Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«Інформаційна система підтримки підприємств малого бізнесу у сфері послуг»

08-23.МКР.001.00.000 ПЗ

Виконала: студентка 2 курсу, групи 1КІ-20 м
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки)
_____ Боднар К. О.
(прізвище та ініціали)

Керівник: к.т.н., професор каф. ОТ
_____ Захарченко С. М.
(прізвище та ініціали)
«____» _____2021 р.

Опонент: к.т.н., доцент каф. МБІС
_____ Карпінець В. В.
(прізвище та ініціали)
«____» _____2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«____» _____2021 року

Вінниця ВНТУ 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень — магістр
Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
Азаров О. Д.
«____» _____2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____Боднар Карині Олександрівні_____
(прізвище, ім'я, по батькові)

1 Тема роботи: Інформаційна система підтримки підприємств малого бізнесу у сфері послуг

керівник роботи: к.т.н., професор кафедри ОТ Захарченко С. М. затверджені наказом Вінницького національного технічного університету від 24.09.2021 року № 277.

2 Строк подання студентом роботи: 10.12.2021

3 Вихідні дані до роботи: перелік функціональних можливостей, клієнт-серверна технологія, інтуїтивний інтерфейс, сумісність з найбільш популярними браузерами, можливість розгортання в хмарному середовищі.

4 Зміст розрахунково-пояснювальної записки: огляд характеристик інформаційних систем, методів класифікації та існуючих систем підтримки, аналіз технологій та розробка інформаційної системи, розробка програмного забезпечення, тестування розробленого програмного забезпечення.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
схема функціональних можливостей інформаційної системи, алгоритм роботи програми, графічний інтерфейс програми, таблиця порівняльного аналізу фреймворку Spring, лістинг програми, протокол перевірки на антиплагіат.

6 Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3,4	Захарченко С. М., к.т.н., доцент каф. ОТ		
5	Кавецький В. В., к.е.н., доцент каф. ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

Етап	Назва етапу	Початок	Кінець	Очікувані результати
1	Інформаційний пошук та огляд літературних джерел.	01.07.21	01.09.21	Розділи 1, 2, 3 та 4.
2	Дослідження сучасних інструментів, які дозволили б реалізувати на практиці дану інформаційну систему. Вивчення Spring фреймворка.	01.09.21	05.10.21	Чернетки матеріалів.
3	Розробка програмних засобів.	05.10.21	01.11.21	Програмний продукт
4	Підготовка матеріалів пояснювальної записки.	01.11.20	8.12.21	Пояснювальна записка.

Студентка _____ Боднар К. О.
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Захарченко С. М.
(підпис) (прізвище та ініціали)

Консультант з економічної частини _____ Кавецький В. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Боднар К.О. Інформаційна система підтримки підприємств малого бізнесу у сфері послуг. Магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2021. 89 с.

У магістерській кваліфікаційній роботі було розглянуто проблему моніторингу підприємств малого бізнесу у сфері послуг. А саме, був проведений огляд основних характеристик, які мають бути включені для проведення аудиту клієнтів та існуючих на сьогоднішній день систем підтримки підприємств.

Детально розглядаються технології Spring, Spring Security, Thymeleaf, Maven та Postgresql, які будуть використовуватись для роботи програми. Було розроблено інформаційну систему для управління клієнтською системою з використанням засобів мови Java.

Проведено тестування продукту на коректність опрацювання клієнтських даних та на наявність помилок. Розроблено інструкцію користувача, що дозволить йому отримати інформацію про основи використання сервісу та інструкцію для використання даного програмного продукту.

Також було проаналізовано економічну доцільність досліджуваної розробки при впроваджені її у виробництво та встановлено, що інформаційна система має хороші шанси на комерційний успіх.

Ключові слова: інформаційна система, моніторинг клієнтів, статистика, аудит, нотифікація

ANNOTATION

Bodnar K.O. Information system to support small businesses in the field of services. Master's thesis in specialty 123 - computer engineering, educational program - computer engineering. Vinnytsia: VNTU, 2021. 89 p.

In the master thesis, the problem of monitoring small businesses in the field of services was considered. This is a review of the main characteristics that were included for the audit of customers and to date, the support system of enterprises.

More details, Spring, Spring Security, Thymeleaf, Maven and Postgresql technologies that will be used to run the programs. A software product for client system management using Java language tools has been developed.

The product was tested for correctness of customer data and errors. Developed user instructions that allow him to obtain information about the basics of using the service and instructions for using this software product.

It was also analyzed the economic feasibility of the studied development in its implementation in production and found that the information system has a good chance of commercial success.

Keywords: information system, customer monitoring, statistics, audit, notification

ЗМІСТ

ВСТУП

У сьогоднішньому світі існує величезна кількість різних малих бізнесів. Головним фактором успіху будь-якого з них є грамотна автоматизація всіх його бізнес-процесів. Управління сучасним власним бізнесом в умовах ринкової економіки є складним процесом, що включає вибір та реалізацію певного набору управлінських впливів на поточні тимчасові відрізки для вирішення стратегічного завдання забезпечення його сталого фінансового та соціально-економічного розвитку. Інформаційні технології, що досягли в останнє десятиліття нового якісного рівня, значною мірою розширюють можливості ефективного управління, оскільки надають у розпорядження менеджерів, фінансистів, маркетологів, керівників виробництва всіх рангів новітні методи обробки та аналізу економічної інформації, яка потрібна на прийняття рішень.

Інформаційні системи розширюють професійні можливості фахівців та дозволяють здійснювати діяльність господарчого суб'єкта більш раціонально, цілеспрямовано та економно, більш ефективно. Сучасний ринок створює ситуацію, за якої необхідно постійно підвищувати ефективність виробництва, максимально швидко реагувати на будь-які зміни, покращувати якість обслуговування клієнтів, знизити втрати, будувати ефективні прогнози на майбутнє. Для цього керівник підприємства повинен мати достовірну інформацію щодо повного аналізу діяльності бізнесу відносно реального часу. Будь-то кількість клієнтів, облік витрат на матеріали, що необхідні для задоволення потреб клієнтів, зручний інтерфейс розкладу, статистика та аналітика необхідних показників оцінювання роботи, на основі яких можна приймати рішення стосовно реклами та фінансовий облік. Тому виникають завдання автоматизації. Відрізнятися можуть лише шляхи та засоби досягнення кінцевого результату.

Ось чому можна вважати, дану тему дослідження достатньо актуальною на сьогоднішній день.

Метою дослідження є вдосконалення інформаційної системи обліку клієнтів та фінансів, що дозволяє реєструвати та супроводжувати клієнтів даного малого бізнесу шляхом розширення функціональних можливостей.

Для досягнення поставленої мети необхідно розв’язати такі **задачі**:

— провести аналіз існуючих на ринку систем підтримки підприємств;

— дослідити сучасні інструменти які дозволили б реалізувати на практиці дану інформаційну систему;

— розробити програмне забезпечення;

— провести тестування розробленого програмного забезпечення.

Об’єктом дослідження є процес розробки інформаційної системи та її розгортання у хмарному середовищі.

Предметом дослідження є інформаційна система підтримки підприємств малого бізнесу у сфері послуг.

Методи дослідження що використовувались у магістерській роботі: методи та технології проектування кросплатформленого програмного забезпечення, методи побудови реляційних баз даних. Також, при розробці програмного продукту, використано методи об’єктно-орієнтованого програмування для побудови моделей даних.

Наукова новизна отриманих результатів магістерської роботи полягає в розширенні функціональних можливостей CRM за рахунок додавання функціоналу відстеження доходу, витрат, цілей клієнтів, статистичного аналізу, сповіщення клієнтів, що дозволило підвищити ступінь користності системи.

Практичне значення отриманих результатів магістерської роботи полягає в розробці інформаційної системи підтримки малого бізнесу. Розробка призначена для спрощення процесу контролю бюджету, фіксації витрат, доходів, обліку клієнтів, повідомлення їх про відвідування тощо.

Апробація результатів магістерської роботи полягає в наступному. Основні результати наукових досліджень за темою роботи доповідалися та обговорювалися на щорічній конференції професорсько-викладацького складу, співробітників та студентів Вінницького національного технічного університету (Вінниця, 2021). За результатами проведених досліджень було опубліковано одну роботу [1].

1 ОГЛЯД ХАРАКТЕРИСТИК ІНФОРМАЦІЙНИХ СИСТЕМ, МЕТОДІВ КЛАСИФІКАЦІЇ ТА ІСНУЮЧИХ СИСТЕМ ПІДТРИМКИ

1.1 Роль та цінність інформаційної системи для підтримки малого бізнесу.

У комерційному світі важливість утримання наявних клієнтів і розширення бізнесу є необхідним. Витрати, пов’язані з пошуком нових клієнтів, означають, що кожен наявний клієнт може бути важливим.

Чим більше можливостей у клієнта для введення бізнесу з компанією, тим краще, і одним із способів досягти цього є відкриття таких каналів, як прямі продажі, онлайн-продажі, франшизи, використання агентів тощо. Однак чим більше у вас клієнтів, тим більша потреба в управлінні взаємодією з клієнтською базою.

Інформаційна система допомагає підприємствам отримати уявлення про поведінку своїх клієнтів і змінити їх бізнес-операції, щоб забезпечити найкраще обслуговування клієнтів. Інформаційна система допомагає бізнесу визнати цінність своїх клієнтів і отримати вигоду з покращення відносин із клієнтами. Для того щоб ефективно задовольнити потреби клієнтів потрібно розуміти, що їм необхідно.

За допомогою інформаційної системи можна дізнатися про купівельні звички та вподобання клієнтів, профілювання окремих осіб і груп для більш ефективного маркетингу та збільшення продажів, зміна способу діяльності для покращення обслуговування клієнтів і маркетингу.

У сучасному бізнес-середовищі, що швидко змінюється та автоматизується, важливо подумати про поєднання інформаційної системи із програмним забезпеченням BPM. BPM, абревіатура від управління бізнес-процесами, — це підхід, який зосереджується на оптимізації бізнес-операцій для підвищення ефективності організації та досягнення бізнес-цілей. Таким чином, BPM включає поєднання моделювання, автоматизації, виконання, контролю, вимірювання та оптимізації потоків ділової активності. Технологія BPM в поєднанні із інформаційною системою дозволяє бізнесу швидко адаптуватися до динамічного

бізнес-середовища. Це дає користувачам негайний доступ до всієї важливої інформації, яка їм потрібна, що значно прискорює робочі процеси. Основна мета BPM – узгодити всі організаційні елементи для підвищення ефективності роботи.

1.2 Поняття інформаційна система

Інформаційна система – прикладне програмне забезпечення для організацій, призначене для автоматизації стратегій взаємодії з замовниками (клієнтами), зокрема для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів та історії взаємин з ними, встановлення і поліпшення бізнес-процесів і подальшого аналізу результатів.

Даний термін міцно засів в системі корпоративних комерційних процедур великої кількості фірм в світі. Говорячи про термін інформаційна система, варто розкрити важливі моменти, які він має на увазі:

— наявність єдиного сховища інформації, звідки в будь-який момент часу можна отримати всі відомості про взаємодію з клієнтом;

— синхронізованість управління множинними каналами взаємодії;

— постійний аналіз зібраної інформації про роботу з клієнтом і прийняття відповідних організаційних рішень.

Такий підхід до справи передбачає, що при будь-якій взаємодії з клієнтом, співробітнику будь-якого каналу організації доступна вся інформація про всі взаємини з клієнтом і рішення про подальшу роботу приймається на її основі.

В інформаційній системі відбуваються такі процеси:

— введення інформації, отриманої з джерел інформації;

— опрацювання (перетворення) інформації;

— зберігання вхідної і опрацьованої інформації;

— виведення інформації, призначеної для користувача;

— відправка / отримання інформації мережею.

Розробка інформаційної системи передбачає вирішення двох таких завдань:

—наповнення системи даними певної предметної області;

—створення графічного інтерфейсу користувача для отримання необхідної інформації.

Дані в інформаційної системи можуть зберігатися у двох видах, або у неструктурованому або у структурованому вигляді. Неструктуровані дані — це текстові документи (можливо, ілюстровані): статті, реферати, журнали, книги тощо. Системи, в яких зберігаються неструктуровані дані, не завжди дають конкретну відповідь на запитання користувача, можуть видати текст документа або перелік документів, у яких потрібно шукати відповідь. Структурування даних передбачає задання правил, що визначають їхню форму, тип, розмір, значення тощо.

Бажаючи підкреслити використання електронно-обчислювальної техніки для автоматизації інформаційних процесів, сучасні інформаційні системи часто називають «автоматизованими інформаційними системами».

Як інформаційну систему можна розглянути багато об'єктів: телебачення, мережу мобільного зв'язку, цифрові фотоапарати і відеокамери.

До інформаційної системи дані надходять від джерела інформації. Ці дані надсилають на зберігання чи певного опрацювання у систему й потім передають споживачеві.

Так як інформаційна система є Web-додатком, то для її роботи обов'язково потрібно Web-сервер і система управління базами даних (СКБД).

На робочому місці клієнта ніякого додатково програмного забезпечення, крім як стандартного браузера, встановлювати не потрібно. Структуру функціонування інформаційної системи можна уявити, як дві бази даних, які обслуговують всі етапи взаємини з замовниками. У якості першої БД виступає PostgreSQL, де міститься інформація про всі проведені операції, з якими не потрібно проводити онлайн роботу, а друга база даних являє собою сервер для онлайн роботи з клієнтами. З її допомогою здійснюється бронювання записів та прийом замовлень. Реалізація клієнт-серверної взаємодії означає, що, як для зовнішніх, так і внутрішніх, по відношенню до компанії, користувачів весь доступ до інформації, яка зберігається

в архівній базі даних, здійснюється за допомогою веб-браузера. Присутні дві складові в аналітичній частини функціоналу CRM-сервера: СУБД для зберігання, а також базової обробки інформації та інструменти OLAP - розробка методів для обробки інформації, що включає формування і динамічну публікацію звітів і документів. Застосовується фахівцями для максимально швидкої обробки складних запитів до бази даних і для аналізу даних в онлайн режимі. Найбільше використання OLAP знаходить в продуктах для бізнес-планування та сховищах даних.

1.3 Класифікація інформаційних систем

Велика кількість інформаційних систем пішло з систем, які автоматизували певні процеси взаємодії з клієнтами. Багато, з нині існуючих це нащадки систем SFA, SMS і CSS. Раніше використовувалися як системи для автоматизації сервісних служб і т.д. Однак, в середині нульових років інформаційні системи отримали всі зазначені можливості по обробці інформації. Існує безліч класифікацій інформаційних систем (рис. 1.1).

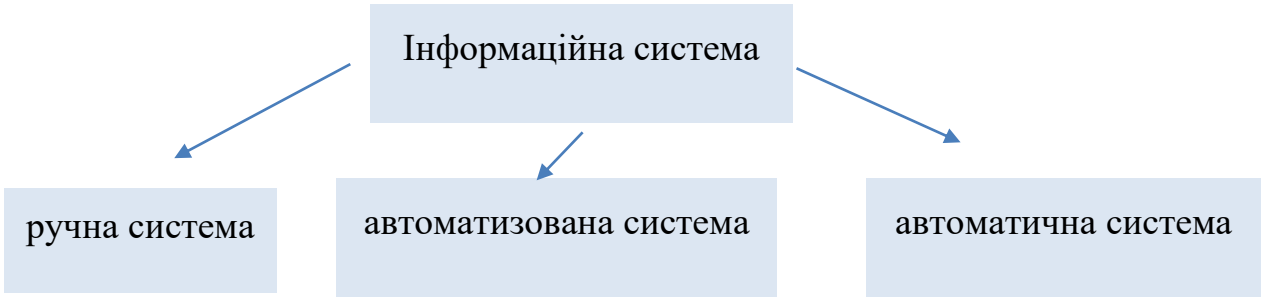

Рисунок 1.1 — Види інформаційних систем за ступенем автоматизації.

За ступенем автоматизації ІС поділяються на три великі групи, такі як, ручні, в яких людина виконує всі необхідні опрацювання, автоматизовані, у яких приблизно половину даних, що потрібно опрацювати виконує теж людина, а інша частина здійснюється автоматично, та виділяють автоматичні, в яких за допомогою технічних засобів, що зовсім не передбачають втручання людини, здійснюється опрацювання даних.

За масштабом використання ІС поділяють на одиночні, тобто ті, що реалізовано комп'ютері, що не має підключення до комп'ютерної мережі, на групові, які будуються на основі локальної комп'ютерної мережі для колективного використання інформації, на корпоративні, що реалізовують клієнт-серверну архітектуру та територіально віддалені комп'ютери, та на глобальні, такі, як Інтернет, що охоплюють територію контененту, держави (рис. 1.2).

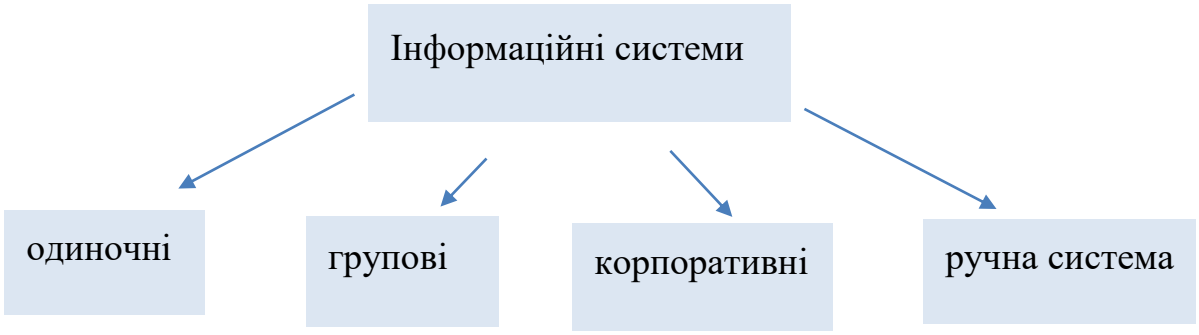

Рисунок 1.2 — Види інформаційних систем за ступенем автоматизації.

За місцем діяльності ІС поділяють на наукові, що призначені для науковців, моніторингу експериментів, аналізу інформації, автоматизованого проектування, які призначені, зазвичай, для роботи інженерів, на керування технологічними процесами та ІС організаційного керування, що використовуються підприємствами (рис. 1.3).

