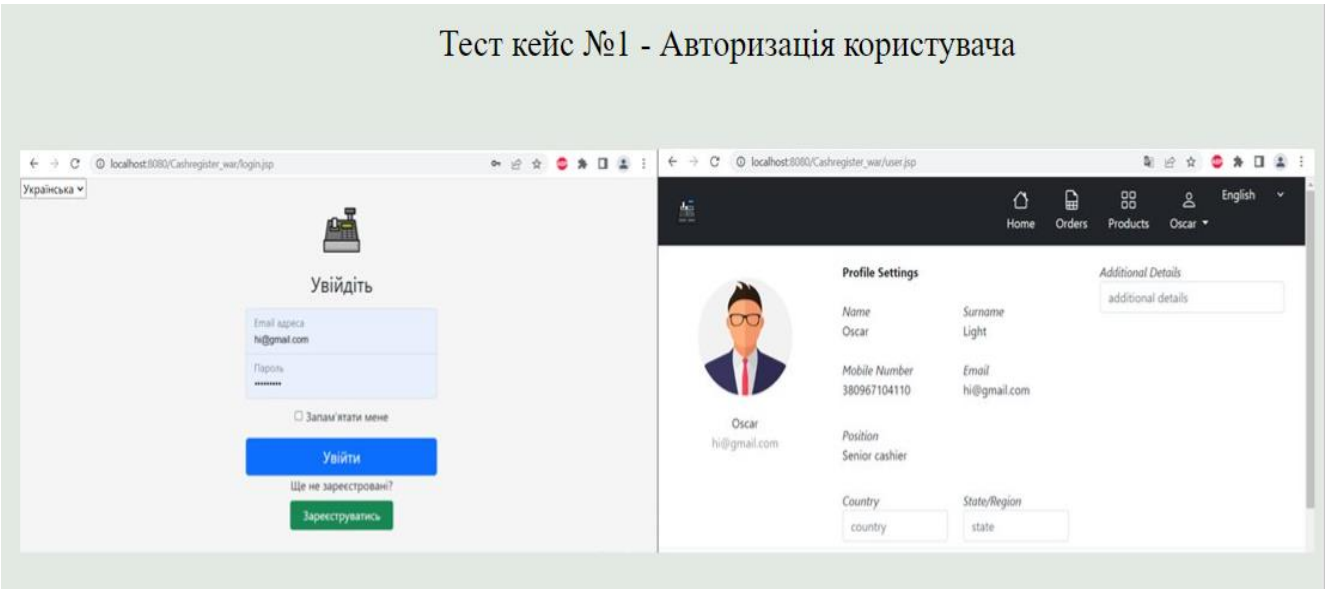


Рисунок Г.9 – Тест кейс номер 1

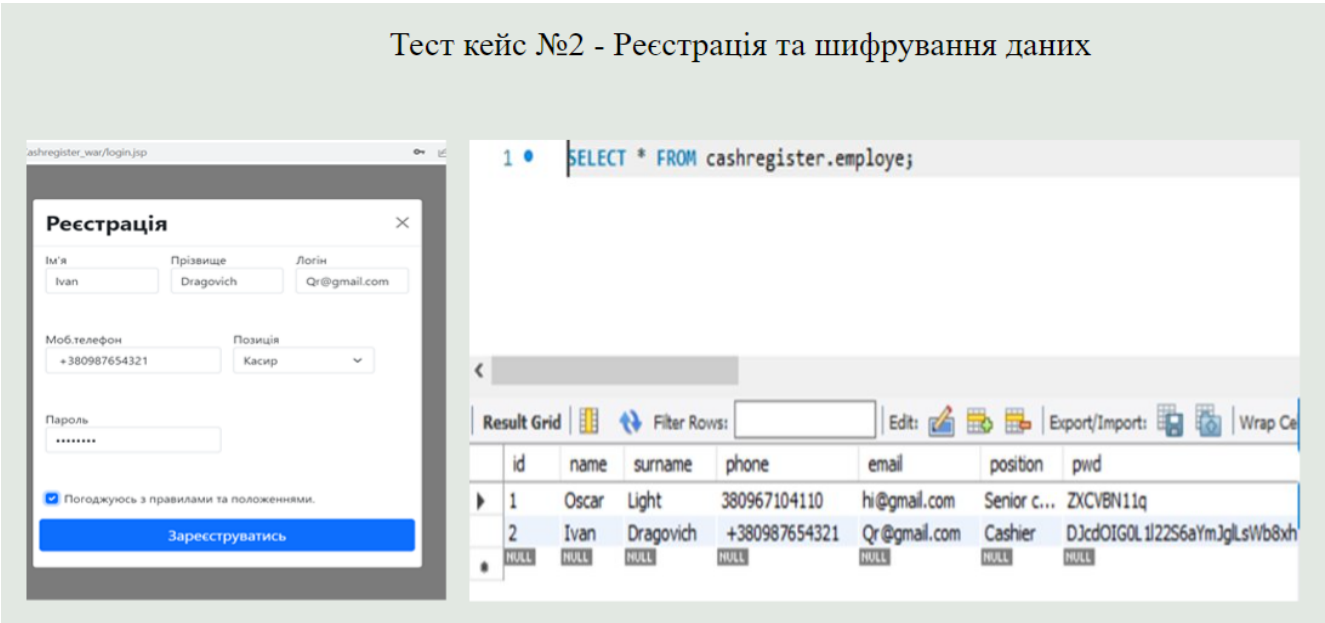


Рисунок Г.10 – Тест кейс номер 2