За сферою призначення ІС утворюються для задоволення інформаційних потреб в межах конкретної предметної галузі, то кожна предметна галузь (в сфері призначення) відповідає свій тип ІС.

Перераховувати всі ці типи немає змісту, оскільки кількість предметних галузей велика, але можна вказати, наприклад, такі типи ІС:

— економічна ІС — інформаційна система призначена для виконання функцій управління на підприємстві;

— медична ІС — інформаційна система призначена для використання в лікувальному або лікувально-профілактичному закладі;

— географічна ІС — інформаційна система, забезпечуюча збір, збереження, обробку, доступ, відображення і розповсюдження даних;

— адміністративні;

— виробничі;

— навчальні;

— екологічні;

— криміналістичні;

— військові та інші.

Рисунок 1.3 — Види інформаційних систем за місцем діяльності.

1.4 Огляд існуючих систем підтримки малого бізнесу

На сьогоднішній день є досить велика кількість програм, які призначені для підтримки малого бізнесу, але усі вони є вузько направленими у своїй області та мають кожен свої недоліки та переваги. Варто виділити систему, яка дає змогу вести облік клієнтів — Capsule. Однак більшість подібних програм складають системи моніторингу відвідувачів салонів краси. Існує велика кількість схожих систем моніторингу. Нижче буде детально розказано про кожну з них.

Capsule — це веб-сервіс з функціями інформаційної системи, який активно використовується малим бізнесом та відділами продажу. Він допомагає керувати відносинами з клієнтами, продажами, листами, зберігати всю історію взаємодії, працювати із задачами та виконувати інші операції. Перевагою Capsule можна назвати можливість інтеграції системи з Google Apps та іншими сервісами. Із важливих особливостей рішення можна відзначити: імпорт контактів із Outlook, Gmail, візитних карток, таблиць і CSV, кастомізація під бізнес-процеси за допомогою категорій, керування списком задач, календар, відстеження етапів операцій (рис. 1.4).

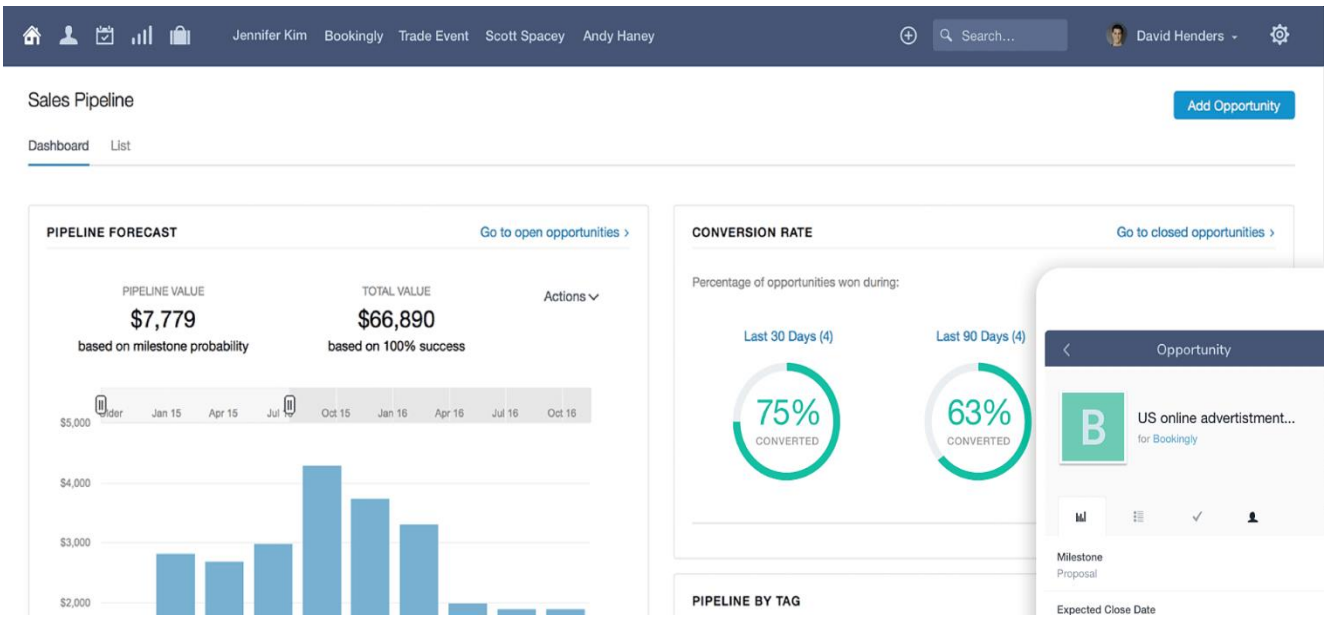

Рисунок 1.4 — Інформаційна система Capsule

Однак тут варто зазначити, що це дає користувачам Capsule явну перевагу перед їхніми старомодними електронними таблицями. Є можливість отримати доступ до всіх даних про продажі та клієнтів у полі. Щоразу, коли вносяться зміни до запису, Capsule автоматично оновлює всю систему. Все, що це, підключення до Інтернету.

Основними перевагами даної програми є нагадування компаніям звернутися до клієнтів, з якими вони давно не спілкувалися, відмінні функції створення нотаток і обміну ними, та велика кількість інтеграцій з найбільш часто використовуваними інструментами для малого та середнього бізнесу. Крім того існують недоліки такі, як малий набір функцій для великих компаній, важкість розуміння інтерфейсу, та складності знайти потрібну функцію, а також нешироки вибір параметрів для аналітики.

Є можливість імпортувати наявні контакти за допомогою файлу CSV. Capsule не має автоматичної синхронізації, яку можна зустріти у інших програм. Але досить легко перенести свої контакти в Outlook, Google або Excel у файл CSV, а потім імпортувати їх у Capsule. Це заощаджує багато часу проти введення інформації вручну. Щойно контакт був імпортований, запис міститиме всі дані клієнтів. Також є розділ, у якому можна включати нотатки та вкладати файли до

кожного контакту. Завдання також можна створювати в записі контакту та автоматично відображатися в лівій частині панелі як "Майбутні завдання".

Capsule інтегрується з телефонними системами Yay, Kixie, Circle Loop і Invoco.

Наступною інформаційною системою, що буде розглядатися є Bitrix24 (рис. 1.5).Ця система пропонує широкий спектр інструментів керування електронною комерцією, включаючи керування продуктами, обробку замовлень і платежів, а також відстеження запасів. Користувачі з платними підписками можуть керувати необмеженою кількістю замовлень і продуктів безпосередньо з менеджера по роботі з клієнтами. Таким чином, це один з наших виборів для найкращого програмного забезпечення для електронної комерції.

Бітрікс24 має надійний вбудований телефон для вхідних і вихідних дзвінків. Власники малого бізнесу можуть отримати доступ до місцевих номерів у більш ніж 50 країнах і здійснювати відеодзвінки, щоб зв’язуватися з віддаленими співробітниками та клієнтами з будь-якої точки світу.

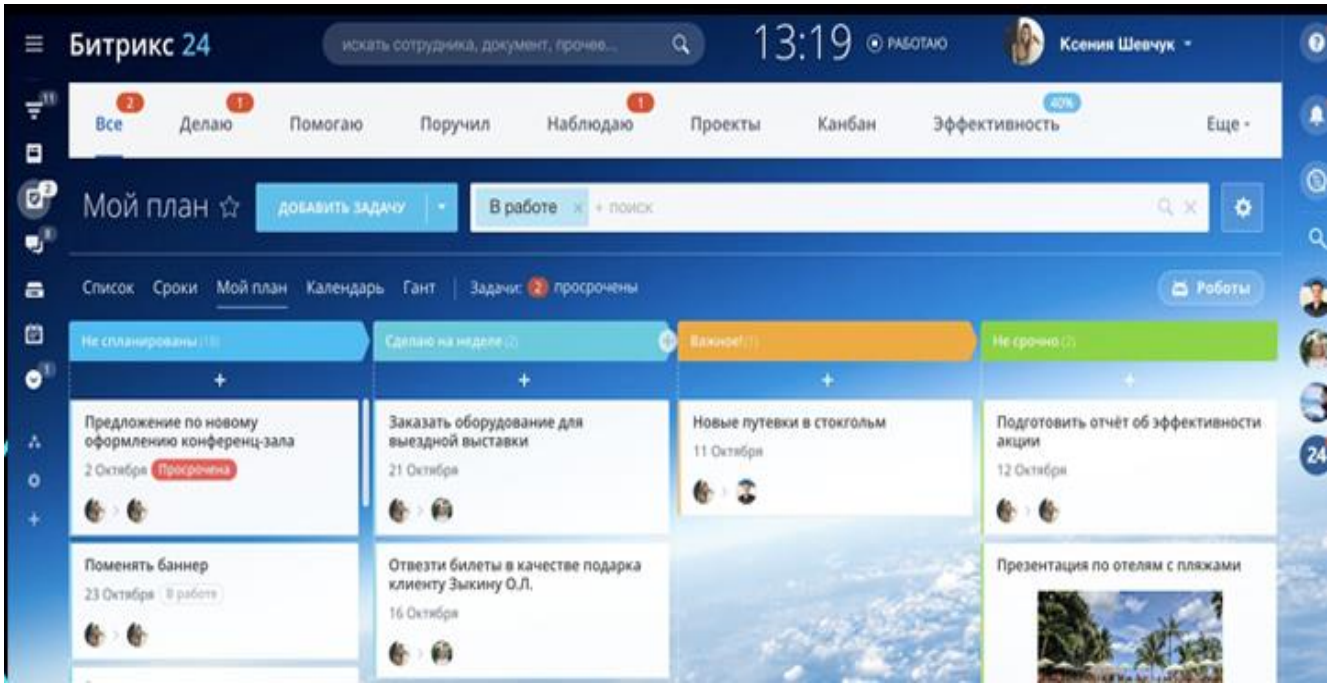

Рисунок 1.5 — Інформаційна система Bitrix24

Основні властивості це введення бази даних, керування замовленнями, історія взаємодії із клієнтами, керування підтримкою, звіти, розсилання листів

через пошту, використання діаграми Ганта, експорт та імпорт даних. Основними перевагами цієї системи є окремий список потенційних клієнтів, та прямий контакт із клієнтами, доступний інтерфейс користувача Kanban, а також постійне сповіщення про можливості, завдання та діяльність.

Основними недоліками є складний інтерфейс користувача, обмеження соціальних мереж та висока вартість даної системи.

Остання у цьому списку інформаційна система має назву Scoro є хмарним інструментом управління бізнесом, розробленим для малого та середнього бізнесу, особливо в сферах консалтингу, реклами, IT тощо. Програмне забезпечення пропонує широкий набір корисних функцій, включаючи відстеження та планування роботи, співпрацю, управління відносинами з клієнтами, управління проектами, інформаційну панель виставлення рахунків і пропозицій, а також розширені звіти. Він навіть має зручний інтерфейс, який робить процеси точними, безперебійними та швидкими для новачків (рис. 1.6).

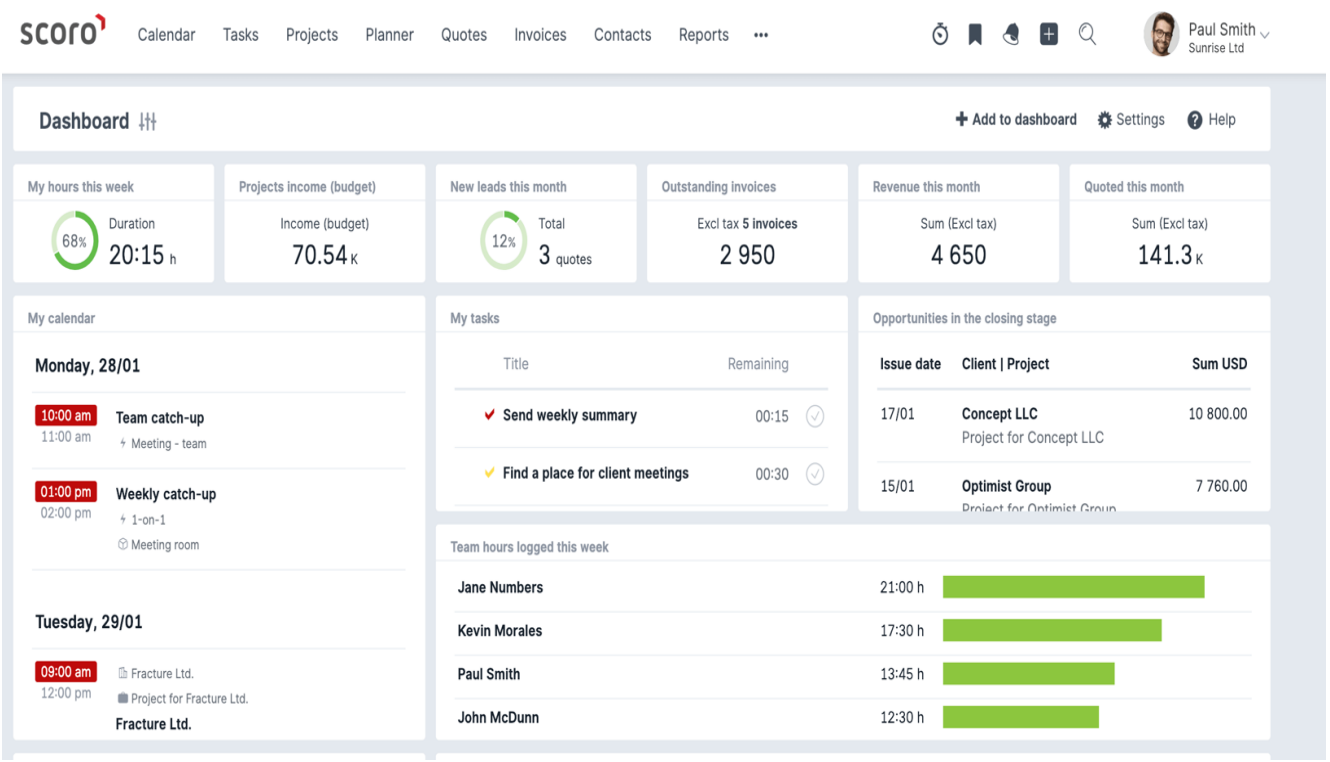

Рисунок 1.6 — Інформаційна система Scoro

Ключова функція Scoro, яка називається центром керування, відображає завдання, події календаря, ключові дані про ефективність, інформацію облікового

запису та інші матеріали вашого бізнесу. Ця програма пропонує локалізоване рішення для забезпечення інтеграції для європейського ринку.

Scoro допомагає бізнесу заощадити багато часу з точки зору управління проектами. Є можливість додавання завдання в будь-який час, щоб зробити кожен процес безперебійним і швидшим. Усі ці завдання автоматично пов'язуються з вашими клієнтами, а завдання електронної пошти можуть бути заархівовані в поштовій скриньці. Крім того, Scoro збирає всі контактні дані постачальників і клієнтів в одному місці для швидкого доступу. За допомогою того ж інтерфейсу є можливість створювати та виставляти рахунки-фактури. Scoro може допомогти вашому бізнесу виставляти рахунки-фактури для продажу, кредиту та передоплати в різних валютах, а також допомагає відстежувати всі рахунки та витрати в певному проекті, спрощуючи фінанси та бухгалтерські процеси.

Інформаційна панель дозволяє бачити та відстежувати хід проекту в режимі реального часу, дає можливість переглядати оплачувані чи неоплачувані проекти та завдання, які ще непризначені чи незавершені. Також пропонуються варіанти для вибору шаблонів PDF для замовлень, звітів, пропозицій та рахунків-фактур. Scoro можна інтегрувати з різним провідним програмним забезпеченням, підтримує пристрої Android та iOS.

Дозволяє налаштовувати індивідуальні інформаційні панелі, щоб отримати вичерпну інформацію про цілі, завдання кожного клієнта. Має можливість налаштування звітів на рівні підприємств, що охопоплює всі аспекти бізнесу, автоматично збирається і налаштовується, для можливості демонстрації загальної картини фінансів фірми.

Зручний інтерфейс, швидка, інформативна та компактна платформа, система виставлення рахунків і нагадувань є основними перевагами системи.

Для налаштування синхронізації з Outlook потрібен час і деякі специфічні опції продажу, що не вклчюені в стандартну версію.

Після проведення аналізу існуючих аналогів інформаційних систем підтримки малого бізнесу, можна скласти порівняльну характеристику переваг та недоліків, що наведена у таблиці 1.1.

Таблиця 1.1 — Переваги та недоліки альтернативних додатків

Назва системи	Переваги	Недоліки
Capsule	Експорт та імпорт даних Система сповіщень	Складний інтерфейс користувача Обмежений список функцій
Bitrix24	Прямий контакт із клієнтами Система сповіщень	Складний інтерфейс користувача Обмеження соціальних мереж
Scoro	Зручний інтерфейс Система нагадувань	Складність синхронізації Висока ціна

Отже, було обгрунтовано цінність та роль інформаційної системи для підтримки малого бізнесу. Проведено огляд складових частин інформаційної системи та її класифікація. Здійснено аналіз існуючих аналогів з проведенням їх порівняння, та на основі цього зроблено варіантний вибір інструментів розробки.

2 АНАЛІЗ ТЕХНОЛОГІЙ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Огляд основних компонентів програмного продукту

Інформаційна система має виконувати такі функції, як авторизацію користувача, створення нового користувача або клієнта, підтримувати змінення та видалення клієнта, можливість створення, змінення та видалення запису користувача, моніторинг доходів, витрат та обігу клієнтів, сповіщення клієнтів про найближчий запис, аналіз важливості клієнтів, та послуг, що приносить більший дохід, користувач має можливість задавати власні цілі, що бажає досягнути за місяць, на основі яких буде здійснюватися порівняння фактичного результату роботи та очікуваного. Також інформаційна система має реалізовувати графічний інтерфейс для користувачів, де є можливість моніторингу активності записів даного користувача та суму, яку цей користувач витратив. Схему функціональних можливостей інформаційної системи показано на рисунку 2.1.

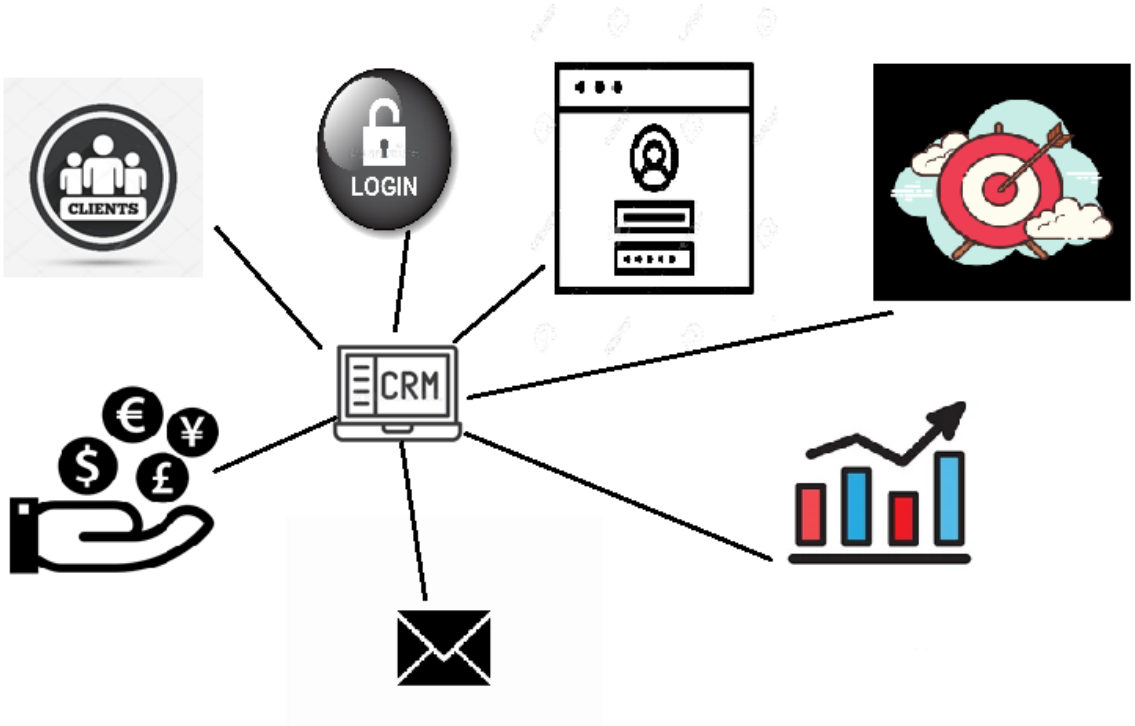

Рисунок 2.1 — Схема функціональних можливостей інформаційної системи

Для розробки інформаційної системи було використано архітектуру Model-View-Controller, що надає Фреймворк Spring Web MVC. Вагомою перевагою даної архітектури є розділення різних аспектів програми (логіки введення, бізнес-логіки та логіки інтерфейсу користувача), забезпечуючи при цьому слабке з'єднання між цими елементами. Модель інкапсулює дані програми, і загалом вони складатимуться з POJO. Представлення відповідає за відтворення даних моделі і загалом генерує вихідні дані HTML, які браузер клієнта може інтерпретувати. Контролер відповідає за обробку запитів користувачів і побудову відповідної моделі та передає її у представлення для візуалізації. Графічне зображення складових компонентів MVC наведено на рисунку 2.2.

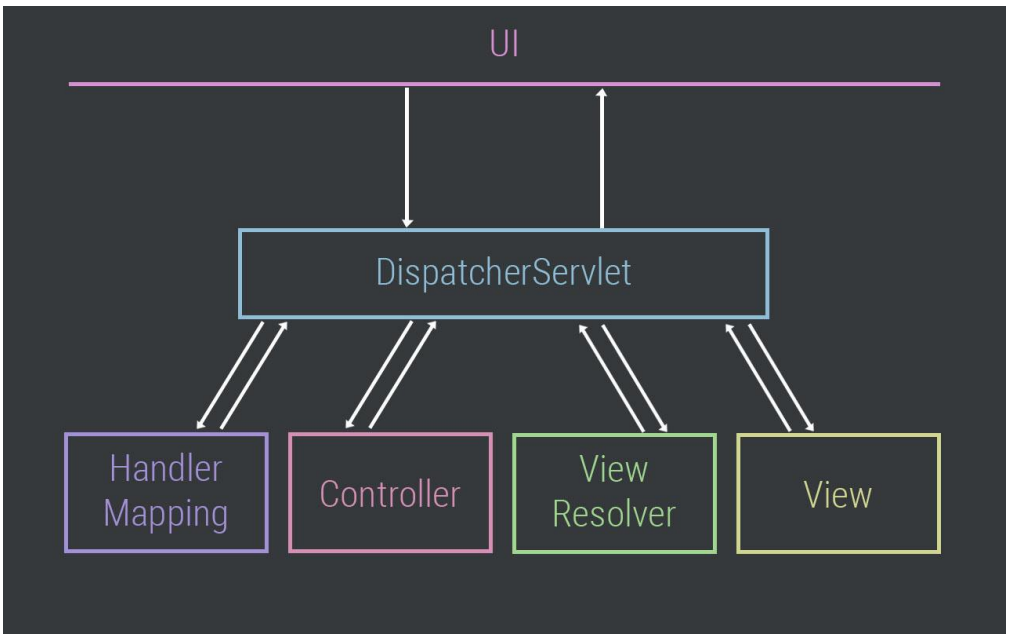

Рисунок 2.2 — Логіка роботи DispatcherServlet

Фреймворк Spring Web model-view-controller (MVC) розроблено навколо DispatcherServlet, який обробляє всі запити та відповіді HTTP. Нижче наведена послідовність подій, що відповідає вхідному HTTP-запиту до DispatcherServlet — після отримання HTTP-запиту DispatcherServlet звертається до HandlerMapping, щоб викликати відповідний контролер. Контролер приймає запит і викликає відповідні методи служби на основі використаних методів GET або POST. Метод служби встановлює дані моделі на основі визначеної бізнес-логіки та повертає ім'я

представлення в DispatcherServlet. Графічне зображення роботи DispatcherServlet наведено на рисунку 2.3.

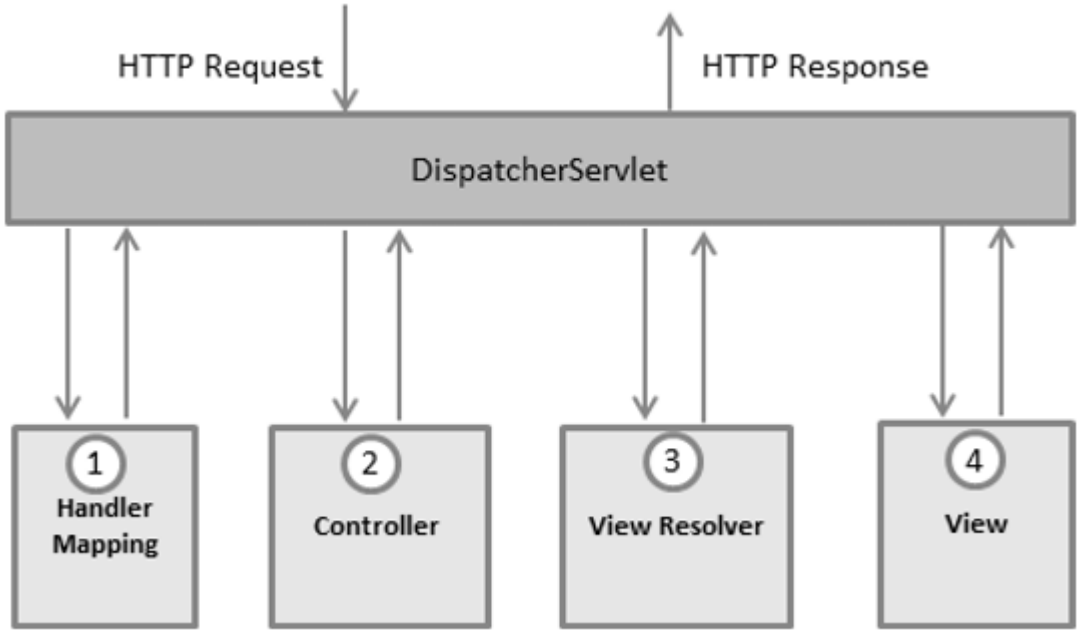

Рисунок 2.3 — Логіка роботи DispatcherServlet

DispatcherServlet скористається допомогою ViewResolver, щоб отримати визначене представлення для запиту. Після завершення перегляду DispatcherServlet передає дані моделі у подання, яке нарешті відображається у браузері.

Усі вищезгадані компоненти, тобто HandlerMapping, Controller і ViewResolver, є частинами WebApplicationContext w, який є розширенням plainApplicationContext з деякими додатковими функціями, необхідними для веб-додатків.

2.2 Вибір мови програмування

Мова програмування Java є найпопулярнішим інструментом для розробки програмних продуктів у наш час. Основною перевагою Java є її кросплатформеність. Java запускається всередині Java Virtual Machine (JVM), а це означає, що ваш код не звертається безпосередньо до операційної системи, тобто теоретично може працювати де завгодно, де є ця сама JVM, будь то Windows, Linux,

MacOS, мобільні платформи, а також у робототехніці та вбудовуваних системах. Так що дуже ймовірно, що ваша розумна кавоварка працює на Java.
За допомогою Java можна створити безліч серверних, корпоративних, мобільних, декспотних додатків, у тому числі навіть веб-додатки.

Основні переваги Java це масштабованість, тому він такий популярний серед великих проектів і стартапів (Twitter перейшов з Ruby на Java). Оскільки Java є мовою статичної типізації, її простіше та швидше підтримувати з меншою кількістю помилок. Він також має зворотну сумісність, що означає, що старі версії програм, як і раніше, працюватимуть ідеально навіть після переходу на нові версії мови.

У світі десятки тисяч Java-програмістів, які готові вам допомогти статтями (D-zone, Baeldung, Mkyong, Habr зрештою) та відповідями на запитання (Stackoverflow, gitHub).
Постійний розвиток. Java не стоїть на місці, а швидко зростає завдяки вкладу спільноти та великих компаній, таких як Google, Twitter, Oracle, RedHat, JetBrains, IBM, Intel, ARM та інші.

Існує переконання, що немає коду простішого і приємнішого до прочитання, ніж код, написаний на Java. Це упереджено. Звичайно, будь-якою мовою можна писати поганий код. І хороший код можна писати будь-якою мовою (або майже будь-якою). Тим не менш, важко заперечувати, що використання кращих практик написання коду і Java в одному місці дають чудовий результат.

Для розв’язання нашої проблеми потрібний інструмент, який би давав змогу працювати з розсиланням повідомлень на пошту, була сумісною із базою даних, містила інструменти для авторизації клієнтів. Саме таким інструментом є фреймворк Spring[26].

2.2.1 Аналітичний огляд фреймворків для створення додатків

У епоху стрімкого розвитку IT-індустрії та попиту створення програмного забезпечення інструменти, що дозволяють здіснити цю задачу набирають колосальних обертів. Одним із популярних інструментів створення програмного

забезпечення є Spring framework. Spring Framework — це платформа Java, яка надає повну підтримку інфраструктури для розробки Java-додатків. Spring дозволяє розробляти програми за допомогою POJO. Перевага використання тільки POJO полягає в тому, що вам не потрібен продукт-контейнер EJB, такий як сервер додатків, є можливість використовувати лише контейнер сервлетів, такий як Tomcat. На противагу Spring Framework будуть розглядатися ще такі фреймворки, як Micronaut, Helidon SE та Ktor, що є його аналогами.

Метою роботи є проаналізувати низку переваг та недоліків фреймворку Spring відносно інших фреймворків.

У даний час немає дефіциту фреймворків для створення мікросервісів на Java. Найстарішим фреймворком є Spring. Він дуже популярний з кількох причин: по-перше, підхід до впровадження залежності Spring заохочує писати тестований, якісний код, по-друге, простий у використанні, але з потужними можливостями керування транзакціями бази даних, по-третє, Spring спрощує інтеграцію з іншими фреймворками Java, такими як JPA/Hibernate ORM, Struts/JSF/i т.д. веб-фреймворками, і є сучасним і незамінним фреймворком Web MVC для створення веб-додатків. Впровадження залежностей (DI) та інверсія управління (IoC) дозволяють писати незалежні один від одного компоненти, що дає переваги у командній розробці, переносимості модулів тощо. Spring IoC контейнер управляє життєвим циклом Spring Bean і налаштовується на зразок JNDI lookup. Проект Spring містить у собі безліч підпроектів, які зачіпають важливі частини створення софту, такі як веб-сервіси, веб-програмування, робота з базами даних, завантаження файлів, обробка помилок і багато іншого. Все це налаштовується в єдиному форматі і спрощує підтримку програм.Усі вище перераховані можливості роблять його таким популярним інструментом для створення додатків. Але існують також інші фреймворки, що є досить молодими, тому не містять багатьох функціональних можливостей, що є у Spring.

Першим розглянемо Micronaut. Micronaut — це фреймворк на JVM для побудови легких модульних додатків. Має кілька можливостей, які роблять його відмінним фреймворком для розробки хмарних додатків (cloud-native). Він

підтримує безліч механізмів виявлення сервісів (відкриття сервісів), такі як Eureka і Consul, а також працює з різними системами розширеного трасування (розподілене відстеження), такими як Zipkin і Jaeger. Додатково, він надає підтримку створення лямбда функцій AWS, що дозволяє легко створювати безсерверні додатки.

Наступним будемо розглядати Helidon. Фреймворк був створений в Oracle для внутрішнього використання, згодом став відкритим. Існують дві моделі розробки, засновані на цій структурі: Standard Edition (SE) і MicroProfile (MP). API MicroProfile, як правило, дотримуються декларативного стилю, який інтенсивно використовує анотації. Helidon SE – це наш власний реактивний API. Він використовує функціональний стиль програмування з невеликим використанням анотацій. Найкраще використовувати цей фреймворк при написанні невеликих додатків, що не працюють із базами даних. Останнім йде Ktor.

Ktor — це асинхронний фреймворк з відкритим кодом для створення мікросервісів і веб-додатків. Він був розроблений спільно з Kotlin компанією Jetbrains. Його легко встановити та використовувати. Його асинхронна властивість дозволяє йому отримувати кілька запитів завдяки використанню coroutines, а також він є мультиплатформенним. Має багато можливостей, але його недоліками є те, що він не підтримує Java, сумісний лише із Kotlin, та використовується лише для розробки мобільних додатків, а не десктопних додатків.

На наведеній нижче таблиці 2.1 представлений порівняльний аналіз фреймворку Spring відносно інших фреймворків.

На основі всього нище написаного можемо зробити висновок, що хоча Micronaut, Helidon SE та Ktor потребують менше пам’яті для свої роботи, вони програють відносно фреймворка Spring, оксільки не мають багато функціональних можливостей, таких як можливість інтеграції із third-party, можливість побудувати стійкі сервіси із Cloud, висока безпека додатків, підтримка automation тестування та наявність детальної документації.

Таблиця 2.1 – Порівняльний аналіз фреймворку Spring відносно інших фреймворків

Фреймворк	Мови програмування, що підтримує	Час компіляції	Можливість інтеграції із third-party	Підтримка automation тестування	Наявність документації	Безпека
Micronaut	Groovy, Java, Kotlin	1.48 с	низька	середня	мінімальна	низька
Helidon	Java, Kotlin	2,2 с	низька	низька	мінімальна	низька
Ktor	Kotlin	1.4 с	низька	низька	мінімальна	низька
Spring	Groovy,Java, Kotlin	1.33 с	висока	висока	повна	висока

Розглянуто та проаналізовано переваги та недоліки фреймворку Spring відносно інших фреймворків та його особливості, що використовуються в середовищі розробки IntelliJ IDEA для реалізації додатків. Показано, що фреймворк Spring має низку переваг та дає можливість розробляти додатки із інтеграцією із Spring Security, шаблонізатором Thymeleaf та Spring Data.

2.3 СУБД PostgreSQL для зберігання даних додатку.

PostgreSQL — це система управління базами даних корпоративного класу з відкритим вихідним кодом. Вона підтримує як SQL, так і JSON для реляційних і нереляційних запитів для розширюваності та відповідності SQL. PostgreSQL підтримує розширені типи даних і функції оптимізації продуктивності, які доступні лише в дорогих комерційних базах даних, таких як Oracle і SQL Server. Допомагає розробникам створювати програми, захищаючи цілісність даних, сумісна з різними платформами, які використовують усі основні мови та проміжне програмне забезпечення, повна підтримка мережевої архітектури клієнт-сервер. Підтримка JSON дозволяє підключатися до інших сховищ даних, таких як NoSQL, які діють як федеративний центр для баз даних поліглотів. PostgreSQL має зручний графічний інтерфейс клієнта pgAdmin.

Він являє собою графічний клієнт для роботи із сервером, через який ми в зручному вигляді можемо створювати, видаляти, змінювати базу даних та керувати ними. Графічний інтерфейс клієнта pgAdmin показано на рисунку 2.4.