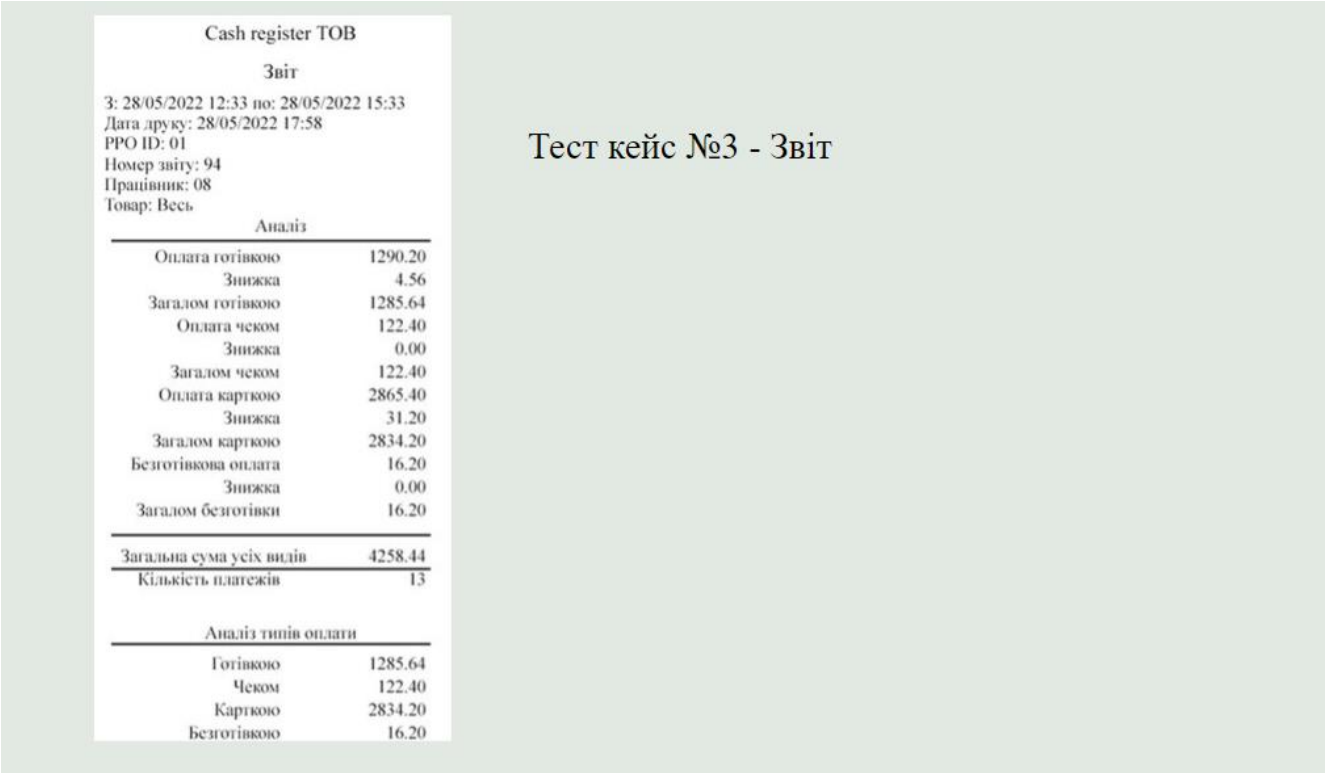


Рисунок Г.11 – Тест кейс номер 3

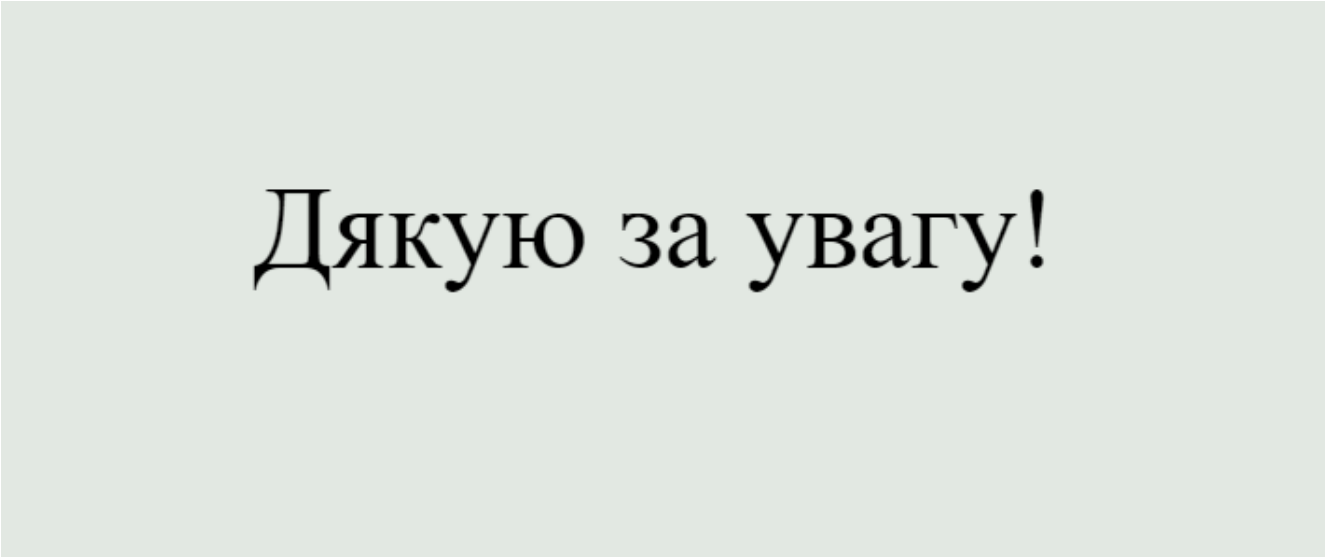


Рисунок Г.12 – Фінальний слайд презентації