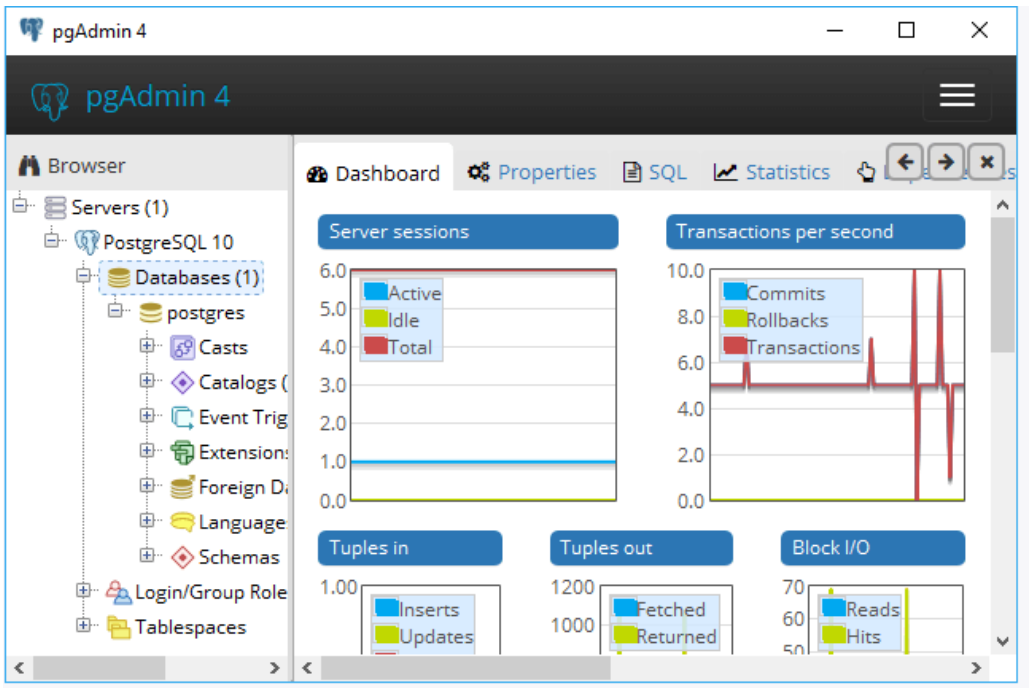

Рисунок 2.4 — Графічний інтерфейс клієнта pgAdmin

Після успішного логіна створюється тільки одна база данних - postgres. Для створення нових баз потрібно написати команди за допомогою консольного клієнта psql або використовуючи графічний клієнт pgAdmin.

2.4 Функціональні можливості та алгоритм роботи програмного продукту

Інформаційна система має містити базові функції для спрощення роботи клієнта, наприклад, відстеження фінансів, категорій фінансів, різноманітна статистика, сторінку для перегляду особистої інформації користувача.

Дана інформаційна система виконує низку різних функцій, таких як:

— перегляд списку створених клієнтів із обов’язковими полями, що містять особисту інформацію користувача ;

— створення нових клієнтів;

— змінення даних вже раніше створених клієнтів;

— нагадування клієнтам, що записані на сьогодні про їхній запис, а саме відправляється смс-нагадування на пошту;

— перегляд графіку записів для аутентифікованого користувача на сьогодні або вибрану дату;

— додавання нових записів відвідування;

— перегляд статистики відвідувань за датою;

— перегляд статистики фінансів за датою або сьогодні.

Для того щоб досягнути поставлених завдань, потрібно реалізувати даний алгоритм програмного забезпечення зображено на рисунку 2.4. Даний алгоритм описує роботу програми від початку і докінця.

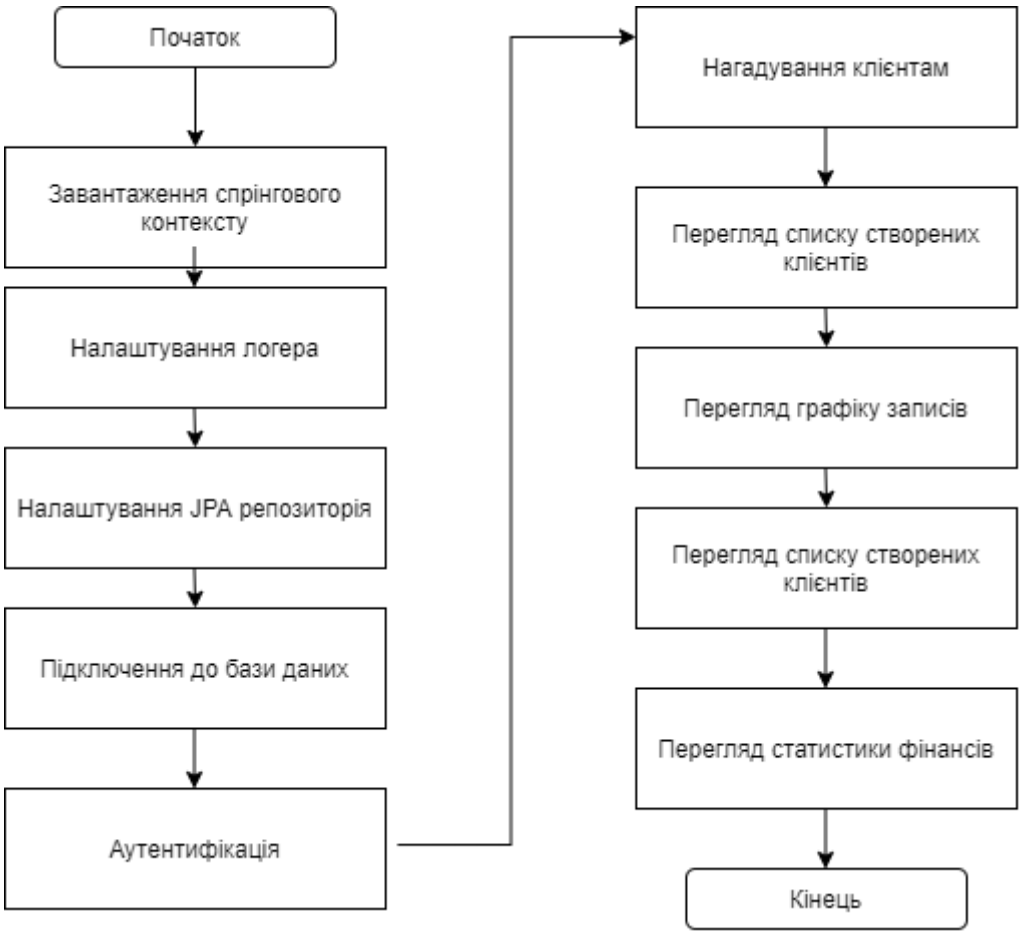

Рисунок 2.5 — Алгоритм програми

Інформаційна система підтримки малого бізнесу у сфері послуг повина мати такі модулі:

—модуль логера;

—модуль JPA репозиторія;

—аутенфікації юзера;

—модуль клієнтів;

—модуль графіка записів;

—модуль фінансів;

—модуль статистики;

—задання цілей клієнта;

2.5 Рекомендації розгортання сервісу в хмарах

У епоху технологічного прогресу додатків, що інтегровані із хмарою є безліч, оскільки це дає купу переваг, таких як економічність, масштабованість, можливість обробки та зберігання великої кількості інформації та багато інших. Одним із найпопулярніших хмарних сервісів є Amazon Web Services. AWS має декілька типів сервісів, що представлені на рисунку 2.5.

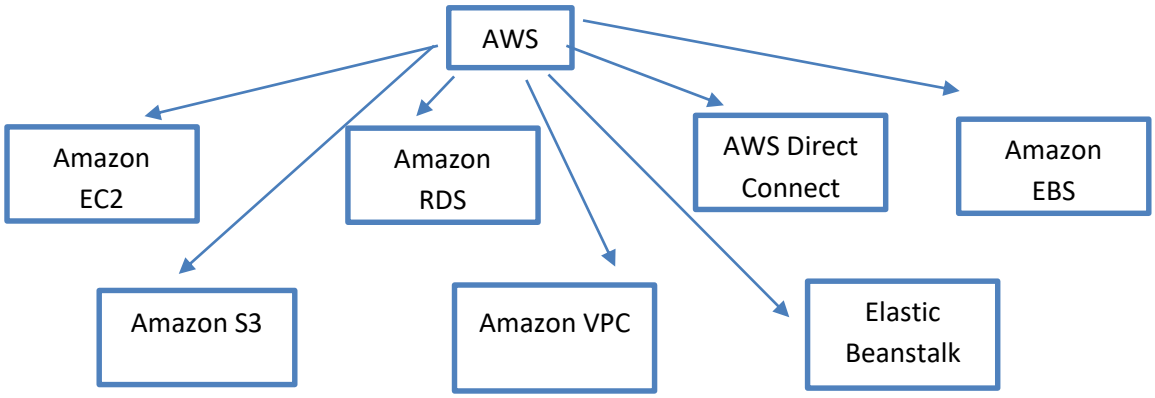

Рисунок 2.5 — типів сервісів AWS

Для розвертання даного програмного продукту рекомендовано використовувати Amazon EC2, Elastic Beanstalk та RDS.

Amazon Elastic Compute Cloud (Amazon EC2) — це веб-сервіс, який забезпечує безпечну обчислювальну потужність у хмарі зі змінним розміром. Він розроблений, щоб полегшити розробники веб-обчислень. Простий веб-інтерфейс Amazon EC2 дозволяє отримати та налаштувати потужність з мінімальним тертям. Він надає вам повний контроль над вашими обчислювальними ресурсами та дозволяє працювати в перевіреному обчислювальному середовищі Amazon. Amazon EC2 скорочує час, необхідний для отримання та завантаження нових екземплярів сервера (так звані екземпляри Amazon EC2), до хвилин, дозволяючи швидко масштабувати потужність, як збільшувати, так і зменшувати, коли ваші вимоги до обчислювальної техніки змінюються. Amazon EC2 змінює економіку

обчислень, дозволяючи платити лише за потужність, яку ви фактично використовуєте. Amazon EC2 надає розробникам і системним адміністраторам інструменти для створення стійких до збоїв додатків та ізоляції від поширених сценаріїв збоїв.

AWS Elastic Beanstalk — це проста у використанні служба для розгортання та масштабування веб-додатків і служб, розроблених за допомогою Java, .NET, PHP, Node.js, Python, Ruby, Go і Docker на знайомих серверах, таких як Apache, Nginx, Passenger , а також інформаційні послуги Інтернету (IIS).

Ви можете просто завантажити свій код, і AWS Elastic Beanstalk автоматично оброблятиме розгортання, починаючи від надання потужності, балансування навантаження та автоматичного масштабування до моніторингу працездатності програми. У той же час ви зберігаєте повний контроль над ресурсами AWS, які забезпечують роботу вашої програми, і можете отримати доступ до базових ресурсів у будь-який час.

Служба реляційної бази даних Amazon RDS спрощує налаштування, експлуатацію та масштабування реляційної бази даних у хмарі. Він забезпечує економну потужність із можливістю зміни розміру, а також автоматизує такі трудомісткі адміністративні завдання, як забезпечення обладнання, налаштування бази даних, виправлення та резервне копіювання. Це звільняє вас зосередитися на ваших програмах, щоб ви могли надати їм необхідну швидку продуктивність, високу доступність, безпеку та сумісність.

Amazon RDS доступний для кількох типів екземплярів бази даних — оптимізованих для пам’яті, продуктивності або введення/виводу — і надає вам на вибір шість знайомих механізмів баз даних, включаючи Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database та SQL Server.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Налаштування Spring Security

Для захисту та контролю доступу у додатків використовується фреймворк Spring Security, що забезпечує функції аутентифікації та авторизації додатків, що написані на мові Java. Його особливостями є комплексна та розширювана підтримка як аутентифікації, так і авторизації, захист від атак, таких як фіксація сесії, клікджекінг, підробка міжсайтових запитів, інтеграція API сервлетів та додаткова інтеграція з Spring Web MVC.

Ключовими правилами роботи аутентифікації є те, що користувачеві буде запропоновано увійти в систему, надавши ім'я (логін або email) та пароль. Ім'я користувача та пароль об'єднуються в екземпляр класу UsernamePasswordAuthenticationToken(екземпляр інтерфейсу Authentication) після чого він передається екземпляру AuthenticationManager для перевірки. Якщо пароль не відповідає імені користувача буде викинуто виняток BadCredentialsException з повідомленням “BadCredentials”. Якщо аутентифікація пройшла успішно, повертає повністю заповнений екземпляр Authentication. Для користувача встановлюється контекст безпеки шляхом виклику методу SecurityContextHolder.getContext().setAuthentication(…), куди передається об'єкт, який повернув AuthenticationManager.

Для того щоб додати в свій проект Spring Security необхідно, додати потрібні залежності до pom.xml.

```
<dependency>
<groupId> org.springframework.security </groupId>
 <artifactId> spring-security-core </artifactId>
<version> 5.5.2 </version>
</dependency>
<dependency>
<groupId> org.springframework.security </groupId>
<artifactId> spring-security-web </artifactId>
<version> 5.5.2 </version>
```

</dependency>

А також залежності для налаштування Spring Security:

<dependency>

<groupId> org.springframework.security </groupId>

<artifactId> spring-security-config </artifactId>

<version> 5.5.2 </version>

</dependency>

<dependency>

<groupId> org.springframework.boot </groupId>

<artifactId> spring-boot-starter-security </artifactId>

<version> 2.4.3 </version>

</dependency>

Необхідно створити новий конфігураційний клас та зробити розширення існуючого класу WebSecurityConfigurerAdapter.

Наступним кроком потрібно перевизначити метод Configure, де вказати поведінку для аутентифікації.

```
protected void configure (HttpSecurity http) throws Exception {
http.authorizeRequests()
.anyRequest().authenticated().and()
.formLogin()
.defaultSuccessUrl("/customers")
.and()
.logout().logoutSuccessUrl("/customers");
```

Для того щоб в базі даних не зберігати незахищений пароль потрібно створити об’єкт сервісного інтерфейсу для кодування паролів.

```
@Bean
public PasswordEncoder passwordEncoder() {
return new BCryptPasswordEncoder();
```

DaoAuthenticationProvider, є одним з перших підтримуваних провайдерів у AuthenticationProvider. Він спирається на UserDetailsService (як DAO) для пошуку

імені користувача, пароля та GrantedAuthority. Він ідентифікує користувачів, просто порівнюючи пароль надісланий в UsernamePasswordAuthenticationToken з паролем, який був завантажений UserDetailsService.

Налаштування провайдера досить просте:

PasswordEncoder та SaltSource є необов'язковими. PasswordEncoder забезпечує кодування та декодування паролів, представлених в об'єкті UserDetails, який повертається настроєним UserDetailsService. SaltSource дозволяє "додавати сіль" у паролі, що підвищує безпеку паролів в автентифікаційному репозиторії.

```
@Bean
 public DaoAuthenticationProvider daoAuthenticationProvider() {
 DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
 authenticationProvider.setPasswordEncoder(passwordEncoder());
 authenticationProvider.setUserDetailsService(customerService);
 return authenticationProvider;
```

3.2 Налаштування Spring Boot Mail Server

Spring Framework надає вам API щоб відправляти email, який включає деякі інтерфейси та класи, все лежить у 2 пакетах org.springframework.mail & org.springframework.mail.javamail.

Щоб використовувати Spring Mail у програмі Spring Boot, необхідно додати такі залежні в pom.xml:

```
<dependency>
 <groupId> org.springframework.boot </groupId>
 <artifactId> spring-boot-starter-mail </artifactId>
<version> 2.6.1 </version>
</dependency>
```

Spring Mail має багато класів (або інтерфейсів). На рисунку 3.1 показано список основних класів та інтерфейсів.

Основним його класом є MailSender, що дає можливість відправляти повідомлення через електронну пошту.

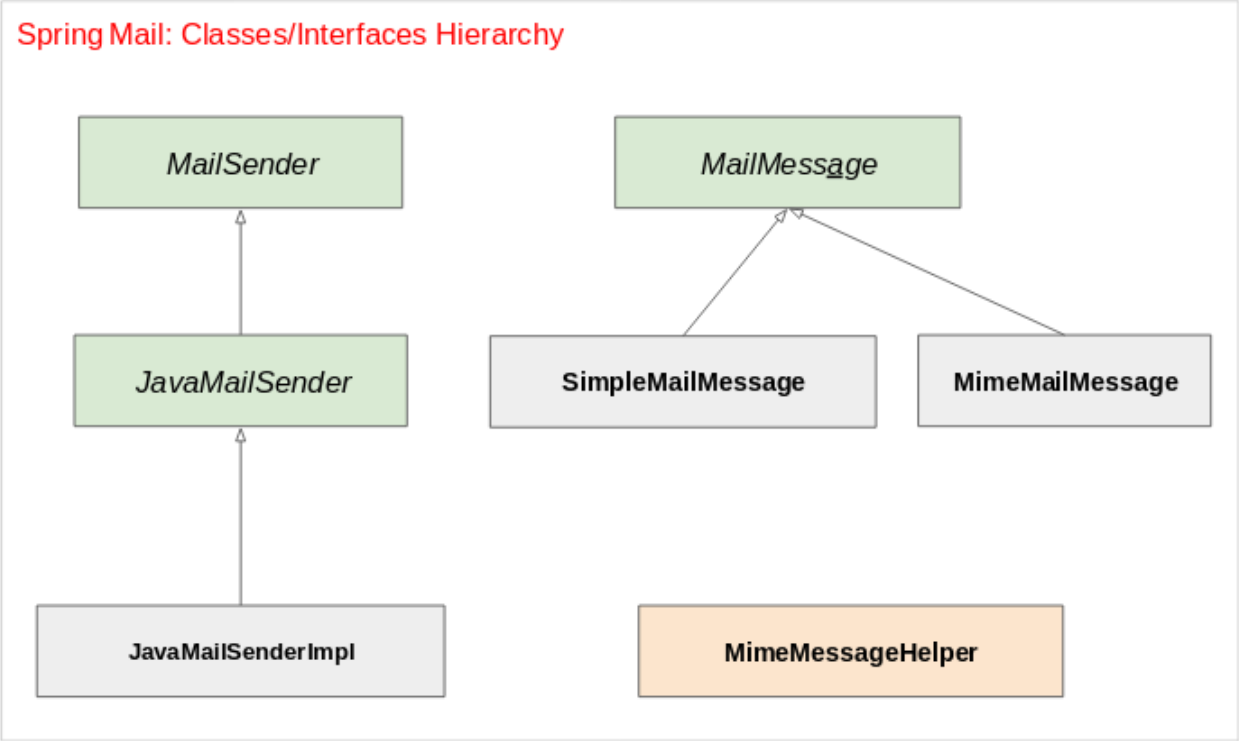

Рисунок 3.1 — Список основних класів та інтерфейсів Spring Mail

Необхідно створити обліковий запис Gmail для відправлення повідомлень, так як Gmail це поширений Mail-Server. Щоб обліковий запис Gmail міг керувати поштою через додаток Java, потрібно через налаштування Gmail дати дозвіл на роботу внутрішніх додатків.

Потрібно створити Spring-Bean для MailSender. Після встановлення залежності наступним кроком є визначення властивостей поштового сервера у файлі application.properties за допомогою простору імен spring.mail.*.Ми можемо вказати властивості для SMTP-сервера Gmail таким чином:

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=myuserforemail@gmail.com
spring.mail.password=cxruksixzdwnfpfd
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Деякі SMTP-сервери вимагають підключення TLS, тому ми використовуємо властивість spring.mail.properties.mail.smtp.starttls.enable, щоб увімкнути з'єднання, захищене TLS.

Після того, як управління залежностями та конфігурація налаштовані, ми можемо використовувати вищезгаданий JavaMailSender для надсилання електронного листа.

Потрібно створити новий клас MailService та передати йому необхідні біни:

```
private final JavaMailSender javaMailSender;
private final AttendanceRepository attendanceRepository;
```

Далі створюємо методи для створення та відправки повідомлень, у нашому випадку, буде підключатися до бази даних, шукати всіх клієнтів, що записані на сьогодні та відправляти нагадування на їхні адреси електронних пошти.

```
public void sendMail() {
List<AttendanceSchedule> customers = attendanceRepository.findAll();
List<String> emails = getEmailAddress(customers);
emails.forEach(this::createMail);
System.out.println("Mail sent successfully");
```

Створюється метод для заповнення змісту нагадування із текстом:

```
private void createMail(String receiver) {
SimpleMailMessage message = new SimpleMailMessage();
String text = " Не забудьте, що ми чекаємо на Вас сьогодні!";
String subject = "Нагадування";
```

Далі вказується адреса відправника на отримувача:

```
message.setFrom("myuserforemail@gmail.com");
message.setTo(receiver);
message.setText(text);
message.setSubject(subject);
javaMailSender.send(message);
```

Метод, що повертає сформований список записів клієнтів:

```
private List<String> getEmailAddress(List<AttendanceSchedule> customers) {
```

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd");
String time = dtf.format(LocalDateTime.now());
return customers.stream().filter(attendanceSchedule ->
attendanceSchedule.getDate().toString().contains(time))
.map(AttendanceSchedule::getClient).map(Customer::getEmail).collect(Collectors.toLi
st());
```

Над методом Sendemail ми вказали аннотацію @Scheduled, що дозволяє нам викликати даний метод з певною періодичністю.

3.3 Підключення до бази даних за допомогою бібліотеки Java.

Щоб підключити програму Spring Boot до бази даних PostgreSQL, потрібно виконати наведені нижче дії.

Додати залежність для драйвера JDBC PostgreSQL, який необхідний для того, щоб програми Java могли спілкуватися з сервером баз даних PostgreSQL.

Налаштувати властивості джерела даних для інформації про підключення до бази даних

Додати залежність для Spring JDBC або Spring Data JPA, залежно від ваших потреб:

Використати Spring JDBC для виконання простих операторів SQL

Використати Spring Data JPA для більш просунутого використання, напр. зіставляючи класи Java в таблиці та об’єкти Java в рядки, і скористайтеся перевагами Spring Data JPA API.

Потрібно додати залежність для драйвера PostgreSQL JDBC, додавши таку залежність у файлі pom.xml проекту:

```
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.3.1</version>
<scope>runtime</scope>
</dependency>
```

Далі потрібно вказати інформацію про підключення до бази даних у файлі конфігурації програми Spring Boot (application.properties) таким чином:

```
spring.h2.console.enabled=true
spring.datasource.url=jdbc:postgresql://localhost:5432/Customers
spring.datasource.username=postgres
spring.datasource.password=Master1
```

Тут URL-адреса JDBC вказує на сервер бази даних PostgreSQL, що працює на локальному хості.

Потрібно відобразити класи Java в таблиці, а об’єкти Java в рядки і скористатися перевагами об’єктно-реляційного відображення (ORM), наприклад Hibernate, потрібно використовувати Spring Data JPA і додати наступну залежність до проекту:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
  <version>2.4.3</version>
</dependency>
```

Окрім URL-адреси JDBC, імені користувача та пароля, необхідно вказати додаткові властивості наступним чином:

```
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.jdbc.time_zone=UTC
```

Потрібно закодувати клас сутності (клас POJO Java), щоб відобразити відповідну таблицю в базі даних, як показано нижче:

```
@Entity
@Data
@Table(name = "Customer")
@NoArgsConstructor
```

Створюється клас Customer, у якому оголошуються приватні змінні для імені, прізвища, ID, паролю та облікового запису клієнта:

```
public class Customer {
  @Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;
@Column(name = "first_name", nullable = false)
private String firstName;
@Column(name = "last_name", nullable = false)
 private String lastName;
@Column(name = "email", unique = true, nullable = false)
  message ="{invalid.email}")
```

Оголошується анотація для замаплення зв’язку ManyToMany:

```
  @ManyToMany(cascade = CascadeType.MERGE)
```

За допомогою анотації @JoinTable у класі Customer надається назва таблиці об’єднання(customers_roles), а також зовнішні ключі з анотаціями @JoinColumn. Атрибут joinColumn підключатиметься до customer, а inverseJoinColumn – до role

```
  @JoinTable(name = "customers_roles",
 joinColumns = @JoinColumn(name = "customer_id"),
inverseJoinColumns = @JoinColumn(name = "role_id"))
```

Створюється колеція Role, що реалізовує зв'язок OneToMany:

```
private Collection<Role> roles;
@OneToMany(mappedBy = "client", cascade = CascadeType.REMOVE)
```

Вказується пряма частина посилання за допомогою анотації JsonManagedReference, яка серилізується:

```
@JsonManagedReference
 private Collection<AttendanceSchedule> customerSchedules;
@OneToMany(mappedBy = "worker", cascade = CascadeType.REMOVE)
@JsonManagedReference
private Collection<AttendanceSchedule> workerSchedulesSchedules;
 private boolean isAdmin;
```

Потрібно оголосити інтерфейс репозиторію наступним чином:

```
@Repository
public interface CustomerRepository extends JpaRepository<Customer, Long> {
Customer findByEmail (String email);
```

Далі ми можемо використовувати цей репозиторій у контролері Spring MVC або бізнес-класі наступним чином:

```
@Service
@RequiredArgsConstructor
public class CustomerServiceImpl implements CustomerService, UserDetailsService {
 private final CustomerRepository clientRepository;
 private final RoleRepository roleRepository;
 @Override
 public List<Customer> getAllCustomers() {
return clientRepository.findAll();
```

3.4 Додавання шаблонізатору Java XML / XHTML / HTML5 в Controller

Реалізація шарів DAO, які забезпечують функціональність CRUD на об’єктах JPA, може бути повторюваним, трудомістким завданням, якого ми хочемо уникнути в більшості випадків. На щастя, Spring Boot дозволяє легко створювати програми CRUD за допомогою рівня стандартних сховищ CRUD на основі JPA.

У цьому випадку ми будемо покладатися на spring-boot-starter-parent для простого керування залежностями, керування версіями та налаштування плагінів. В результаті нам не потрібно буде вказувати версії залежностей проекту в нашому файлі pom.xml, за винятком заміни версії Java:

```
<dependencies>
 <dependency>
  <groupId> org.springframework.boot </groupId>
  <artifactId> spring-boot-starter-thymeleaf </artifactId>
  <version> 2.4.3 </version>
  </dependency>
```

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<version>2.4.3</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<version>2.4.3</version>
<scope>runtime</scope>
<optional>true</optional>
</dependency>
```

У підході Spring до створення веб-сайтів HTTP-запити обробляються контролером. Ви можете легко визначити контролер за анотацією @Controller. У наступному прикладі CustomerController обробляє запити GET для /greeting, повертаючи ім’я представлення (у даному випадку greeting). Представлення відповідає за відтворення вмісту HTML. У наступному списку(з \masterspreparation\controller\CustomerController.java) показано контролер:

```
@Controller
@RequestMapping(value = "/customers")
@RequiredArgsConstructor
public class CustomerController {
private final CustomerService customerService;
@GetMapping()
```

Оголошується метод, що повертає список всіх клієнтів:

```
public String listCustomers(Model model) {
 model.addAttribute("customers", customerService.getAllCustomers());
return "customers";
```

Метод, що приймає CustomerDto обєкт і повертає html-сторінку:

```
public String createCustomerForm(Model model) {
CustomerDto customer = new CustomerDto();
 model.addAttribute("customer", customer);
return "create_customer";
```

Метод, що обробляє запити HTTP POST, та перенапрямляє на конкретну сторінку:

```
 @PostMapping()
 public String saveCustomer(@ModelAttribute("customer") CustomerDto customer) {
customerService.saveCustomer(customer);
 return "redirect:/customers";
```

Метод, що обробляє запити HTTP Get, та перенапрямляє на конкретну сторінку:

```
@GetMapping("/edit/{id}")
 public String editCustomerForm(@PathVariable Long id, @NotNull Model model) {
 model.addAttribute("customer", customerService.getCustomerById(id));
 return "edit_customer";
@PostMapping("/{id}")
```

Отримує клієнта з бази даних за id:

```
Customer existingCustomer = customerService.getCustomerById(id);
existingCustomer.setId(id);
existingCustomer.setFirstName(customer.getFirstName());
existingCustomer.setLastName(customer.getLastName());
existingCustomer.setEmail(customer.getEmail());
```

Зберігає модифікований об’єкт студента:

```
customerService.updateCustomer(existingCustomer);
return "redirect:/customers";
```

Метод обробки для обробки запиту студента на видалення:

```
@GetMapping("/{id}")
public String deleteCustomer(@PathVariable Long id)
customerService.deleteCustomerById(id);
 return "redirect:/customers";
```

Анотація @GetMapping гарантує, що HTTP-запити GET до /customer відображаються на метод listCustomers().@RequestParam пов’язує значення імені параметра рядка запиту з параметром name методу customer().

Реалізація тіла методу покладається на технологію перегляду (у даному випадку Thymeleaf) для виконання відтворення HTML на стороні сервера. Thymeleaf аналізує шаблон customer.html і оцінює вираз th:text, щоб відобразити значення параметра ${name}, яке було встановлено в контролері. У наступному списку (з src/main/resources/templates/customers.html) показано шаблон customers.html:

```
 <div class="collapse navbar-collapse" id="collapsibleNavbar">
 <ul class="navbar-nav">
 <li class="nav-item">
 <a class="nav-link" th:href="@{/customers}">Client Management</a>
 </li>
 <li class="nav-item">
 <a class="nav-link" th:href="@{/attendance/date}">By Date</a>
 </li>
 <li class="nav-item">
 <a class="nav-link" th:href="@{/attendance}">Attendance Schedule</a>
 <div class ="container">
 <div class = "row">
<h1> List Clients </h1>
```

3.5 Додавання фреймворку для автоматизації структури проекту

Apache Maven — це інструмент для керування проектами програмного забезпечення та його розуміння. На основі концепції об’єктної моделі проекту

(POM), Maven може керувати збіркою проекту, звітами та документацією з центральної частини інформації.

Проекти Maven визначаються за допомогою файлу XML з іменем pom.xml. Серед іншого, цей файл містить назву проекту, версію та залежності, які він має від зовнішніх бібліотек. Необхідно створити файл pom.xml в корені проекту (тобто необхідно створити його поруч із папкою src) і надати йому такий вміст:

```
<?xml version="1.0" encoding="UTF-8"?>
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
<groupId>net.javaguides</groupId>
<artifactId>masters-preparation</artifactId>
 <version>0.0.1-SNAPSHOT</version>
<name>masters-preparation</name>
 <description>Client Management System using Spring Boot and Thymeleaf</description>
 <properties>
 <java.version>14</java.version>
 </properties>
<dependencies>
 <dependency>
```

У Maven кожен проект має пару groupId, artifactId. Щоб уникнути конфлікту імен, groupId — найменування організації чи підрозділи і зазвичай діють такі самі правила, як і за іменуванні пакетів в Java —записують доменне ім'я організації чи сайту проекту, artifactId — назва проекту. Всередині тега version зберігається версія проекту. Трійкою groupId, artifactId, version можна однозначно ідентифікувати jar-файл програми або бібліотеки. Якщо стан коду для проекту не зафіксований, то в кінці до імені версії додається "SNAPSHOT", що означає, що версія в розробці та результуючий jar файл може змінюватися.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для гарантування якості розробленого програмного продукту проводиться тестування. Тестування можна поділити на два підтипа, функціональне та нефункціональне. На рисунку 4.1 показано види тестування програмного забезпечення.

Рисунок 4.1 — Види тестування програмного забезпечення.

Ручне тестування проводиться особисто, клацнувши програму або взаємодіючи з програмним забезпеченням та API за допомогою відповідних інструментів. Автоматизоване тестування виконуються машиною, яка виконує заздалегідь написаний тестовий сценарій. Ці тести можуть сильно відрізнятися за складністю, від перевірки одного методу в класі до переконання, що виконання послідовності складних дій в інтерфейсі користувача призводить до однакових результатів. Автоматичне тестування є ключовим компонентом безперервної інтеграції та безперервної доставки, і це чудовий спосіб масштабувати процес QA, коли ви додаєте нові функції до своєї програми. Модульні тести дуже низького рівня, вони полягають у тестуванні окремих методів і функцій класів, компонентів або модулів, які використовуються вашим програмним забезпеченням. Модульні

тести, як правило, досить дешеві для автоматизації і можуть виконуватися дуже швидко за допомогою сервера безперервної інтеграції.

Інтеграційні тести підтверджують, що різні модулі або служби, які використовує ваша програма, добре працюють разом. Наприклад, це може бути тестування взаємодії з базою даних або переконання, що мікросервіси працюють разом, як очікувалося. Ці типи тестів є дорожчими у виконанні, оскільки вони вимагають, щоб кілька частин програми були запущені та запущені.

Функціональне тестування стосується перевірки того, як функціонує продукт, виконання функціонального тесту передбачає перевірку та тестування кожної функціональності програмного забезпечення, щоб переконатися, що ви отримуєте очікувані результати.

Нефункціональне тестування стосується тестування нефункціональних аспектів програмного забезпечення, таких як зручність, продуктивність, безпека, надійність тощо. Цей тест проводиться після проведеного функціонального тесту. Нефункціональний тест не зосереджується на тому, чи працює програмне забезпечення чи ні, він фокусується на тому, наскільки добре воно працює.

У даній роботі будуть виконуватися як функціональне так і нефункціональне тестування, а саме юніт, компонент, димове, тестування установки та тестування зручності використання та тестування графічного інтерфейсу.

4.1. Тестування на базі mock-mvc.

За допомогою тестового фреймворку Spring MVC, що надає клас MockMvc можливо здійснити тестування контролерів шляхом ініціювання контейнера сервлетів. Spring MVC дає можливість виконувати HTTP-запити до контролера без його реального запуску. Для налаштування Mock MVC використовується MockMVCBuilders. Цей клас має два статичний метод – standaloneSetup() і webAppContextSetup(). Метод standaloneSetup() будує Mock MVC для обслуговування одного чи декілька вручну створених і налаштованих контролерів. Метод webAppContextSetup () будує Mock MVC, використовуючи Spring контекс, який містить один чи більше налаштованих контролерів.

Для створення тестів необхідно додати залежність spring-boot-starter-test до pom.xml. Ця залежність включає всі необхідні залежності для створення та виконання тестів.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<version>2.4.3</version>

<scope>test</scope>

/dependency>

Для того щоб перевірити, що контролер працює належним чином, коли буде знайдено клієнта.

public void test1() throws Exception {

when(customerRepository.findAll()).thenReturn(new ArrayList<>());

mockMvc.perform(get("/customers"))

.andExpect(status().is2xxSuccessful());

verify(customerRepository, times(1)).findAll();

Налаштування фіктивного об’єкта для повернення створеного об’єкта ArrayList, виконання запиту GET до URL-адреси '/customers',перевірка чи повернуто код статусу HTTP 200, перевірка чи викликається метод хоча б один раз через mockito verify.

Тест для перевірки успішного збереження клієнта

public void saveCustomer_okTest() {

CustomerDto customerDto = new CustomerDto("Test", "Test", "Test@gmail.com", "1234", true);

when(clientRepository.save(refEq(getCustomer(),"password"))).thenReturn ());

when(roleRepository.findByName("ROLE_ADMIN")).thenReturn(new Role("ROLE_ADMIN"));

Customer actual = customerServiceTest.saveCustomer(customerDto);

verify(clientRepository, times(1)).save(any());

verify(roleRepository, times(1)).findByName(any());

assertEquals("Test", actual.getFirstName());

Тест для перевірки помилки збереження клієнта, приймає на вхід CustomerDto об’єкт, та порівнює із ти, що є у базі.

public void saveCustomer_errorTest() {

CustomerDto customerDto = new CustomerDto("", "", "", "", true);

Тест для перевірки отримання клієнта, шукає отриманого клієнта у базі і порівнює із значенням, потім повертає статус 200.

public void getAllCustomer_okTest() {

when(clientRepository.findAll()).thenReturn(new ArrayList<>());

List<Customer> actual = customerServiceTest.getAllCustomers();

assertTrue(actual.isEmpty());

Тест для перевірки отримання клієнта

public void getCustomerById() {

when(clientRepository.findById(1L)).thenReturn(Optional.of(getCustomer()));

Customer actual = customerServiceTest.getCustomerById(1L);

assertEquals("Test", actual.getFirstName());

Тест для перевірки успішного видалення клієнта із бази даних, перевіряє по унікальному ідентифікатору.

public void deleteCustomer_okTest() {

Customer customer = getCustomer();

customer.setAdmin(false);

when(clientRepository.findById(1L)).thenReturn(Optional.of(customer));

customerServiceTest.deleteCustomerById(1L);

verify(clientRepository, times(1)).findById(any());

verify(clientRepository, times(1)).deleteById(any());

Тест для перевірки видалення клієнта шукає отриманого клієнта у базі і порівнює із значенням, потім повертає статус 400, що означає, що видалення клієнта закінчилося із помилкою.

public void deleteCustomer_errorTest() {

when(clientRepository.findById(1L)).thenReturn(Optional.of(getCustomer()));

customerServiceTest.deleteCustomerById(1L);

verify(clientRepository, times(1)).findById(any());

verify(clientRepository, times(0)).deleteById(any());

Тест для перевірки зберігання успішного запису клієнта, що був тільки створений.

public void saveAttendance_okTest() {

LocalDateTime localDateTime = LocalDateTime.now();

Customer customer = new Customer();

customer.setEmail("1@gmail.com");

Customer worker = new Customer();

worker.setEmail("2@gmail.com");

when(customerService.findByEmail("1@gmail.com")).thenReturn(customer)

when(attendanceRepository.save(getAttendanceSchedule(localDateTime))).thenReturn(getAttendanceSchedule(localDateTime));

verify(customerRepository, times(2)).findByEmail(anyString());

verify(attendanceRepository, times(1)).save(any());

Тест для перевірки роботи контролера записів клієнтів у вибраний термін

public void getAttendanceScheduleForDate_okTest() {

when(attendanceRepository.findAll()).thenReturn(getListOfAttendance());

assertEquals(3, actual.size());

Тест для перевірки роботи контролера записів клієнтів сьогодні

public void getAttendanceScheduleForToday_okTest() {

when(attendanceRepository.findAll()).thenReturn(getListOfAttendance());

attendanceService.getAttendanceScheduleForToday("worker@gmail.com");

assertEquals(1, actual.size());

4.2 Розробка інструкції користувача

Після запуску програми перед користувачем з’явиться вікно авторизації у систему, що представлено на рисунку 4.2.

Після успішної авторизації, відкриється головна сторінка, що містить список клієнтів із їхніми даними. Головна сторінка представлена на рисунку 4.3. Якщо клієнт чи працівник здійснює авторизацію із неправильними даними, на екрані з’явиться помилка авторизації.

Основними функціями, що можуть здійснюватися є можливість створення нового клієнта із детальною інформацією про нього, можливість зміни його персональних даних, та видалення клієнта, у разі необхідності, що представлено на рисунку 4.4

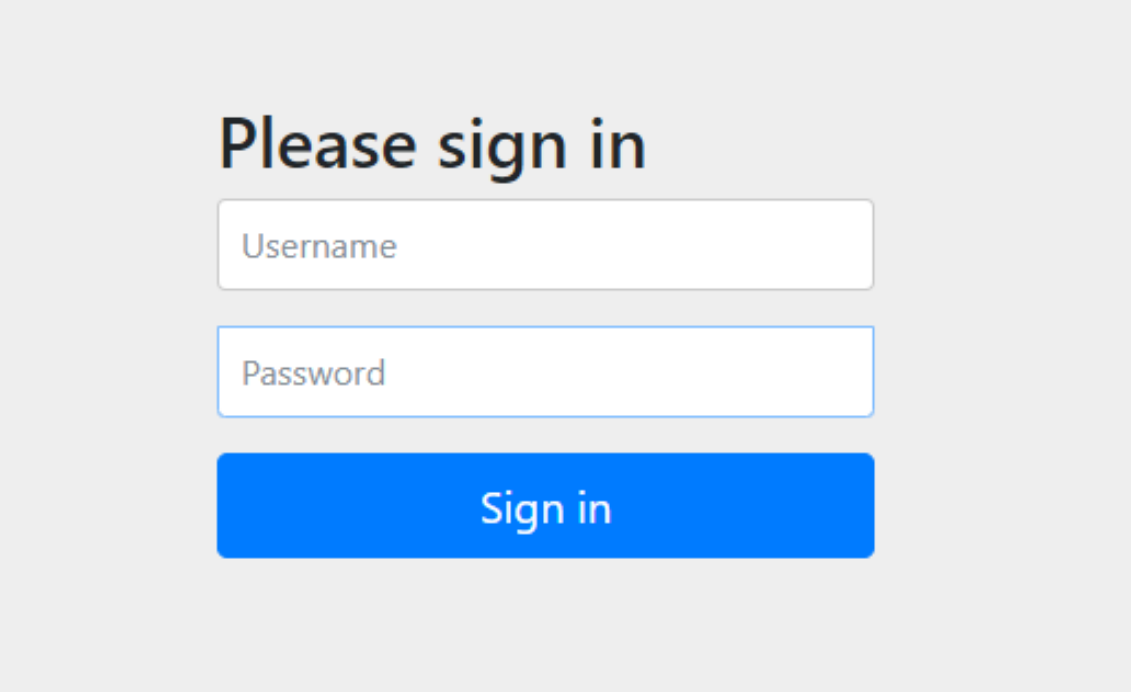

Рисунок 4.2 — Вікно авторизації у систему

Також перейшовши на вкладку “Attendance Schedule”, клієнт та працівник можуть додати новий запис та побачити свій графік вже існуючих записів на сьогодні або інший вибраний термін.

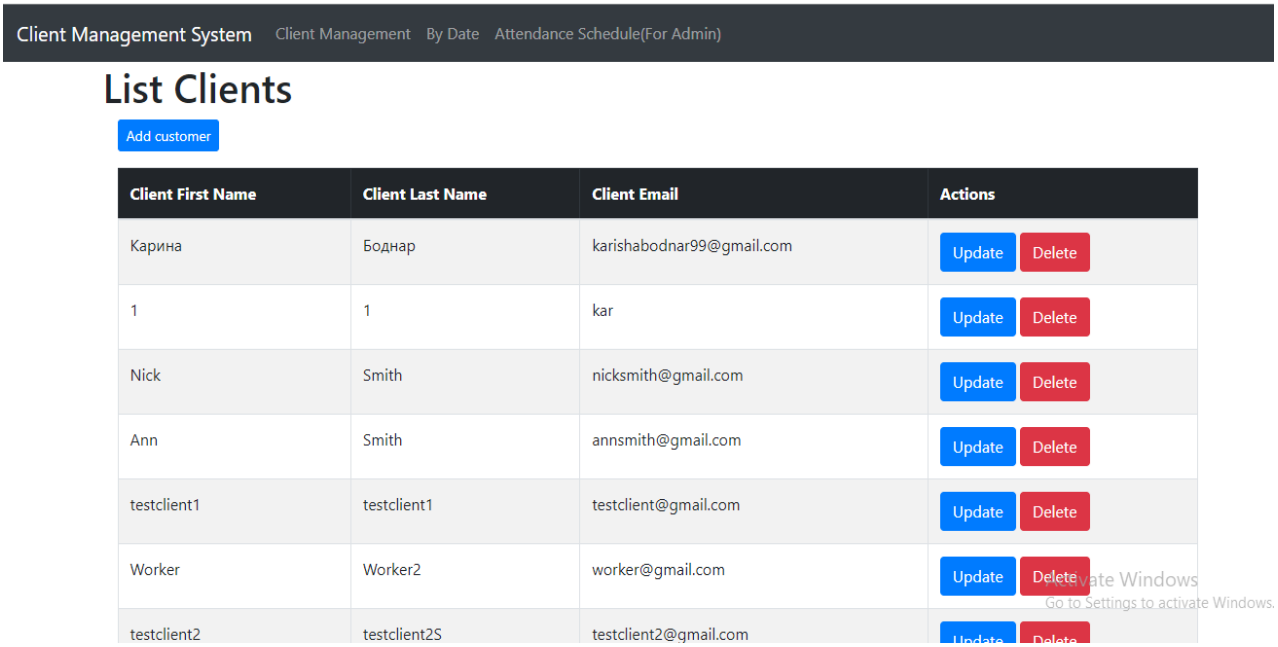

Рисунок 4.3 — Головна сторінка системи

На рисунку 4.4 представлено можливість створення нового клієнта із вказанням його імені, прізвища, особистого облікового запису, паролю та вказання його ролі(чи адмін чи користувач).

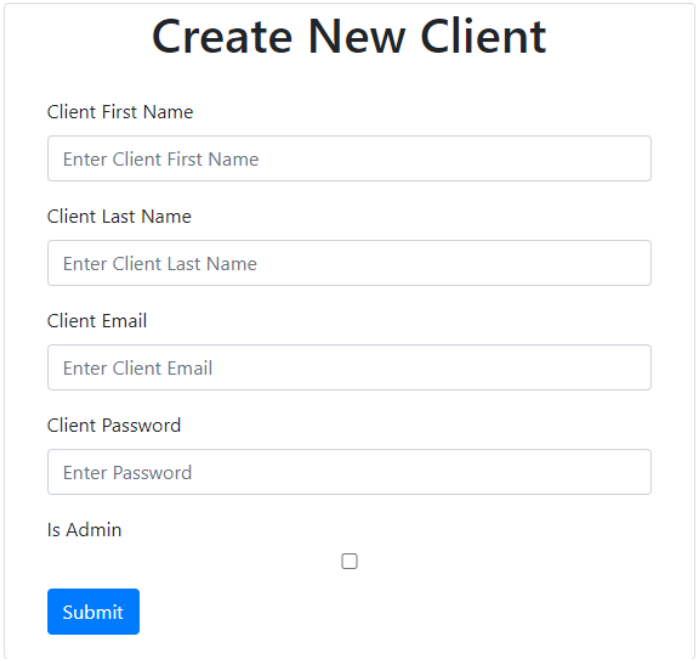

Рисунок 4.4 — Створення нового клієнта

Задля забезпечення комфорту користувачів, у програмі реалізовано можливість розсилання нагадувань про найближчий запис із використанням електронної

пошти. На рисунку 4.5 продемонстровано нагадування, що кожний день розсилається користувачам.

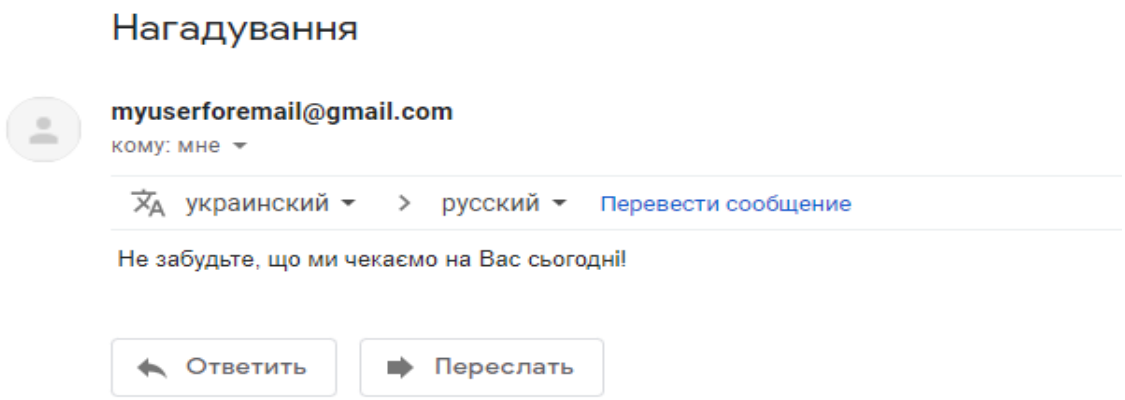

Рисунок 4.5— Нагадування про найближчий запис

Задля забезпечення комфорту працівників, у програмі реалізовано можливість моніторингу фінансів. На рисунку 4.6 продемонстровано моніторинг фінансів.

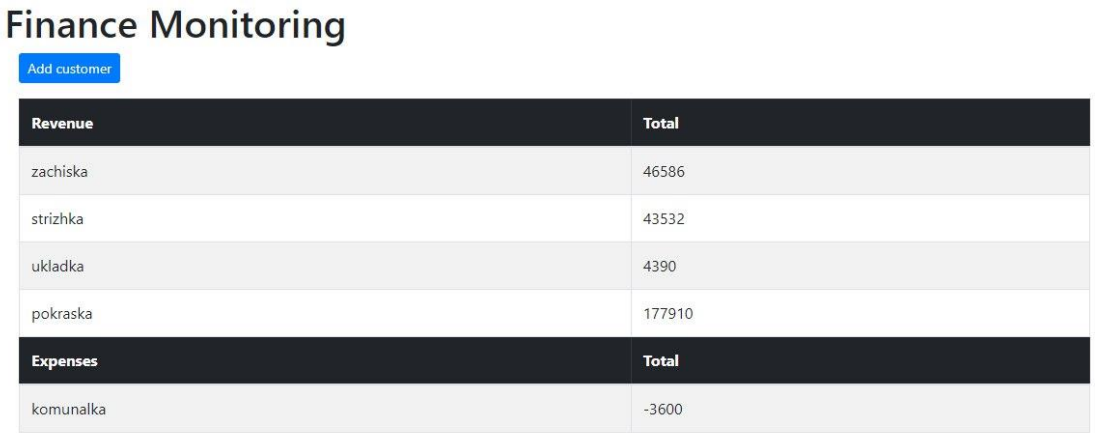

Рисунок 4.6— Моніторинг фінансів

Отож, для виконання розробки програмного продукту було обрано мову програмування Java із використання фреймворку Spring, через кросплатформеність, багатофункціональність, високу швидкодію та зручність підтримки.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

— проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

— розраховано витрати на здійснення науково-технічної розробки;

— розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері

послуг» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [27].

Таблиця 5.1 — Критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
1	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Закінчення таблиці 5.1.

1	2	3	4	5	6
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	5
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	2	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	36	37	38
Середньоарифметична сума балів $СБ_c$	37,0		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [29].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» становить 37,0 бала, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою [29]:

$$q_i = \frac{P_i}{P_{базі}} \quad (5.1)$$

де q_i — одиничний параметричний індекс, розрахований за i-м параметром;

P_i — значення i-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 — пристрій відповідає нормам і стандартам; 0 — не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [30]:

$$I_{НП} = \prod_{i=1}^{n} q_i \quad (5.2)$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за i-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірю-вання	Аналог	Розроблювана система	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість доступних показників в запиті	шт.	15	24	1,6	0,1
Кількість аналітичних функцій	шт.	32	54	1,68	0,2
Кількість аналітичних модулів	шт.	8	11	1,37	0,25
Швидкість пошуку та видачі запиту (тест)	мс	0,45	0,45	1	0,3
Швидкість та легкість доступу (дружність інтерфейсу)	бали	6	8	1,33	0,15
Експлуатаційні витрати (підтримка інформаційної системи)	грн	400	310	0,77	0,5
Вартість підписки (доступу до ресурсу)	грн	300	250	0,83	0,5

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$. Значення групового параметричного індексу за технічними

параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [30]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \qquad (5.3)$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник i-го параметра;

α_i — вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 5.4.

I_{mn} = 1,6·0,1+1,68·0,2+1,37·0,25+1·0,3+1,33·0,15 = 1,34.

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [30]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \qquad (5.4)$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$I_{ЕП}$ =0,77 · 0,5 + 0,83 · 0,5 = 0,80.

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [29]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}} \qquad (5.5)$$

$$КІНТ = 1 \cdot 1,34 / 0,80 = 1,67.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників (Зо) розраховуємо у відповідності до посадових окладів працівників, за формулою [Козловський, Лесько, Кавецький]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p} \qquad (5.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

де k — кількість посад дослідників залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=21 дні.

$З_o$ = 13240,00 · 37 / 21 = 23327,62 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	13240,00	630,48	37	23327,62
Аналітик з розробки інформаційних систем	13100,00	623,81	11	6861,90
Інженер-програміст 1-ї категрії	12750,00	607,14	26	15785,71
Консультант (менеджер-адміністратор сфери послуг)	13200,00	628,57	5	3142,86
Технік	7400,00	352,38	11	3876,19
Всього				52994,29

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників (Зр) за відповідними найменуваннями робіт НДР на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \qquad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду Сі можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \quad (5.8)$$

де M_M — розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=2379,00 грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду [34];

K_c — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати;

T_p — середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ — тривалість зміни, год.

$C_1 = 2379,00 \cdot 1,10 \cdot 1,65 / (21 \cdot 8) = 25,70$ грн.

$З_{р1} = 25,70 \cdot 6,00 = 154,21$ грн.

Проведені розрахунки занесемо до таблиці 5.6.

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати, приймемо 11%.

$З_{дод} = (52994,29 + 2291,54) \cdot 11 / 100\% = 6081,44$ грн.

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обчислювального обладнання для проведення досліджень	6,00	2	1,10	25,70	154,21
Підготовка робочого місця розробника програмного забезпечення	4,50	3	1,35	31,54	141,94
Інсталяція програми	5,00	4	1,50	35,05	175,24
Введення дослідних баз даних інформаційного ресурсу	12,00	3	1,35	31,54	378,52
Ведення кодів модульних блоків інформаційної системи	18,00	3	1,35	31,54	567,77
Компіляція програмних модулів інформаційної системи підтримки підприємств малого бізнесу у сфері послуг	7,00	5	1,70	39,72	278,05
Налагодження блоків інформаційної системи	10,00	6	2,00	46,73	467,30
Тестування інформаційної системи	5,00	2	1,10	25,70	128,51
Всього					2291,54

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \qquad (5.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (52994,29 + 2291,54 + 6081,44) · 22 / 100% = 13500,80 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj} , \qquad (5.11)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1$ = 4,00 · 116,00 · 1,11 - 0,000 · 0,00 = 515,04 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Багатофункціональний білий офісний папір FAXE-500 A4	116,00	4,00	0,000	0,00	515,04
Папір для записів FAXE 70 A5-250	47,00	5,00	0,000	0,00	260,85
Органайзер офісний OFFICE 100	220,00	2,00	0,000	0,00	488,40
Набір офісний DATUM X-2	205,00	3,00	0,000	0,00	682,65
Картридж для принтера HP-2100	1010,00	2,00	0,000	0,00	2242,20
Диск оптичний OPTIMA CD	15,50	5,00	0,000	0,00	86,03
Flesh-пам'ять GOODRAM 64 C10A	420,00	2,00	0,000	0,00	932,40
Всього					5207,57

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» відсутні.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \quad (5.12)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 35680,00 \cdot 1 \cdot 1,11 = 39604,80$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Сервер - Компютер Expert PC Balance (I91F8H1S115E429)	1	35680,00	39604,80
Всього			39604,80

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.13)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 78,75 \cdot 1 \cdot 1,11 = 87,41$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Доступ (підписка) для вивчення аналогу "Інформаційна система Capsule"	1	78,75	87,41
Доступ (підписка) для вивчення аналогу "Інформаційна система Scoro"	1	325,00	360,75
Доступ (підписка) для вивчення аналогу "Інформаційна система Bitrix24"	1	650,00	721,50
Емулятор серверної інтернет-платформи для моделювання поведінки інформаційного ресурсу	1	5800,00	6438,00
Всього			7607,66

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.14)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (25125{,}00 \cdot 2) / (2 \cdot 12) = 2093{,}75$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	25125,00	2	2	2093,75
Обчислювально-графічна система програмної розробки	22498,00	2	2	1874,83
Ноутбук HP Laptop 15s-eq2037ua Natural Silver (422G7EA)	32780,00	2	2	2731,67
1	2	3	4	5

Робоче місце розробника програмного забезпечення	9580,00	5	2	319,33
Пристрій виводу інформації HP-2100	8765,00	4	2	365,21
Оргтехніка	8925,00	4	2	371,88
Приміщення лабораторії	286000,00	20	2	2383,33
ОС Windows 11	7400,00	3	2	411,11
Прикладний пакет Microsoft Office 2019	6700,00	3	2	372,22
Прикладний пакет розробки інформаційних систем (мови Java, C#, C++, Perl, PHP)	7676,00	3	2	426,44
Всього				11349,78

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.15)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 4,45 грн;

$К_{вп і}$ – коефіцієнт, що враховує використання потужності, $К_{вп і} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$В_е = 0,25 \cdot 244,0 \cdot 4,45 \cdot 0,95 / 0,97 = 271,45$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,25	244,0	271,45
Обчислювально-графічна система програмної розробки	0,40	244,0	434,32
Ноутбук HP Laptop 15s-eq2037ua Natural Silver (422G7EA)	0,05	220,0	48,95
Робоче місце розробника програмного забезпечення	0,20	200,0	178,00
Пристрій виводу інформації HP-2100	0,60	12,0	32,04
Оргтехніка	0,65	10,0	28,93
Сервер - Компютер Expert PC Balance (I91F8H1S115E429)	0,45	220,0	440,55
Всього			1434,24

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням

досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.16)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (52994,29 + 2291,54) · 20 / 100% = 11057,17 грн.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

5.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (52994,29 + 2291,54) · 50 / 100% = 27642,91 грн.

5.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.18)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв} = (52994,29 + 2291,54) \cdot 100 / 100\% = 55285,83$ *грн.*

Витрати на проведення науково-дослідної роботи на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.19)$$

$B_{заг}$= *52994,29 +2291,54 +6081,44 +13500,79861 +5207,57 +0,00 +39604,80 + 7607,66 + 11349,78 +1434,24 +11057,17 +0,00 +27642,91 +55285,83 = 234058,01 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.20)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 234058,01 / 0,95 = 246376,85 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	10500	12000	12000	10000

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 35000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 200,00 грн;

$\pm\Delta Ц_o$ – зміна вартості послуги від впровадження результатів, приймемо 50,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [29]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.21)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(50,00·35000,00+250,00·10500)·0,83·0,4·(1-0,18/100%)=1191050,00 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(50,00·35000,00+250,00·22500)·0,83·0,4·(1-0,18/100%)=2007770,00 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(50,00·35000,00+250,00·34500)·0,83·0,4·(1-0,18/100%)=2824490,00 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(50,00·35000,00+250,00·44500)·0,83·0,4·(1-0,18/100%)=3505090,00 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.22)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,09;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 1191050,00/(1+0,09)^1+2007770,00/(1+0,09)^2+2824490,00/(1+0,09)^3+$$
$$+3505090,00/(1+0,09)^4=1092706,42+1689899,84+2181024,52+2483094,12=$$
$$=7446724,90 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.23)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 246376,85 грн.

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 246376,85 = 492753,70 \text{ грн.}$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.24)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 7446724,90 грн;

PV – теперішня вартість початкових інвестицій, 492753,70 грн.

$E_{абс} = ПП - PV$ = 7446724,90 - 492753,70 = 6953971,20 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.25)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 6953971,20 грн;

PV – теперішня вартість початкових інвестицій, 492753,70 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+6953971,20/492753,70)$^{1/4}$= 0,97.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$

$$\tau_{мін} = d + f, \quad (5.26)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,97 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.27}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,97 = 1,03 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг» становить 37,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,67 рази.

Також термін окупності становить 1,03 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг».

ВИСНОВКИ

У першому розділі було розглянуто роль та цінність інформаційної системи для підтримки малого бізнесу. Також був проведений огляд уже існуючих інформаційних систем підтримки малого бізнесу, та проаналізовано їх основні переваги та недоліки, що продемонструвало обмеженість їх застосування для таких задач, як задання цілей клієнта, розсилання повідомлення-нагадування в особистий електронний кабінет клієнта та аналізу (урахування) ступеня важливості клієнта.

У другому розділі проведено аналітичний огляд технологій, що будуть використовуватися при розробці інформаційної системи. Для практичної реалізації розробки використано мову Java, що дозволило підвищити швидкодію роботи системи. Важливим фактором при обрані мови також став той факт, що вона сумісна з фреймворком Spring.

У третьому розділі було показано налаштування Spring Security, що забезпечує функції автентифікації та авторизації додатків, що написані мовою Java. Було налаштовано Spring Boot Mail Server, що дало можливість розсилати повідомлення-нагадування клієнтам. Додано шаблонізатор Java XML / XHTML / HTML5 в Controller для реалізації представлення бізнес-логіки додатку. Було додано фреймворк для автоматизації структури проекту.

У четвертому розділі було проведено тестування на базі mock-mvc, для написання component та unit тестів, а також було проведено UI-тестування розробленого програмного продукту, що дозволило перевірити працездатність та гарантувати безпеку. Була створена інструкція користувача, яка допоможе людині, незнайомій з програмою, швидко розібратись з усіма її особливостями та зрозуміти структуру програми та принципи роботи із нею.

У п’ятому розділі було проведено обрахунки і приведено доцільність реалізації проекту за темою магістерської роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Micronaut Faremework Documentation, [Електронний ресурс]. – Режим доступу: https://micronaut.io/docs/
2. Helidon Faremework Documentation, [Електронний ресурс]. – Режим доступу: https://helidon.io/
3. Spring Faremework Documentation, [Електронний ресурс]. – Режим доступу: http://docs.spring.io/
4. Koin Faremework Documentation, [Електронний ресурс]. – Режим доступу: https://insert-koin.io/
5. Хорстманн, К. С., Корнелл, Г. Библиотека профессионала. Java 2 : Том 1.
6. Основы. 8-е изд. М. : Вильямс, 2013. 816 с.
7. Шевчук І. Б. Інформаційні технології в регіональній економіці: теорія і практика впровадження та використання : монографія. Львів : Видавництво ННВК "АТБ", 2018. 448 с.
8. Шилдт Г. Полный справочник по Java SE 6. М.: Вильямс, 2010. 1040 с.
9. Эккель Б. Философия Java. 4-е изд. СПб. : Питер, 2011. 640 с.
10. Васильев А.Н. Самоучитель Java с примерами и программами. СПб.: Наука и Техника, 2011. 352 с.
11. Ковалюк Т.В. Основи програмування. / Ковалюк Т.В. Київ: BHV Киев, 2005. 400 с.
12. Николайчук Я. М. Проектування спеціалізованих комп'ютерних систем навч. посібник / Я. М. Николайчук, Н. Я. Возна, І. Р. Пітух. - Тернопіль ТзОВ "Терно-граф", 2010. - 392 с.
13. Хорстманн К.С., Корнелл Г. Java 2. Библиотека профессионала. Т. 2,
14. Тонкости программирования. М.: Вильямс, 2010. 992 с.
15. Шилдт Г.6. Java: руководство для начинающих. М.: Вильямс, 2008. 720 с.
16. Java Учебник для начинающих программистов : [Електрон. ресурс]. -
17. Режим доступу: http://proglang.su/java
18. Популярные технологии программирования в 2017 году : [Електрон. ресурс]. - Режим доступу: https://vc.ru/dev/21483-what-language-2017

19. Программирование на Java: [Електрон. ресурс]. - Режим доступу: https://www.intuit.ru/studies/courses/16/16/info
20. Рагулин П.Г. Информационные технологии: электронный учебник :[Електрон. ресурс]. - Режим доступу: http://window.edu.ru/catalog/pdf2txt/007/41007/18312 p_page=4
21. Теоретические основы технологии программирования: [Електрон. ресурс].- Режим доступу: http://bourabai.kz/alg/technology.htm
22. Технология программирования : [Електрон. ресурс]. - Режим доступу: https://studref.com/441961/informatika/tehnologiya_programmirovaniya
23. Топ-5 полезных видеокурсов по Java : [Електрон. ресурс]. - Режим доступу: https://javarush.ru/groups/posts/528-top-5-poleznihkh-videokursov-po-javachastjh-1--
24. Уроки по основам языка программирования JAVA для начинающих : [Електрон. ресурс]. - Режим доступу: https://www.fandroid.info/tutorial-poosnovam-yazyka-programmirovaniya-java-dlya-nachinayushhih/
25. Энциклопедия языков программирования : [Електрон. ресурс]. – Режим доступу: http://progopedia.ru/
26. Язык программирования Java и среда NetBeans : [Електрон. ресурс]. -Режим доступу: https://www.intuit.ru/studies/courses/569/425/inf
27. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
28. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.
29. Васильєв А.Н. Об’єктно-орієнтоване програмування. Java; Навчальний посібник. – СПб.: Пітер, 2011. – 400 с.
30. Електронна бібліотека. Wikipedia. [Електронний ресурс] : Веб проргамування на Java. – Режим доступу: https://uk.wikibooks.org/wiki/Веб-програмування_на_Java

31. Електронна бібліотека. Wikipedia. [Електронний ресурс] : Освоюємо Java – Режим доступу: https://uk.wikibooks.org/wiki/Освоюємо_Java
32. Гіберт Ш.В. Java. Полное руководство, 8-е издание; Посібник програміста. – СПб.: Київ, 2013. – 175 с.
33. Java посібник. [Електронний ресурс] : Изучаем Java. Обучающие курси. Java-самоучитель: программирование на языке Java. Режим доступу – http://java-study.ru/samouchitel
34. Брюс Эккель — Философия Java [Електронний ресурс]. — 2019. — Режимдоступу: https://bit.ly/2JbSgHe.
35. Katty Sierra — Head first Java, 2nd edition [Електронний ресурс]. — 2005. — Режим доступу: https://www.amazon.com/dp/0596009208.

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____

(наук. ст., вч. зв., ініц. та прізв.) (підпис)

"____" _____20__р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської роботи

Інформаційна система підтримки підприємств малого бізнесу у сфері послуг

08-023.МКР.001.00.000 ТЗ

Науковий керівник: _____ Захарченко С. М.

(підпис)

студентка групи _____ Боднар К. О.

(підпис)

Вінниця ВНТУ 2021

1 Область застосування

Інформаційна система підтримки підприємств малого бізнесу у сфері послуг може використовуватись для реєстрування та супроводжування клієнтів малого бізнесу. У поєднані із розвертанням програмний продукт може бути використаний для доступу у будь-який час та з будь-якого куточку світу.

2 Підстави для розробки

Підставою для розробки магістерської кваліфікаційної роботи є наказ про затвердження тем магістерських робіт.

3 Призначення розробки

— провести аналіз існуючих на ринку систем підтримки підприємств;

— дослідити сучасні інструменти які дозволили б реалізувати на практиці дану інформаційну систему;

— розробити програмне забезпечення;

— провести тестування розробленого програмного забезпечення.

4 Вимоги до виконання МКР

Необхідно реалізувати клієнт-серверну архітектуру інформаційної системи. Також потрібно враховувати можливу затримку підключення до бази даних.

5 Матеріали, що подаються до захисту

Пояснювальна записка МКР, графічні та ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до МКР українською та іноземною мовами, відповідність оформлення МКР діючим вимогам.

6 Стадії та етапи розробки

Стадії та етапи розробки наведені у таблиці А.1

Таблиця А.1 — Етапи виконання роботи

Етап	Назва етапу	Початок	Кінець	Очікувані результати
1	Інформаційний пошук та огляд літературних джерел.	01.07.21	01.09.21	Розділи 1, 2, 3 та 4.
2	Дослідження сучасних інструментів, які дозволили б реалізувати на практиці дану інформаційну систему. Вивчення фреймворка Spring.	01.09.21	05.10.21	Чернетки матеріалів.
3	Розробка програмних засобів.	05.10.21	01.11.21	Програмний продукт
4	Підготовка матеріалів пояснювальної записки.	01.11.20	8.12.21	Пояснювальна записка.

Технічне завдання до виконання прийняла _____

ДОДАТОК Б

Лістинг програми

```
/**
 * @author KarinaBodnar
 */
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final CustomerServiceImpl customerService;
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .anyRequest().authenticated().and()
                .formLogin()
                .defaultSuccessUrl("/customers")
                .and()
                .logout().logoutSuccessUrl("/");
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance();
    }
    @Bean
    public DaoAuthenticationProvider daoAuthenticationProvider() {
        DaoAuthenticationProvider authenticationProvider = new
DaoAuthenticationProvider();
        authenticationProvider.setPasswordEncoder(passwordEncoder());
        authenticationProvider.setUserDetailsService(customerService);
        return authenticationProvider;
```

```
    }
}
@Controller
@RequiredArgsConstructor
public class AttendanceController {
    private final AttendanceService attendanceService;
    @GetMapping("/attendance")
    public String listCustomers(Model model, Principal principal) {
        model.addAttribute("attendances",
attendanceService.getAttendanceScheduleForToday(principal.getName()));
        return "attendances";
    }
    @GetMapping("/attendance/date")
    public String listCustomersByDate(Model model) {
        ScheduleRequestDto requestDto = new ScheduleRequestDto();
        model.addAttribute("requestDto", requestDto);
        return "create_request-by-date";
    }
    @GetMapping("/attendance/new")
    public String createAttendanceForm(Model model) {
        AttendanceScheduleDto attendance = new AttendanceScheduleDto();
        model.addAttribute("attendance", attendance);
        return "create_attendance";
    }
//    @GetMapping(value = "/attendance/today")
//    public List<AttendanceScheduleDto>
getAttendanceForToday(@ModelAttribute("attendances") AttendanceScheduleDto
attendance, Principal principal) {
//     return
attendanceService.getAttendanceScheduleForToday(principal.getName());
```

```
//    }
    @GetMapping(value = "/attendance/byDate")
    public String getAttendanceByDate(@ModelAttribute ScheduleRequestDto
attendanceRequestDto, Model model, Principal principal) {
        model.addAttribute("attendances",
attendanceService.getAttendanceScheduleForDate(LocalDateTime.parse(attendanc
eRequestDto.getStartDate()),
LocalDateTime.parse(attendanceRequestDto.getFinishDate()),
principal.getName()));
        return "attendances";
    }
    @PostMapping(value = "/attendance")
    public String addAttendance(@ModelAttribute("attendance")
AttendanceScheduleDto attendance, Model model, Principal principal) {
        attendanceService.saveAttendance(attendance);
        model.addAttribute("attendances",
attendanceService.getAttendanceScheduleForToday(principal.getName()));
        return "attendances";
    }
    @GetMapping("/attendance/edit/{id}")
    public String editCustomerForm(@PathVariable Long id, @NotNull Model
model) {
        model.addAttribute("attendance",
attendanceService.getAttendanceScheduleById(id));
        return "edit_attendance";
    }
    @PostMapping("/attendance/{id}")
    public String updateAttendance(@PathVariable Long id,
                                   @ModelAttribute("attendance") AttendanceScheduleDto
attendanceScheduleDto, Principal principal, Model model) {
```

```
        attendanceService.updateAttendance(attendanceScheduleDto, id);
        model.addAttribute("attendances",
attendanceService.getAttendanceScheduleForToday(principal.getName()));
        return "attendances";
    }
    @GetMapping("/attendance/{id}")
    public String deleteCustomer(@PathVariable Long id, Model model, Principal
principal) {
        attendanceService.deleteAttendanceById(id);
        model.addAttribute("attendances",
attendanceService.getAttendanceScheduleForToday(principal.getName()));
        return "attendances";
    @GetMapping(value = "/finance/today")
    public Double getFinanceForToday(Principal principal) {
        return
attendanceService.getFinance(attendanceService.getAttendanceScheduleForToday
(principal.getName()));
    }
    @GetMapping(value = "/finance/byDate")
    public Double getFinanceForByDate(@ModelAttribute ScheduleRequestDto
attendanceRequestDto, Principal principal) {
        return attendanceService.getFinance(
attendanceService.getAttendanceScheduleForDate(LocalDateTime.parse(attendanc
eRequestDto.getStartDate()),
LocalDateTime.parse(attendanceRequestDto.getFinishDate()),
principal.getName()));
    }
}
@Controller
@RequestMapping(value = "/customers")
```

```
@RequiredArgsConstructor
public class CustomerController {
    private final CustomerService customerService;
    @GetMapping()
    public String listCustomers(Model model) {
        model.addAttribute("customers", customerService.getAllCustomers());
        return "customers";
    }
    @GetMapping("/new")
    public String createCustomerForm(Model model) {
        CustomerDto customer = new CustomerDto();
        model.addAttribute("customer", customer);
        return "create_customer";
    }
    @PostMapping()
    public String saveCustomer(@ModelAttribute("customer") CustomerDto
customer) {
        customerService.saveCustomer(customer);
        return "redirect:/customers";
    }
    @GetMapping("/edit/{id}")
    public String editCustomerForm(@PathVariable Long id, @NotNull Model
model) {
        model.addAttribute("customer", customerService.getCustomerById(id));
        return "edit_customer";
    }
    @PostMapping("/{id}")
    public String updateCustomer(@PathVariable Long id,
                                 @ModelAttribute("customer") Customer customer) {
```

```
        // get student from database by id
        Customer existingCustomer = customerService.getCustomerById(id);
        existingCustomer.setId(id);
        existingCustomer.setFirstName(customer.getFirstName());
        existingCustomer.setLastName(customer.getLastName());
        existingCustomer.setEmail(customer.getEmail());
        // save updated student object
        customerService.updateCustomer(existingCustomer);
        return "redirect:/customers";
    }
    // handler method to handle delete student request
    @GetMapping("/{id}")
    public String deleteCustomer(@PathVariable Long id) {
        customerService.deleteCustomerById(id);
        return "redirect:/customers";
    }
}
@RestController
public class FinanceController {
    @GetMapping("/")
    public String getLoggedOutMessage(){
        return "Logged successfully";
    }
}
@Data
@NoArgsConstructor
public class AttendanceScheduleDto {
    private Long id;
    private String date;
```

```
    private String comment;
    private String clientEmail;
    private String workerEmail;
    private Double sum;
}
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CustomerDto {
    private String firstName;
    private String lastName;
    private String email;
    private String passwo
    private Boolean isAdmin;
}
@Data
public class ScheduleRequestDto {
    private String startDate;
    private String finishDate;
}
@ControllerAdvice
public class MyExceptionHandler extends ResponseEntityExceptionHandler {
    @ExceptionHandler(CustomerCreationException.class)
    public String handleException(){
        return "error";
    }
}
@Entity
@Table(name = "attendance")
@Data
```

```
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class AttendanceSchedule {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private LocalDateTime date;
   private String comment;
   private Double sum;
   @ManyToOne()
   @JsonBackReference
   @JoinColumn(name = "customer_email")
   private Customer client;
   @ManyToOne()
   @JsonBackReference
   @JoinColumn(name = "customer_id")
   private Customer worker;
}
public class Customer {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @Column(name = "first_name", nullable = false)
   private String firstName;
   @Column(name = "last_name", nullable = false)
   private String lastName;
   @Column(name = "email", unique = true, nullable = false)
   @Pattern(regexp ="/^[a-zA-Z0-9_`{|}~-]+@[a-zA-Z0-9-]+(?:\\.[a-zA-Z0-9-
]+)*$/",
```

```
    message ="{invalid.email}")
    private String em
    @Column()
    private String password;
    @ManyToMany(cascade = CascadeType.MERGE)
    @JoinTable(name = "customers_roles",
            joinColumns = @JoinColumn(name = "customer_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Collection<Role> roles;
    @OneToMany(mappedBy = "client", cascade = CascadeType.REMOVE)
    @JsonManagedReference
    private Collection<AttendanceSchedule> customerSchedules;
    @OneToMany(mappedBy = "worker", cascade = CascadeType.REMOVE)
    @JsonManagedReference
    private Collection<AttendanceSchedule> workerSchedulesSchedules;
    private boolean isAdmin;
}
@Entity
@Data
@Table(name = "roles")
@NoArgsConstructor
public class Role {
    public Role(String name) {
        this.name = name;
    }
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(unique = true)
    private String name;
```

```
public interface AttendanceService {
   void saveAttendance(AttendanceScheduleDto attendanceScheduleDto);
   void updateAttendance(AttendanceScheduleDto attendanceScheduleDto, Long
id);
   List<AttendanceScheduleDto> getAttendanceScheduleForDate(LocalDateTime
startDate, LocalDateTime finishDate, String email);
   List<AttendanceScheduleDto> getAttendanceScheduleForToday(String email);
   Double getFinance(List<AttendanceScheduleDto> attendanceScheduleDtos);
   AttendanceScheduleDto getAttendanceScheduleById (Long id);
   void deleteAttendanceById(Long id);
}
@RequiredArgsConstructor
public class AttendanceServiceImpl implements AttendanceService {
   private final AttendanceRepository attendanceRepository;
   private final CustomerService customerService;
   public void saveAttendance(AttendanceScheduleDto attendanceScheduleDto) {
      AttendanceSchedule attendanceSchedule = new AttendanceSchedule();
      attendanceSchedule.setComment(attendanceScheduleDto.getComment());
attendanceSchedule.setDate(LocalDateTime.parse(attendanceScheduleDto.getDate
()));
attendanceSchedule.setClient(customerService.findByEmail(attendanceScheduleD
to.getClientEmail()));
      attendanceSchedule.setWorker(
customerService.findByEmail(attendanceScheduleDto.getWorkerEmail()));
      attendanceSchedule.setSum(attendanceScheduleDto.getSum());
      attendanceRepository.save(attendanceSchedule);
   }
   public void updateAttendance(AttendanceScheduleDto attendanceScheduleDto,
Long id) {
      AttendanceSchedule attendanceSchedule = new AttendanceSchedule();
```

```
        attendanceSchedule.setId(id);
        attendanceSchedule.setComment(attendanceScheduleDto.getComment());
attendanceSchedule.setDate(LocalDateTime.parse(attendanceScheduleDto.getDate
()));
attendanceSchedule.setClient(customerService.findByEmail(attendanceScheduleD
to.getClientEmail()));
        attendanceSchedule.setWorker(
customerService.findByEmail(attendanceScheduleDto.getWorkerEmail()));
        attendanceSchedule.setSum(attendanceScheduleDto.getSum());
        attendanceRepository.save(attendanceSchedule);
    }
    public List<AttendanceScheduleDto>
getAttendanceScheduleForDate(LocalDateTime startDate, LocalDateTime
finishDate, String email) {
        return attendanceRepository.findAll().stream()
                .filter(attendanceSchedule ->
attendanceSchedule.getWorker().getEmail().equalsIgnoreCase(email))
                .filter(attendanceSchedule ->
attendanceSchedule.getDate().compareTo(startDate) >= 0 &&
attendanceSchedule.getDate().compareTo(finishDate) <= 0)
                .map(this::toAttendanceScheduleDto)
                .collect(Collectors.toList());
    }
    public List<AttendanceScheduleDto> getAttendanceScheduleForToday(String
email) {
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        String time = dtf.format(LocalDateTime.now());
        return attendanceRepository.findAll().stream()
                .filter(attendanceSchedule ->
attendanceSchedule.getDate().toString().contains(time))
```

```
            .filter(attendanceSchedule ->
attendanceSchedule.getWorker().getEmail().equalsIgnoreCase(email))
            .map(this::toAttendanceScheduleDto)
            .collect(Collectors.toList());
    }
    public Double getFinance(List<AttendanceScheduleDto>
attendanceScheduleDtos) {
        Double allSum = (double) 0;
        List<Double> sumList = attendanceScheduleDtos.stream()
            .map(AttendanceScheduleDto::getSum)
            .collect(Collectors.toList());
        for (Double s : sumList) {
            allSum += s;
        }
        return allSum;
    }
    public AttendanceScheduleDto getAttendanceScheduleById(Long id) {
        return toAttendanceScheduleDto(attendanceRepository.findById(id).get());
    }
    @Override
    public void deleteAttendanceById(Long id) {
        AttendanceSchedule attendanceSchedule = attendanceRepository.getOne(id);
        attendanceRepository.delete(attendanceSchedule);
    }
    private AttendanceScheduleDto toAttendanceScheduleDto(AttendanceSchedule
attendanceSchedule) {
        AttendanceScheduleDto attendanceScheduleDto = new
AttendanceScheduleDto();
        attendanceScheduleDto.setId(attendanceSchedule.getId());
        attendanceScheduleDto.setDate(attendanceSchedule.getDate().toString());
```

```
        attendanceScheduleDto.setComment(attendanceSchedule.getComment());

attendanceScheduleDto.setClientEmail(attendanceSchedule.getClient().getEmail())
;
attendanceScheduleDto.setWorkerEmail(attendanceSchedule.getWorker().getEmai
l());
        attendanceScheduleDto.setSum(attendanceSchedule.getSum());
        return attendanceScheduleDto;
    }
}
public interface CustomerService {
    List<Customer> getAllCustomers();
    Customer saveCustomer(CustomerDto customer);
    Customer getCustomerById(Long id);
    Customer updateCustomer(Customer student);
    void deleteCustomerById(Long id);
    Customer findByEmail(String email);
}
@Service
@RequiredArgsConstructor
public class CustomerServiceImpl implements CustomerService,
UserDetailsService {
    private final CustomerRepository clientRepository;
    private final RoleRepository roleRepository;
    @Override
    public List<Customer> getAllCustomers() {
        return clientRepository.findAll();
    }
    @Override
    public Customer saveCustomer(CustomerDto customer) {
```

```
        BCryptPasswordEncoder bCryptPasswordEncoder = new
BCryptPasswordEncoder();
        if (customer.getFirstName().equals("") || customer.getLastName().equals("")
                || customer.getEmail().equals("") || customer.getPassword().equals("")) {
            throw new CustomerCreationException("Всі поля клієнта повинні бути
заповнені");
        }
        Customer customerModel = new Customer();
        customerModel.setFirstName(customer.getFirstName());
        customerModel.setLastName(customer.getLastName());
customerModel.setPassword(bCryptPasswordEncoder.encode(customer.getPasswo
rd()));
        customerModel.setEmail(customer.getEmail());
        if (customer.getIsAdmin()) {
customerModel.setRoles(Collections.singletonList(roleRepository.findByName("R
OLE_ADMIN")));
            customerModel.setAdmin(true);
        } else {
customerModel.setRoles(Collections.singletonList(roleRepository.findByName("R
OLE_USER")));
        }
        return clientRepository.save(customerModel);
    }
    @Override
    public Customer getCustomerById(Long id) {
        return clientRepository.findById(id).get();
    }
    @Override
    public Customer updateCustomer(Customer customer) {
        return clientRepository.save(customer);
```

```
    }
    @Override
    public void deleteCustomerById(Long id) {
        Customer customer = getCustomerById(id);
        if (!customer.isAdmin()) {
            clientRepository.deleteById(id);
        }
    }
public Customer findByEmail(String email) {
        return clientRepository.findByEmail(email);
    }
    @Override
    @Transactional
    public UserDetails loadUserByUsername(String email) throws
UsernameNotFoundException {
        Customer customer = findByEmail(email);
        if (customer == null) {
            throw new UsernameNotFoundException(String.format("User %s not
found", email));
        }
        return new User(customer.getEmail(), customer.getPassword(),
mapRolesToAuthorities(customer.getRoles()));
    }
    private Collection<? extends GrantedAuthority>
mapRolesToAuthorities(Collection<Role> roles) {
        return roles.stream().map(role -> new
SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
    }
}
@Service
```

```
@RequiredArgsConstructor
public class MailService {
    private final JavaMailSender javaMailSender;
    private final AttendanceRepository attendanceRepository;
    @Scheduled(cron = "* 0 8 * * *")
//    @Scheduled(cron = "30 * * * * *")
    public void sendMail() {
        List<AttendanceSchedule> customers = attendanceRepository.findAll();
        List<String> emails = getEmailAddress(customers);
        emails.forEach(this::send);
 System.out.println("Mail sent successfully");
    }
    private void send(String receiver) {
        SimpleMailMessage message = new SimpleMailMessage();
        String text = " Не забудьте, що ми чекаємо на Вас сьогодні!";
        String subject = "Нагадування";
        message.setFrom("myuserforemail@gmail.com");
        message.setTo(receiver);
        message.setText(text);
        message.setSubject(subject);
        javaMailSender.send(message);
    }
    private List<String> getEmailAddress(List<AttendanceSchedule> customers) {
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        String time = dtf.format(LocalDateTime.now());
        return customers.stream().filter(attendanceSchedule ->
attendanceSchedule.getDate().toString().contains(time))
.map(AttendanceSchedule::getClient).map(Customer::getEmail).collect(Collectors.
toList());
    }
```

```
@Component
@RequiredArgsConstructor
public class SetupDataLoader implements
ApplicationListener<ContextRefreshedEvent> {
    private final RoleRepository roleRepository;
    private boolean alreadySetup;
    @Override
    @Transactional
    public void onApplicationEvent(ContextRefreshedEvent event) {
        if (alreadySetup) {
            return;
        }
        createRoleIfNotFound("ROLE_ADMIN");
        createRoleIfNotFound("ROLE_USER");
        alreadySetup = true;
    }
    @Transactional
    Role createRoleIfNotFound(
            String name) {
        Role role = roleRepository.findByName(name);
        if (role == null) {
            role = new Role(name);
            roleRepository.save(role);
        }
        return role;
    }
}
```

ДОДАТОК В

Алгоритм роботи програми

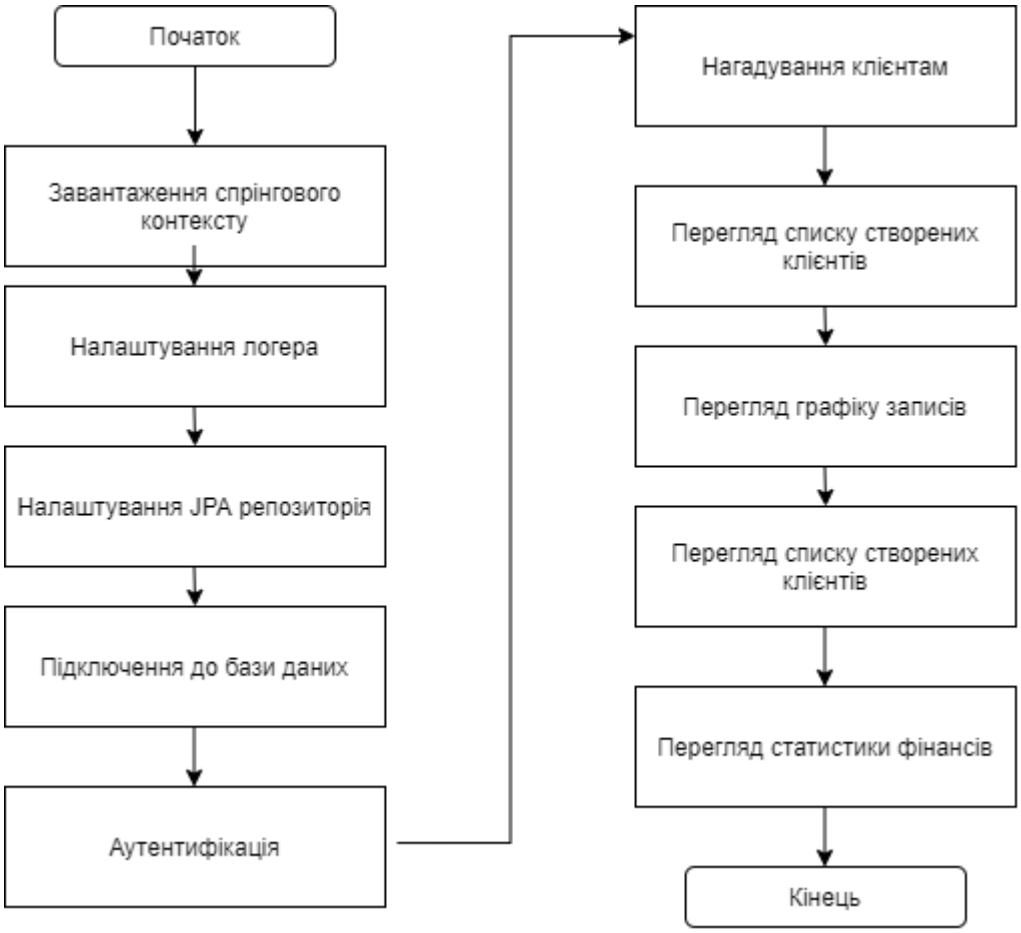

Рисунок В.1 — Алгоритм роботи програми

ДОДАТОК Г

Переваги та недоліки альтернативних додатків

Таблиця Г.1 — Переваги та недоліки альтернативних додатків

Назва системи	Переваги	Недоліки
Capsule	Експорт та імпорт даних Система сповіщень	Складний інтерфейс користувача Обмежений список функцій
Bitrix24	Прямий контакт із клієнтами Система сповіщень	Складний інтерфейс користувача Обмеження соціальних мереж
Scoro	Зручний інтерфейс Система нагадувань	Складність синхронізації Висока ціна

ДОДАТОК Д

Графічний інтерфейс моніторингу статистики

Рисунок Д.1 — Графічний інтерфейс моніторингу статистики

ДОДАТОК Е

Моніторинг статистики по конкретній даті

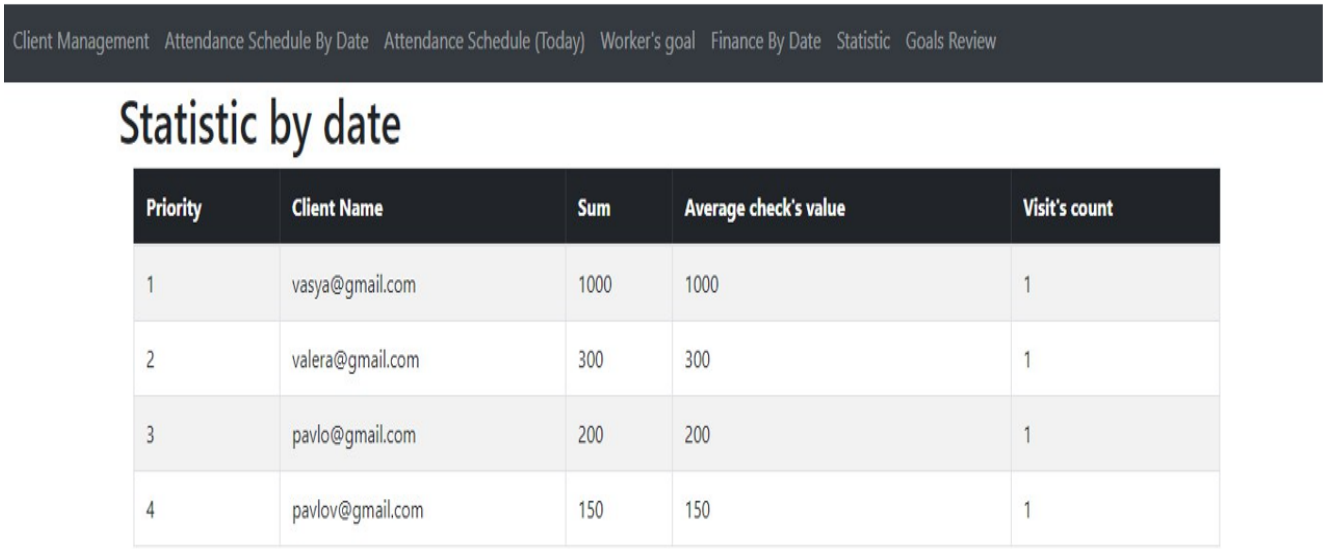

Рисунок Е.1 — Моніторинг статистики по конкретній даті

ДОДАТОК Ж
ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:___Інформаційна система підтримки підприємств малого __

_ бізнесу у сфері послуг_____

Тип роботи:_____магістерська кваліфікаційна робота _

Підрозділ _____кафедра обчислювальної техніки, 1КІ-20м

Науковий керівник _____Захарченко С.М., проф. кафедри ОТ_____

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	91,5
Тривога/Білі знаки	/	Схожість	8,5

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення 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 і відсутності самостійності її автора. Робот 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.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ __Боднар К. О. _____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

____Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

_____

Особа, відповідальна за перевірку_____ _Захарченко С.М.__

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали)