НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.45_

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інтегровані інформаційні системи»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Інформаційна система управління чергами обслуговування в державних установах»

Виконав:
студент 2 курсу, групи ІА-32мп
Дудік Анатолій Анатолійович _____

Керівник:
д.ф.-м.н., професор
Дорошенко Анатолій Юхимович _____

Рецензент:
д.т.н., професор кафедри ІПІ
Стеценко Інна Вячеславівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Дудіку Анатолію Анатолійовичу

1. Тема дисертації «Інформаційна система управління чергами обслуговування в державних установах», науковий керівник дисертації Дорошенко Анатолій Юхимович, д.ф.-м.н., професор, затверджені наказом по університету від «08» 11 2024 р. № 5016-с

2. Термін подання студентом дисертації «09» 12 2024 р.

3. Об’єкт дослідження: процес організації та управління чергами у державних установах

4. Вихідні дані: мова програмування Java, система управління бази даних MySQL

5. Перелік завдань, які потрібно розробити: провести аналіз існуючих рішень, сформувати вимоги до системи, обрати елементи та технології для розроблення системи, спроєктувати архітектуру системи, реалізувати бізнес-логіку та інтерфейс користувача, провести тестування, підготувати текстову та графічну частину пояснювальної записки

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: структурна схема, функціональна схема, діаграма використання, діаграма діяльності, діаграма послідовності, ER-діаграма

7. Орієнтовний перелік публікацій: не планується

8. Дата видачі завдання 02.09.2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Затвердження теми роботи	08.09.2024	
2.	Аналіз предметної області	12.09.2024	
3.	Аналіз існуючих рішень	16.09.2024	
4.	Визначення вимог до системи	23.09.2024	
5.	Розробка сценарію використання системи	30.09.2024	
6.	Визначення технологій та елементів розробки	10.10.2024	
7.	Розробка програмного застосунку	04.11.2024	
8.	Тестування програмного застосунку	11.11.2024	
9.	Створення стартап проєкту	17.11.2024	
10.	Оформлення дипломного проєкту	24.11.2024	
11.	Передзахист	25.11.2024	
12.	Захист		

Студент Анатолій ДУДІК

Науковий керівник Анатолій ДОРОШЕНКО

РЕФЕРАТ

Інформаційна система управління чергами обслуговування в державних установах: 136 с., 56 табл., 39 рис., 9 дод., 22 джерела.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ ЧЕРГАМИ, ДЕРЖАВНІ УСТАНОВИ, ЯКІСТЬ ОБСЛУГОВУВАННЯ, АВТОМАТИЗАЦІЯ, АДАПТИВНИЙ АЛГОРИТМ, ВЕБ-ДОДАТКИ.

Актуальність теми дослідження зумовлена необхідністю вдосконалення процесів обслуговування громадян у державних установах шляхом впровадження сучасних інформаційних систем управління чергами. Такі системи дозволяють зменшити час очікування, підвищити продуктивність роботи персоналу та покращити якість послуг, що позитивно впливає на довіру громадян до державних органів.

Метою роботи є створення методики розроблення інформаційних систем управління чергами, які забезпечать автоматизацію процесів реєстрації, управління чергами та надання послуг громадянам. Особливістю розробки є використання адаптивного алгоритму для рівномірного розподілу клієнтів між працівниками, інтеграція зручного інтерфейсу, створеного державною мовою, та підтримка сучасних технологій.

Об’єктом дослідження є процес організації та управління чергами у державних установах, а предметом — інформаційна система, що забезпечує автоматизацію цього процесу.

Наукова новизна роботи полягає у створенні адаптивної інформаційної системи управління чергами, яка оптимізує процеси реєстрації, виклику та розподілу клієнтів. Розроблена система включає інноваційний адаптивний алгоритм розподілу, ергономічний інтерфейс та можливість попередньої реєстрації громадян.

ABSTRACT

Information system of service queues management in public institutions: 136 p., 56 tables, 39 figures, 9 appendix, 22 sources.

Keywords: INFORMATION SYSTEM, QUEUE MANAGEMENT, PUBLIC INSTITUTIONS, QUALITY OF SERVICE, AUTOMATION, ADAPTIVE ALGORITHM, WEB APPLICATIONS.

The relevance of the research topic is due to the need to improve the processes of servicing citizens in public institutions through the introduction of modern information systems for queue management. Such systems can reduce waiting times, increase staff productivity and improve the quality of services, which has a positive impact on public confidence in government agencies.

The aim of the work is to create a methodology for developing information systems for queue management that will automate the processes of registration, queue management and provision of services to citizens. The peculiarity of the development is the use of an adaptive algorithm for the uniform distribution of clients among employees, the integration of a user-friendly interface created in the state language, and the support of modern technologies.

The object of the study is the process of organizing and managing queues in public institutions, and the subject is an information system that automates this process.

The scientific novelty of the work is the creation of an adaptive information system for queue management that optimizes the processes of registration, calling and distribution of clients. The developed system includes an innovative adaptive allocation algorithm, an ergonomic interface and the possibility of pre-registration of citizens.

ЗМІСТ

ВСТУП

Організація процесу обслуговування громадян у державних установах є важливим аспектом функціонування сучасного суспільства. Неефективність існуючих методів управління чергами призводить до великих черг, втрати часу громадян та підвищення навантаження на працівників державних органів.

Впровадження інформаційних систем для управління чергами дозволяє автоматизувати цей процес, підвищуючи продуктивність роботи установи та забезпечує комфортне обслуговування громадян. Розробка даної інформаційної системи управління чергами є актуальним завданням, що сприяє вдосконаленню якості обслуговування державних послуг та підвищенню рівня довіри до державних установ.

Наукова проблема, що вирішується в цій роботі, полягає у створенні ефективної системи управління чергами, яка дозволить автоматизувати процес обслуговування громадян, зменшити час очікування та підвищити загальну ефективність державних установ. Дослідження спрямоване на подолання існуючих проблем у сфері управління чергами за допомогою впровадження сучасних інформаційних технологій.

Вибраний напрямок досліджень відповідає галузевим і державним програмам, спрямованим на підвищення якості надання державних послуг та розвиток інформаційних технологій у державному секторі.

Метою дослідження є розробка методики створення інформаційної системи управління чергами, яка дозволить автоматизувати процеси реєстрації, управління чергами та надання послуг громадянам шляхом адаптивного алгоритму, поєднати весь функціонал у одному програмному застосунку з ергономічним інтерфейсом, написаним державною мовою.

Для досягнення поставленої мети вирішуються наступні задачі дослідження:

– провести аналіз існуючих рішень для управління чергами у державних установах, виявити їхні переваги та недоліки;

– сформувати вимоги до інформаційної системи, що відповідають потребам державних установ;

– розробити архітектуру інформаційної системи управління чергами та обґрунтувати вибір технологій і елементів системи;

– підвищити ефективність інф системи за рахунок адаптивного алгоритму, ергономічності графічного інтерфейсу, впровадження державної мови в інтерфейс користувачів;

– реалізувати прототип інформаційної системи, що забезпечить ефективне управління чергами;

– провести тестування інформаційної системи на основі реальних даних та оцінити її ефективність.

Об’єктом дослідження є процес організації та управління чергами у державних установах.

Предметом дослідження є інформаційна система управління чергами, що забезпечує автоматизацію процесу обслуговування громадян у державних установах.

Для досягнення мети дослідження використовуються такі методи як системний аналіз для аналізу існуючих рішень та визначення вимог до системи, методи моделювання для створення архітектури інформаційної системи та процесів управління чергами, програмування для розробки інформаційної системи і тестування для оцінки правильності роботи системи

Наукова новизна роботи полягає у створенні методики розроблення інформаційної системи управління чергами, яка дозволить автоматизувати процеси реєстрації та обслуговування громадян у державних установах. Завдяки розробленій методиці створена інформаційна система включатиме можливість попередньої реєстрації громадян, автоматичного розподілу клієнтів між працівниками, що забезпечує зменшення часу очікування та підвищення якості наданих послуг.

Практичне значення результатів роботи полягає у можливості впровадження розробленої інформаційної системи у державних установах різного рівня, таких як

Центри надання адміністративних послуг, паспортні столи, податкові інспекції. Впровадження системи дозволить значно скоротити час очікування громадян, підвищити ефективність роботи працівників та покращити якість наданих послуг.

Магістерська дисертація складається зі вступу, опису предметної області, аналізу існуючих рішень, формування вимог до системи, вибору та обгрунтування елементів та технологій, структурної схеми системи, , er-діаграми, реалізації бізнес логіки системи, розроблення інтерфейса користувача, тестування системи, стартап проєкту і висновків, списку використаних джерел із 22 найменувань та 9 додатків. Загальний обсяг роботи становить 129 сторінок. Графічна частина включає 8 ілюстрацій формату А3.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційні системи управління чергами

Система керування чергою (СКЧ) - це інструмент, призначений для впорядкування та управління потоком клієнтів у сфері обслуговування [1]. Вона складається з наступних основних компонентів:

– система реєстрації клієнтів: громадяни можуть зареєструватися через інтерфейс (кіоск самообслуговування або онлайн-сервіси), обравши необхідну послугу. Система автоматично надає клієнту номер черги та інформацію про час очікування;

– система розподілу клієнтів: після реєстрації система автоматично розподіляє клієнтів між працівниками, враховуючи їхнє навантаження та види послуг, які вони можуть надати;

– монітори для відображення черги: екрани встановлені в залі очікування для відображення інформації про поточний стан черги та виклику клієнтів до відповідних працівників;

– система звітності та аналітики: інформаційна система дозволяє керівництву отримувати аналітичні дані про роботу установи: середній час обслуговування, кількість оброблених запитів, завантаженість працівників тощо.

1.2 Актуальні проблеми управління чергами

Управління чергою в державних установах зазвичай базується на принципі «хто прийшов, той і обслужений». Однак сучасні вимоги до якості обслуговування змушують звернути увагу на низку проблем, які виникають у процесі обслуговування громадян:

– непередбачуваність часу очікування: у державних установах клієнти часто стикаються з великими чергами через відсутність попереднього запису або системи розподілу черги. Це створює ситуації, коли клієнти можуть чекати кілька годин, що свідчить про низьку ефективність роботи установи;

– нерівномірний розподіл клієнтів: відсутність автоматизованих систем розподілу клієнтів призводить до надмірного навантаження на окремі відділи або працівників. Як наслідок, працівники можуть працювати з різною інтенсивністю протягом дня, що негативно впливає на загальну ефективність роботи установи;

– відсутність звітності: більшість державних установ не мають можливості контролювати роботу персоналу та ефективність обслуговування клієнтів. Це ускладнює прийняття управлінських рішень для оптимізації роботи установи;

– незадоволеність клієнтів: через непередбачуваний час очікування, низьку якість обслуговування та нерівномірний розподіл навантаження клієнти часто незадоволені роботою державних установ. Це може призвести до втрати довіри до державних органів та негативного іміджу установи.

1.3 Переваги впровадження інформаційних систем управління чергами

Впровадження інформаційної системи управління чергою дозволяє досягти наступних результатів:

– скорочення часу очікування: система дозволяє зменшити середній час очікування в черзі за рахунок раціонального розподілу клієнтів та можливості попереднього запису;

– підвищення ефективності роботи персоналу: завдяки автоматичному розподілу клієнтів система рівномірно завантажує співробітників, що дозволяє підвищити продуктивність праці;

– підвищення рівня задоволеності клієнтів: завдяки можливості попереднього запису та швидшому обслуговуванню клієнти більш задоволені роботою державних органів;

– моніторинг та аналіз ефективності роботи: інформаційна система надає керівництву звіти про роботу установи в режимі реального часу, що дозволяє швидко реагувати на проблеми та впроваджувати покращення.

Висновки до розділу 1

Впровадження інформаційних систем управління чергою є важливим кроком на шляху до підвищення ефективності роботи державних установ, що дозволяє зменшити навантаження на персонал, скоротити час очікування клієнтів та підвищити якість надання послуг. Вирішення описаних проблем підтверджує актуальність дослідження та розробки таких систем для державного сектору України.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Впровадження систем управління чергами є одним із ключових напрямів модернізації державних установ України. З кожним роком попит на державні послуги зростає, що потребує вдосконалення процесів обслуговування громадян.

Сучасні інформаційні системи управління чергами дозволяють автоматизувати процес обслуговування, зменшити час очікування та підвищити загальну ефективність роботи установ.

У цьому розділі буде проведено аналіз трьох найпоширеніших систем управління чергами, які впроваджуються в державних установах України: Waitwhile, Qmatic та Qmate. Кожна з цих систем має свої унікальні можливості, переваги та обмеження, які слід враховувати при створенні інформаційної системи управління чергами.

2.1 Система управління чергами Qmatic

Qmatic[2] – це одна з провідних світових систем управління чергами, яка використовується у більш ніж 120 країнах, включаючи Україну. Її впровадження можна побачити в багатьох державних установах, таких як Центри надання адміністративних послуг (ЦНАП), державні міграційні служби, податкові інспекції та інші організації. Система Qmatic дозволяє не лише організувати процес управління чергами, а й інтегрувати функціонал для аналізу та оптимізації робочих процесів установи.

Переваги системи Qmatic:

– гнучкість та адаптація до різних умов. Qmatic має модульну архітектуру, яка дозволяє адаптувати систему до специфіки кожної установи, налаштовуючи різні процеси, типи послуг і категорії клієнтів. Це дозволяє налаштувати систему під потреби як великих державних установ, так і менших регіональних офісів;

– попередня реєстрація та дистанційне керування чергою. Громадяни можуть записатися на прийом заздалегідь через онлайн-платформи або мобільні додатки, що значно зменшує час очікування на місці та дозволяє краще планувати візит;

– моніторинг у реальному часі. Керівництво установи може отримувати в реальному часі дані про завантаженість черг, середній час обслуговування клієнтів та ефективність роботи персоналу. Це дозволяє швидко реагувати на перевантаження, оптимізувати роботу співробітників та приймати управлінські рішення для покращення процесів обслуговування;

– інтеграція з державними інформаційними системами. Система легко інтегрується з іншими інформаційними системами державних органів, що дозволяє швидко обмінюватися даними та забезпечувати безперервний робочий процес.

Недоліки системи Qmatic:

– висока вартість впровадження та підтримки. Впровадження Qmatic потребує значних фінансових ресурсів, оскільки система включає як програмне забезпечення, так і спеціалізоване обладнання (кіоски для самореєстрації, дисплеї для виклику, сенсорні термінали). Це робить її недоступною для невеликих державних установ або установ із обмеженим бюджетом;

– складність налаштування та обслуговування. Система потребує високої кваліфікації ІТ-персоналу для налаштування та подальшого адміністрування. Через велику кількість функцій та інтеграційних можливостей, налаштування системи може зайняти чимало часу;

– потреба в регулярних оновленнях. Для забезпечення безперебійної роботи та збереження високого рівня безпеки система потребує постійних оновлень та технічної підтримки, що також збільшує загальні витрати на її утримання.

2.2 Система управління чергами Qmate

Qmate[3] – це ще одне рішення для автоматизації процесу управління чергами від Servus System International, яке підходить для середніх і великих організацій. Вона має простий у використанні інтерфейс, що робить її зручною для швидкого впровадження у державних установах.

Переваги системи Qmate:

– простий інтерфейс: система qmate легко освоюється як клієнтами, так і працівниками, завдяки чому мінімізується потреба в навчанні персоналу;

– планування часу обслуговування. qmate дозволяє клієнтам вибирати зручний час відвідування та самостійно реєструватися на прийом;

– можливості звітності. qmate надає аналітичні дані для відстеження ефективності роботи установи та рівня задоволеності клієнтів, що допомагає приймати обґрунтовані управлінські рішення;

– інтеграція з crm-системами. qmate підтримує інтеграцію з crm-платформами, що робить систему зручною для подальшого обслуговування та підтримки клієнтів.

Недоліки системи Qmate:

– обмежена функціональність для великих установ: система підходить для середніх установ, але для масштабних організацій вона може виявитися недостатньо гнучкою;

– відсутність можливості індивідуального налаштування: Qmate не пропонує розширених можливостей для складних налаштувань, що може бути обмеженням для організацій зі специфічними вимогами;

– залежність від стабільного підключення до інтернету: оскільки система працює в режимі онлайн, проблеми з доступом до мережі можуть ускладнити її використання.

2.3 Система управління чергами Waitwhile

Waitwhile[4] – це гнучка хмарна система управління чергами, яка підходить для організацій різного масштабу, включаючи державні установи. Система дозволяє клієнтам записуватися онлайн, керувати своїм часом очікування та отримувати повідомлення про стан черги, що робить процес обслуговування більш зручним та ефективним.

Переваги системи Waitwhile:

– гнучкість і масштабованість: система доступна як для невеликих, так і для великих організацій, дозволяючи масштабувати потужність в залежності від кількості клієнтів та навантаження;

– попередня реєстрація і планування: waitwhile дозволяє клієнтам реєструватися в черзі заздалегідь або в реальному часі через онлайн-платформу. це зменшує час очікування на місці та дозволяє клієнтам краще планувати свій візит;

– автоматизовані повідомлення та оновлення. клієнти можуть отримувати повідомлення про час наближення своєї черги через sms або електронну пошту, що знижує потребу в фізичній присутності у залі очікування;

– інтеграція з іншими сервісами. waitwhile інтегрується з популярними платформами, такими як google calendar, crm-системами та платформами для відгуків, що полегшує управління обслуговуванням.

Недоліки системи Waitwhile:

– обмежена функціональність у безкоштовній версії: хоча waitwhile пропонує безкоштовний тариф, він має обмежені функції, що може бути незручним для державних установ із великим потоком клієнтів;

– залежність від інтернету. оскільки система є хмарною, вона вимагає стабільного підключення до інтернету, що може бути проблемою для віддалених установ із слабким сигналом;

– відсутність повного локалізованого інтерфейсу. деякі елементи інтерфейсу можуть бути не повністю адаптовані під локальні мови та специфіку державних установ.

Висновки до розділу 2

У даному розділі розглянуто декілька інформаційних систем, які поширено використовуються в державних установах. Усі розглянуті системи підтримують функцію автоматизованого інформування клієнтів про стан черги, що дає змогу клієнтам краще планувати час і знижує навантаження на персонал. Наостанок, кожній системі притаманний високий рівень безпеки та захисту персональних даних.

Враховуючи ці аспекти, можна зробити висновок, що для створення ефективної системи управління чергами необхідно включити такі вимоги як надійність і точність роботи системи, автоматизація інформування клієнтів та захист даних користувачів.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Розробка інформаційної системи управління чергами в державних установах вимагає визначення чітких вимог, що стосуються функціональних можливостей та якості її роботи. Вимоги до системи поділяються на дві основні категорії: функціональні та нефункціональні вимоги.

3.1 Функціональні вимоги

Функціональні вимоги визначають, що саме система повинна виконувати, тобто описують її функціональність та основні завдання, які система повинна вирішувати для забезпечення ефективного управління чергами. Перелік функціональних вимог системи є наступним:

– реєстрація клієнтів: система повинна забезпечувати можливість реєстрації через веб-інтерфейс. Реєстрація включає вибір послуги та отримання номера черги;

– попередня реєстрація: система повинна надавати можливість попередньої реєстрації через веб-інтерфейс. Клієнти можуть обрати зручний для них час для отримання послуги, що зменшує ймовірність переповнених черг та поліпшує планування ресурсів установи. Попередня реєстрація має бути інтегрована з основним процесом обслуговування, що дозволить організувати загальну чергу;

– виклик клієнтів: для зручності обслуговування клієнтів система повинна забезпечувати виклик до віконця обслуговування. Виклик повинен відображати номер талона та віконце, куди йому слід підійти. Це автоматизує процес виклику та дозволяє знизити кількість помилок, які можуть виникати під час використання ручних методів;

– адаптивний алгоритм розподілення талонів: система повинна використовувати алгоритми для автоматичного розподілу клієнтів між робочими місцями. Алгоритм враховує поточне завантаження працівників, щоб забезпечити рівномірне навантаження та мінімізувати час очікування.

– зміна черги: система повинна дозволяти клієнтам змінювати свій номер у черзі або перенести час візиту. Наприклад, у випадку непередбачених обставин, клієнт має мати можливість самостійно перенести своє обслуговування на інший час або дату. Це підвищить гнучкість та зручність використання системи.

3.2 Нефункціональні вимоги

Нефункціональні вимоги описують якість роботи системи, обмеження на її виконання, такі як продуктивність, безпека, доступність, та інші параметри, що забезпечують надійність та ефективність системи. Перелік нефункціональних вимог до системи є наступним:

– доступність: система повинна бути доступною 99.9% часу, що забезпечить безперебійну роботу під час робочих годин установи. Це важливо для підтримання високого рівня обслуговування громадян, які розраховують на доступ до послуг у будь-який момент протягом робочого дня;

– продуктивність: система повинна обробляти запити на реєстрацію клієнтів із затримкою не більше 2 секунд. Це дозволить забезпечити швидкий відгук системи на дії користувачів і зменшити час, який витрачається на обслуговування кожного клієнта;

– масштабованість: система повинна підтримувати можливість масштабування для обслуговування збільшеної кількості клієнтів та нових типів послуг. Це означає, що система має бути гнучкою до змін і здатною до розширення без значних змін у своїй архітектурі.

– безпека: безпека є критичним аспектом при роботі з персональними даними клієнтів. Система повинна забезпечувати захист даних, а доступ до неї має бути обмежений за допомогою автентифікації користувачів;

– надійність: система повинна бути стійкою до збоїв і автоматично відновлюватися після короткочасних збоїв у роботі серверів або мережі. Це дозволить знизити ризики простоїв та забезпечити стабільну роботу навіть у разі виникнення технічних проблем;

– зручність використання: інтерфейс системи повинен бути інтуїтивно зрозумілим, щоб користувачі з різним рівнем технічної підготовки могли легко ним користуватися. Це важливо для зменшення часу навчання персоналу та підвищення загальної ефективності використання системи;

– підтримка та обслуговування: система повинна мати можливість легкої модернізації та обслуговування, з мінімальними вимогами до зупинки роботи під час оновлення. Це забезпечить безперервне надання послуг громадянам навіть під час проведення технічних робіт;

– Інтеграція державної мови**:** інтерфейс системи повинен повністю підтримувати державну мову, забезпечуючи коректний переклад і відповідність правовим вимогам щодо використання української мови в державних установах.

– документація: повинна бути доступна повна документація для адміністратора системи та користувачів, що дозволить легко налаштовувати і використовувати систему, а також швидко вирішувати технічні проблеми.

Висновки до розділу 3

У цьому розділі було визначено функціональні та нефункціональні вимоги до інформаційної системи управління чергами, які є основою для її розробки та впровадження в державних установах. Визначення чітких функціональних вимог дозволило зрозуміти, які саме задачі має виконувати система для забезпечення ефективного управління чергами та покращення якості обслуговування громадян. За допомогою нефункціональних вимог були визначені параметри якості роботи системи, такі як доступність, надійність, продуктивність та безпека, що дозволяють забезпечити стабільну та ефективну роботу системи в умовах високого навантаження та вимог сучасних державних установ.

4 СЦЕНАРІЙ ВИКОРИСТАННЯ СИСТЕМИ

Сценарії використання інформаційної системи управління чергами є основним елементом, що дозволяє визначити функціональні можливості системи та взаємодію користувачів із нею. Для візуалізації сценаріїв використання було застосовано UML-діаграми (Unified Modeling Language), що забезпечують наочне представлення функціональних вимог і взаємодії основних акторів із системою. У цьому розділі наведено UML-діаграми сценаріїв використання (use-case diagrams) та детальний опис кожного сценарію, що охоплює всі ключові процеси в системі.

На додатку Б представлено акторів, які взаємодіють із системою, та основні сценарії використання, що забезпечують функціональність системи.

На діаграмі використання показано основних акторів та сценарії взаємодії:

– адміністратор: відповідальна особа, що здійснює налаштування роботи системи, а також отримує звіти про її роботу;

– працівник: співробітник, який безпосередньо обслуговує клієнтів та працює з чергою;

– клієнт: громадянин, який звертається до державної установи для отримання певної послуги.

Діаграма використання системи наведена у додатку Б. У підрозділах 4.1-4.4 наведено детальний опис основних прецедентів для кожного з акторів. Опис включає основні дії, альтернативні сценарії та залежності між прецедентами.

4.1 Опис спільних прецедентів

Таблиця 4.1 – Прецедент автентифікації

Назва	Автентифікація
Актори	Працівник, адміністратор, клієнт
Опис	Кожен актор повинен автентифікуватись у системі, вводячи свої облікові дані для отримання доступу до функцій
Передумова	Актор має облікові дані в системі
Успішний сценарій	1. Актор вводить логін та пароль 2. Система перевіряє дані 3. Якщо дані правильні, система надає доступ до функцій
Результат	Актор отримує доступ до системи
Виключення	Якщо дані введені неправильно, система виводить помилку і просить повторити введення

Таблиця 4.2 – Прецедент моніторингу статусу черги

Назва	Моніторинг стану черги
Актори	Працівник, адміністратор, клієнт
Опис	Користувач може переглядати стан черги, який талон було викликано до якого робочого місця
Передумова	-
Успішний сценарій	1. Користувач переходить на основну сторіку системи 2. Користувач переглядає список викликаних талонів
Результат	Користувач отримує доступ до поточної інформації про чергу
Виключення	Якщо дані недоступні, система виводить повідомлення про помилку або відсутність даних

4.2 Прецеденти клієнта

Таблиця 4.3 – Прецедент створення попереднього запису

Назва	Створити попередній запис
Актори	Клієнт
Опис	Актор може створити попередній запис на послугу, вибравши вільний час і тип послуги
Передумова	Актор автентифікований у системі
Успішний сценарій	1. Актор вибирає послугу та час 2. Система перевіряє наявність вільного часу 3. Система фіксує запис
Результат	Система генерує запис із номером черги
Виключення	Якщо немає вільного часу, система пропонує інші варіанти

Таблиця 4.4 – Прецедент редагування попереднього запису

Назва	Редагувати попередній запис
Актори	Клієнт
Опис	Актор може змінити існуючий запис на новий час, обравши іншу вільну годину
Передумова	Створений попередній запис у черзі
Успішний сценарій	1. Актор вибирає новий час 2. Система перевіряє наявність часу Система оновлює запис та змінює номер черги відповідно до нового часу
Результат	Попередній запис змінено на новий час
Виключення	Якщо новий час недоступний, система виводить повідомлення про неможливість перезапису

Таблиця 4.5 – Прецедент видалення попереднього запису

Назва	Видалити попередній запис
Актори	Працівник, адміністратор, клієнт
Опис	Актор може видалити раніше створений запис із черги
Передумова	Створений попередній запис у черзі
Успішний сценарій	1. Актор вибирає опцію видалення запису 2. Система запитує підтвердження дії 3. Система видаляє запис із бази даних
Результат	Запис успішно видалено
Виключення	Якщо актор не підтверджує видалення, запис залишається в черзі

4.3 Прецеденти працівника

Таблиця 4.6 – Прецедент поточної черги працівника

Назва	Перегляд поточної черги працівника
Актори	Працівник
Опис	Працівник може переглядати поточну чергу до прив'язаного робочого місця
Передумова	Працівник автентифікований в системі
Успішний сценарій	1. Працівник переходить на панель працівника 2. Система ідентифікує номер робочого місця 3. Система показує інформацію про поточну чергу та поточного клієнта
Результат	Інформація про поточну чергу для робочого місця надано
Виключення	Якщо дані недоступні, система виводить повідомлення про помилку або відсутність даних

Таблиця 4.7 – Прецедент створення талону

Назва	Створити талон
Актори	Працівник
Опис	Працівник може створити талон в черзі
Передумова	Працівник автентифікований в системі
Успішний сценарій	1. Актор обирає створити талон 2. Актор обирає послугу 3. Система створює талон, фіксує робоче місце, якому призначується талон та розміщує його в черзі
Результат	Талон створено
Виключення	Якщо актор не підтверджує створення, процес створення талону скасовується

Таблиця 4.8 – Прецедент редагування талону

Назва	Редагувати талон
Актори	Працівник
Опис	Працівник може редагувати талон в черзі
Передумова	Працівник автентифікований в системі Талон вже існує в системі
Успішний сценарій	1. Актор обирає редагувати талон 2. Актор обирає послугу 3. Система оновлює талон
Результат	Талон створено
Виключення	Якщо актор не підтверджує редагування, процес редагування талону скасовується

Таблиця 4.9 – Прецедент видалення талону

Назва	Видалити талон
Актори	Працівник
Опис	Працівник може видаляти талон талон в черзі
Передумова	Працівник автентифікований в системі Талон вже існує в системі
Успішний сценарій	1. Актор обирає редагувати талон 2. Актор обирає талон 3. Система видаля талон
Результат	Талон видалено з системи
Виключення	Якщо актор не підтверджує видалення, процес видалення талону скасовується

Таблиця 4.10 – Прецедент виклику клієнта

Назва	Виклик клієнта
Актори	Працівник, адміністратор
Опис	Актор може викликати наступного клієнта через систему
Передумова	Є зареєстровані клієнти в черзі
Успішний сценарій	1. Працівник натискає на кнопку виклику клієнта 2. Система визначає наступного клієнта 3. Система відображає номер клієнта на екрані
Результат	Клієнт отримує виклик до віконця обслуговування
Виключення	Якщо немає клієнтів, система виводить повідомлення про відсутність клієнтів

Таблиця 4.11 – Прецедент передачі клієнта

Назва	Передати клієнта
Актори	Працівник, адміністратор
Опис	Працівник може передати клієнта іншому працівникові для продовження обслуговування.

Назва	Передати клієнта
Передумова	Клієнт викликаний до віконця обслуговування
Успішний сценарій	1. Працівник вибирає опцію передачі клієнта 2. Система пропонує список вільних віконець для передачі 3. Система оновлює статус клієнта
Результат	Клієнт переданий іншому працівнику для обслуговування
Виключення	Якщо немає вільних працівників, система пропонує завершити сеанс або залишити клієнта у черзі

Таблиця 4.12 – Прецедент завершення сеансу

Назва	Завершити сеанс
Актори	Працівник, адміністратор
Опис	Працівник може завершити сеанс після надання послуги клієнту
Передумова	Клієнт отримав послугу
Успішний сценарій	1. Працівник натискає кнопку завершення сеансу 2. Система фіксує, що послугу надано 3. Система закриває сеанс для цього клієнта
Результат	Сеанс із клієнтом завершено
Виключення	Якщо працівник не завершив сеанс, система може автоматично завершити його після певного часу бездіяльності

4.4 Прецеденти адміністратора

Таблиця 4.13 – Прецедент керування послугами

Назва	Керування послугами
Актори	Адміністратор
Опис	Адміністратор може додавати, редагувати або видаляти послуги, які надаються через систему
Передумова	Система налаштована для роботи з послугами
Успішний сценарій	1. Адміністратор вибирає опцію «Керування послугами» 2. Адміністратор може додати нову послугу, відредагувати або видалити існуючу послугу
Результат	Послуги успішно додані, відредаговані або видалені
Виключення	Якщо адміністратор не має прав для внесення змін, система видає повідомлення про недостатні права

Таблиця 4.14 – Прецедент керування працівниками

Назва	Керування працівниками
Актори	Адміністратор
Опис	Адміністратор може додавати, редагувати або видаляти облікові записи працівників системи
Передумова	Система налаштована для роботи з працівниками
Успішний сценарій	1. Адміністратор вибирає опцію «Керування працівниками» 2. Адміністратор додає нового працівника, змінює інформацію про існуючого або видаляє його
Результат	Дані працівників успішно оновлені або видалені
Виключення	Якщо адміністратора не має прав для внесення змін, система видає повідомлення про недостатні права

Таблиця 4.15 – Прецедент керування робочими місцями

Назва	Керування робочими місцями
Актори	Адміністратор
Опис	Адміністратор може додавати, редагувати або видаляти робочі місця, за якими закріплені працівники і послуги
Передумова	Система налаштована для роботи з працівниками
Успішний сценарій	1. Адміністратор вибирає опцію «Керування робочими місцями» 2. Адміністратор додає нове робоче місце, змінює інформацію про існуюче або видаляє його
Результат	Робоче місце успішно оновлене або видалене
Виключення	Якщо адміністратора не має прав для внесення змін, система видає повідомлення про недостатні права

Таблиця 4.16 – Прецедент керування прив’язками послуг до робочих місць

Назва	Керування робочими прив’язками послуг до робочих місць
Актори	Адміністратор
Опис	Адміністратор може прив’язувати і відв’язувати послуги від робочих місць
Передумова	Послуга і робоче місце наявне в системі
Успішний сценарій	1. Адміністратор вибирає опцію «Керування робочими прив’язками послуг до робочих місць» 2. Адміністратор обирає прив’язати чи відв’язати послугу від робочого місця 3. Адміністатор обирає перелік послуг та перелік робочих місць для обраної операції 4. Система виконує прив’язку чи відв’якзку послуг
Результат	Послуги відв’язані чи прив’язані до робочих місць
Виключення	Якщо адміністратора не має прав для внесення змін, система видає повідомлення про недостатні права

Таблиця 4.17 – Прецедент формування статистики

Назва	Формування статистики
Актори	Адміністратор
Опис	Адміністратор може генерувати звіти про роботу черг, включаючи кількість клієнтів та час очікування.
Передумова	Дані про роботу системи накопичені
Успішний сценарій	1. Адміністратор вибирає період для генерації звіту 2. Система генерує звіт із детальною статистикою 3. Адміністратор отримує файл зі звітом
Результат	Звіт успішно згенеровано та надано адміністратору
Виключення	Якщо дані недоступні або пошкоджені, система виводить повідомлення про неможливість генерації звіту

Висновки до розділу 4

У даному розділі був розроблений кожен із сценаріїв з урахуванням реальних потреб користувачів, таких як клієнти, працівники та адміністратори.

Прецеденти клієнтів спрямовані на забезпечення зручності використання, включаючи можливість створювати, редагувати чи видаляти попередні записи. Система враховує реальні ситуації, наприклад, коли обраний час недоступний, і пропонує альтернативи.

Прецеденти працівників зосереджені на управлінні чергами та клієнтами. Вони дозволяють створювати, редагувати й видаляти талони, викликати клієнтів і передавати їх між робочими місцями. Це підвищує ефективність роботи співробітників та мінімізує час простою.

Прецеденти адміністратора забезпечують гнучкість в управлінні системою, включаючи керування послугами, працівниками, робочими місцями та їх зв’язками. Адміністратор також може формувати звіти для оцінки ефективності системи, що полегшує її оптимізацію.

Важливим доповненням до описаних сценаріїв є врахування винятків, які демонструють реакцію системи на нестандартні ситуації, такі як відсутність доступних даних чи недостатні права користувачів. Це гарантує стабільну роботу системи та забезпечує позитивний користувацький досвід.

5 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Вибір технологій є одним із ключових етапів розробки програмного забезпечення, що безпосередньо впливає на продуктивність, масштабованість, витрати та терміни реалізації системи. Важливо підібрати стек технологій, що дозволить ефективно реалізувати поставлені функціональні та нефункціональні вимоги до системи управління чергами. Технологічний стек включає мови програмування, фреймворки, бази даних та інші інструменти, необхідні для створення сучасного веб-застосунку.

У цьому розділі будуть розглянуті ключові технології, що використовуватимуться для реалізації серверної та клієнтської частини системи, а також бази даних, з якою працюватиме система.

5.1 Тип та архітектура застосунку

Для цільового продукту було вирішено створити веб-додаток. Веб-додатки є ідеальним вибором для проєктів, що потребують публічного доступу через Інтернет. Такий підхід дозволяє створити універсальне рішення для різних платформ, оскільки доступ до системи можна отримати через будь-який пристрій з браузером. Це не потребує додаткових установок програмного забезпечення на пристроях користувачів і спрощує процес оновлення системи, оскільки зміни можна вносити централізовано на сервері. Серед основних переваг веб-застосунків:

– незалежність від платформи: користувачі можуть отримати доступ до системи з будь-якого пристрою, що підтримує браузер;

– легкість розгортання: оновлення веб-застосунків відбувається централізовано, що дозволяє миттєво вносити зміни в систему без необхідності перезавантаження чи оновлення клієнтської частини на пристрої користувача;

– простота обслуговування: адміністратори системи можуть легко керувати чергами та іншими параметрами через централізований інтерфейс управління.

Для реалізації було обрано клієнт-серверну архітектуру, яка чітко розподіляє завдання між клієнтською і серверною частинами [5]. Основна логіка архітектури полягає в тому, що клієнтська частина відповідає за взаємодію з користувачем, в той час як серверна частина виконує обробку даних, управління базою даних і реалізацію бізнес-логіки. Клієнт-серверна архітектура забезпечує наступні переваги:

– гнучкість системи: незалежна розробка клієнтських і серверних компонентів;

– масштабованість: можливість додавання серверів для обробки збільшених навантажень;

– стабільність: централізоване управління ресурсами для високої продуктивності.

На рисунку 5.1 зображений приклад даної архітектури.

Рисунок 5.1 – Архітектура системи

Архітектура сприяє простому масштабуванню, оптимізації серверів, а також розробці незалежних додатків для різних платформ. Компоненти системи функціонують автономно, що підвищує гнучкість і зручність у підтримці.

Для створення логіки веб-додатка обрано архітектурний шаблон MVC (Model-View-Controller), який реалізує принцип розподілу обов’язків і розділяє додаток на три взаємопов'язані компоненти: модель, представлення і контролер. Кожен з цих компонентів відіграє певну роль у забезпеченні чіткого розподілу обов'язків в системі і полегшує модульну розробку програмного забезпечення [6]:

– model: інкапсулює бізнес-логіку та структури даних, забезпечує доступ до бази даних і змінює інформацію відповідно до отриманих запитів;

– view: відповідає за відображення інформації з моделі та взаємодію користувачів із системою;

– controller: отримує дані від користувача, обробляє їх і синхронізує модель та представлення.

На рисунку 5.2 зображено діаграму взаємодії між компонентами шаблону MVC.

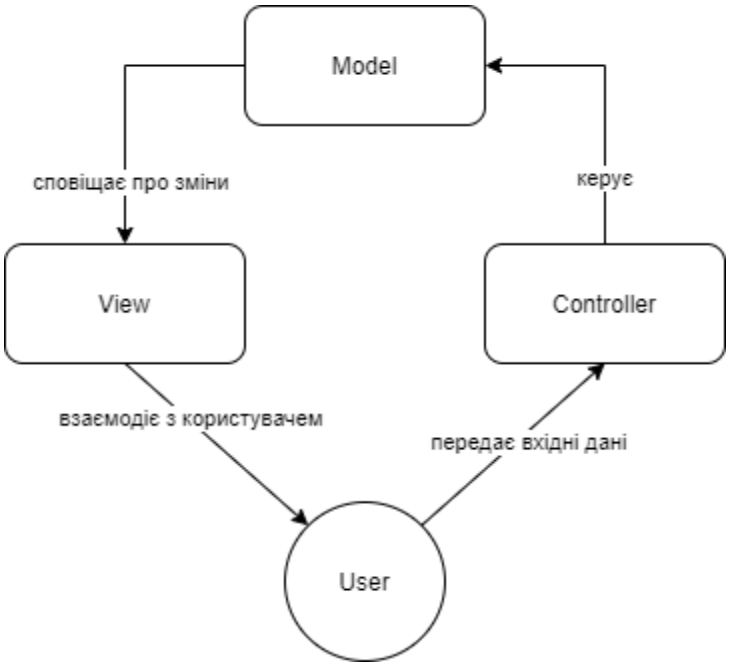

Рисунок 5.2 – Діаграма взаємодії між компонентами шаблону MVC

Використання MVC у середовищі розробки IntelliJ IDEA дозволяє ефективно організувати структуру додатка, розділивши його на окремі модулі, що спрощує тестування, рефакторинг та інтеграцію.

5.2 Мова програмування

Один із важливих кроків у розробці серверної частини веб-додатка – це вибір мови програмування. Мова програмування має бути високопродуктивною, безпечною та з підтримкою сучасних фреймворків і бібліотек для розробки.

5.2.1 Java

Java є широко визнаною у світі корпоративних рішень і пропонує високий рівень стабільності та безпеки. Java є мовою з багатою екосистемою бібліотек і фреймворків, таких як Spring, Hibernate та інші, що значно полегшує розробку складних серверних додатків.

Переваги Java:

– кросплатформеність: Java працює на будь-якій операційній системі, яка підтримує JVM, що робить її універсальною мовою для розробки серверних рішень;

– масштабованість: підтримка багатопоточності і можливість обробки великих обсягів даних робить Java ідеальним вибором для розробки систем, які можуть масштабуватися відповідно до зростання навантаження;

– безпека: вбудовані механізми управління пам'яттю та обробки винятків роблять Java однією з найбільш безпечних мов для роботи із чутливими даними;

– широка підтримка спільноти: величезна спільнота розробників Java забезпечує постійну підтримку та доступ до різноманітних бібліотек і рішень для реалізації майже будь-яких завдань.

Недоліки Java:

– складність: Java має досить складний синтаксис, що може вимагати більше часу для написання і підтримки коду у порівнянні з іншими сучасними мовами;

– вимоги до ресурсів: Java-додатки можуть бути ресурсомісткими, що може впливати на використання серверних ресурсів.

5.2.2 Python

Python є універсальною мовою, що також популярна для створення серверних рішень завдяки своїй простоті і швидкості розробки. Для цього вибору були враховані такі плюси Python, як простота синтаксису і велика кількість готових бібліотек для веб-розробки, таких як Django і Flask.

Переваги Python:

– простота і зрозумілість: python має легкий і читабельний синтаксис, що робить його чудовим вибором для швидкої розробки і прототипування;

– велика стандартна бібліотека: python пропонує безліч готових рішень для роботи з мережею, базами даних, веб-фреймворками і наукою про дані;

– широка підтримка спільноти: велика кількість бібліотек, таких як tensorflow для машинного навчання або django для веб-розробки, робить python універсальним інструментом.

Недоліки Python:

– низька продуктивність: як інтерпретована мова, python є значно повільнішою, ніж java, що може бути критичним для систем із високими вимогами до продуктивності.

– проблеми з масштабуванням: python менш ефективний для розробки великих розподілених систем із високими навантаженнями через обмеження його продуктивності.

5.2.3 Node.js (JavaScript)

Node.js – це серверна платформа, що базується на JavaScript і використовується для розробки веб-серверів та додатків реального часу. Node.js добре підходить для обробки одночасних запитів і широко застосовується для розробки легковагових і швидких серверних додатків.

Переваги Node.js:

– неблокуюча модель введення-виведення: node.js обробляє велику кількість запитів одночасно завдяки своїй подієво-орієнтованій архітектурі;

– швидкість виконання: оскільки node.js працює на движку v8 від google, він забезпечує швидке виконання javascript-коду;

– одна мова для фронтенду і бекенду: використання javascript на обох рівнях розробки спрощує процес і дозволяє легко керувати як клієнтськими, так і серверними аспектами програми.

Недоліки Node.js:

– однопоточна модель: node.js працює на одному потоці, що може стати обмеженням для високонавантажених систем із вимогою до паралельної обробки;

– менш стабільний для великих систем: хоча node.js підходить для невеликих застосунків, він може бути менш надійним для складних корпоративних систем порівняно з java.

5.3 Вибір фреймворку

5.3.1 Spring та Spring Boot

Spring – це потужний фреймворк для Java, який дозволяє розробляти комплексні корпоративні додатки. Spring пропонує багато готових рішень для роботи з безпекою, базами даних та іншими службами, що робить його одним із найбільш гнучких інструментів у розробці [7].

Spring Boot – це надбудова над Spring, яка автоматизує більшість налаштувань, спрощує розробку та прискорює процес створення Java-додатків [12].

Переваги Spring:

– гнучкість: spring дозволяє налаштовувати додаток відповідно до специфічних потреб проєкту;

– модульність: завдяки модульній архітектурі, можна використовувати лише ті компоненти, які дійсно потрібні для конкретного проєкту.

Недоліки Spring:

– складність: spring потребує багато конфігураційних файлів, що збільшує складність налаштування;

Переваги Spring Boot:

– простота використання: spring boot спрощує конфігурацію додатка, дозволяючи розробнику зосередитися на бізнес-логіці.

– швидке розгортання: завдяки вбудованим веб-серверам (tomcat, jetty), spring boot дозволяє швидко розгортати додатки.

Недоліки Spring Boot:

– обмежена гнучкість: через автоматизацію багатьох процесів, spring boot може бути менш гнучким для специфічних налаштувань у порівнянні зі звичайним spring.

5.3.2 Django (Python)

Django – це потужний фреймворк для веб-розробки на Python, який дозволяє швидко створювати веб-системи. Django забезпечує багато готових рішень для роботи з безпекою, авторизацією та базами даних, що робить його ідеальним для проєктів із швидкою розробкою.

Переваги Django:

– швидка розробка: django надає готові компоненти, що дозволяє швидко створювати додатки;

– зручна робота з базами даних: django включає в себе orm (object-relational mapping), що полегшує роботу з реляційними базами даних.

Недоліки Django:

– низька продуктивність: django, як і python, менш продуктивний у порівнянні з java для великих систем із високими вимогами до продуктивності.

5.3.3 Express (Node.js)

Express – це мінімалістичний веб-фреймворк для Node.js, який використовується для створення серверних додатків на основі JavaScript. Express забезпечує швидку розробку завдяки простоті налаштування та мінімуму коду.

Переваги Express:

– легкість налаштування: express потребує мінімум конфігурації, що дозволяє швидко почати розробку.

– висока продуктивність для невеликих додатків: express добре підходить для невеликих серверних застосунків, таких як веб-сервіси або api.

Недоліки Express:

– менша стабільність: express не забезпечує того рівня стабільності та безпеки, що потрібні для складних корпоративних рішень.

5.3 Вибір бази даних

5.3.1 MySQL

MySQL – це реляційна база даних, що забезпечує високу продуктивність, надійність і є одним із найпопулярніших рішень для зберігання даних у веб-застосунках. MySQL використовується в багатьох корпоративних рішеннях завдяки своїй стабільності та ефективності в роботі з великими обсягами даних[9].

Переваги MySQL:

– продуктивність: mysql забезпечує швидке оброблення транзакцій, що робить її ідеальним вибором для систем із високим навантаженням;

– широка підтримка: mysql легко інтегрується з java через jdbc або orm-фреймворки, такі як hibernate, що полегшує розробку.

Недоліки MySQL:

– обмежена підтримка нереляційних даних: mysql не підходить для зберігання неструктурованих даних, таких як json або xml.

5.3.2 PostgreSQL

PostgreSQL – це реляційна база даних із відкритим кодом, що забезпечує більші можливості порівняно з MySQL, зокрема підтримку складних типів даних і нереляційних структур (JSON, XML).

Переваги PostgreSQL:

– підтримка складних типів даних: postgresql дозволяє працювати з більш складними структурами даних, що робить її більш гнучкою.

– масштабованість: postgresql підтримує великі обсяги даних і добре підходить для систем із високими вимогами до складних запитів.

Недоліки PostgreSQL:

– складність налаштування: postgresql вимагає більше конфігурацій та налаштувань, що може ускладнити розробку та підтримку.

5.3.3 MongoDB

MongoDB – це нереляційна база даних (NoSQL), що використовується для зберігання неструктурованих даних, таких як JSON-документи. MongoDB часто використовується в системах, що потребують гнучкості в роботі з даними.

Переваги MongoDB:

– гнучкість у роботі з даними: mongodb дозволяє зберігати неструктуровані дані, що робить її ідеальною для проєктів, де структура даних може змінюватись.

Недоліки MongoDB:

– відсутність підтримки реляційних запитів: mongodb не підходить для систем, де необхідно використовувати складні реляційні запити.

Висновки до розділу 5

У даному розділі було розглянуто технології, які будуть використовуватись при розробці системи управління чергами. Після ретельного аналізу різних варіантів було обрано та обґрунтовано використання таких технологій:

– Java як основна мова програмування для серверної частини, що забезпечує високу продуктивність, стабільність і безпеку.

– Spring Boot як фреймворк для серверної частини для автоматизації налаштувань і швидкого створення додатків.

– JavaScript для клієнтської частини, що дозволяє створити інтерактивний веб-інтерфейс.

– MySQL як реляційна база даних для зберігання інформації, що забезпечує стабільність і високу продуктивність системи.

Ці технології дозволяють розробити масштабовану, надійну та продуктивну систему, яка відповідатиме функціональним і нефункціональним вимогам проєкту. Використання цих інструментів гарантує, що система буде здатна підтримувати високе навантаження, масштабуватись і забезпечувати безпеку даних клієнтів.

6 СТРУКТУРНА СХЕМА СИСТЕМИ

Структурна схема системи управління чергами є важливим компонентом, який ілюструє взаємодію між різними складовими системи для забезпечення її ефективного функціонування. Основною ідеєю є реалізація багаторівневої архітектури, що дозволяє досягти модульності та забезпечити гнучкість, масштабованість і зручність для подальшого розвитку та підтримки системи. Завдяки такій архітектурі, кожен компонент системи виконує чітко визначені завдання, що забезпечують стабільну роботу та зручність для користувачів на всіх рівнях взаємодії[10].

Система складається з трьох основних рівнів, кожен з яких виконує свою роль:

– клієнтський рівень (frontend) — цей рівень відповідає за взаємодію з користувачем через веб-інтерфейс. він забезпечує відображення даних і дозволяє користувачам виконувати дії, такі як реєстрація на прийом, перегляд черги та управління запитами;

– серверний рівень (backend) — основний рівень для обробки запитів від користувачів, управління бізнес-логікою, взаємодії з базою даних і контролю за виконанням функцій системи, таких як створення талонів, виклик клієнтів і завершення сеансів обслуговування;

– рівень даних (database) — цей рівень відповідає за зберігання та обробку всіх даних системи. використовується реляційна база даних, що зберігає всі сутності, такі як клієнти, працівники, черги, послуги та робочі місця;

У додатку В наведено структурну схему взаємодії між рівнями архітектури.

6.1 Компоненти системи

6.1.1 Клієнтська частина

Клієнтська частина є ключовим елементом для взаємодії користувачів із системою. Вона надає інтерфейс для реєстрації клієнтів, працівників та

адміністраторів, а також для виконання основних операцій, таких як запис на прийом, перегляд черги, виклик клієнтів до робочих місць і завершення сеансів. Клієнтська частина реалізована як веб-інтерфейс, що дозволяє користувачам з різним рівнем доступу взаємодіяти з системою через веб-браузер[13].

Основна роль клієнтської частини полягає в тому, щоб надавати користувачам можливість:

– реєструватися для отримання послуг;

– вибирати час та послугу;

– переглядати статус черги;

– отримувати повідомлення про виклик на обслуговування.

Працівники можуть створювати нові талони, викликати клієнтів до робочих місць, передавати їх на інші місця або завершувати сеанси обслуговування. Адміністратори мають доступ до додаткових функцій, таких як управління послугами, працівниками та генерування звітів. Клієнтська частина системи взаємодіє з серверною частиною через RESTful API [8], використовуючи HTTP-запити. Це дозволяє отримувати та обробляти відповіді у форматі JSON[15-16], що забезпечує динамічне оновлення інтерфейсу без необхідності перезавантаження сторінки.

6.1.2 Серверна частина

Серверна частина системи управління чергами забезпечує виконання основної бізнес-логіки та відповідає за обробку запитів, що надходять від клієнтів, працівників та адміністраторів. Вона забезпечує взаємодію між базою даних і клієнтською частиною, обробляючи запити на створення талонів, оновлення черг, виклик клієнтів до робочих місць, перенаправлення клієнтів та завершення сеансів.

Сервер взаємодіє з базою даних, яка зберігає інформацію про клієнтів, послуги, черги, працівників та робочі місця. База даних забезпечує швидкий доступ до необхідної інформації, зберігання і постійну актуалізацію даних. У випадку помилок або невірно виконаних операцій сервер надає відповідні повідомлення або

інформує про необхідність коригування ситуації. Завдяки цьому серверна частина є важливою для забезпечення надійної роботи всієї системи.

Сервер також реалізує безпеку даних, забезпечуючи автентифікацію користувачів та захист персональних даних відповідно до чинного законодавства. Використовуються сучасні методи шифрування і захисту для зберігання конфіденційної інформації[14,17-18].

6.1.3 База даних

База даних є основою для зберігання та організації всіх даних системи. Вона використовує реляційну модель, де дані зберігаються у таблицях, що взаємопов'язані між собою. Наприклад, таблиця queue зберігає інформацію про талони, статуси черг, прив'язку до конкретних робочих місць і послуг. Таблиця workplace містить дані про робочі місця, а таблиця service_entity зберігає відомості про надані послуги. Завдяки цій структурі, база даних забезпечує ефективне управління інформацією, що дозволяє підтримувати безперебійну роботу всіх процесів.

База даних також підтримує ACID-транзакції, що гарантують цілісність даних навіть у разі помилок або збоїв. Це важливо для забезпечення стабільності та надійності роботи системи. Крім того, база даних реалізує механізми оптимізації запитів для швидкого доступу до великої кількості даних, що особливо важливо для масштабованості системи.

6.1.4 API

API серверної частини є інтерфейсом для взаємодії між клієнтською частиною та сервером. Воно реалізоване через RESTful сервіси, які приймають запити через HTTP і відповідають у форматі JSON. Це дозволяє системі бути гнучкою, масштабованою та сумісною з різними типами клієнтських додатків.

Основні функції API включають управління послугами, робочими місцями, працівниками, чергами та звітами.

Зокрема, основні ендпоїнти API включають:

– /services — для управління послугами (створення, редагування, видалення);

– /workplaces — для управління робочими місцями;

– /reports — для формування звітів про ефективність роботи;

– /employees — для управління працівниками;

– /queues — для обробки черг (створення, виклик клієнтів, передача, завершення сеансу);

– /appointments — для обробки попередніх записів клієнтів.

API дозволяє серверу приймати запити від клієнтської частини, обробляти їх, взаємодіяти з базою даних та відправляти результати назад до клієнта. Це забезпечує інтерактивність та швидкість обробки даних у реальному часі.

6.2 Взаємодія компонентів

Взаємодія між серверною та клієнтською частинами відбувається через чітко визначену структуру запитів та відповідей. Клієнтська частина надсилає запити на сервер для виконання різних операцій, таких як створення талонів, виклик клієнтів або оновлення черги. Сервер обробляє ці запити, взаємодіє з базою даних для отримання або збереження необхідної інформації та повертає результат у форматі JSON.

Процес взаємодії виглядає так:

– клієнтські запити — користувачі, використовуючи веб-інтерфейс, надсилають запити на сервер (наприклад, для створення талону, виклику клієнта);

– обробка запитів на сервері — сервер отримує запит і передає його відповідному сервісу для обробки;

– взаємодія з базою даних — необхідні дані зберігаються або отримуються з бази даних за допомогою sql-запитів;

– відповідь клієнту — сервер повертає відповідь у форматі json, яку клієнтська частина обробляє і використовує для оновлення інтерфейсу.

Всі ці етапи забезпечують автоматизацію процесів управління чергами, що підвищує ефективність обслуговування клієнтів і зменшує навантаження на персонал.

Висновок до розділу 6

У даному розділі було розроблено структурну схему системи управління чергами, яка чітко відображає взаємодію основних компонентів, що забезпечують ефективну роботу всієї системи. Клієнтська частина (frontend), серверна частина (backend) та база даних (database) працюють у тісній взаємодії, забезпечуючи коректну обробку запитів користувачів і ефективне зберігання даних.

Клієнтська частина системи, як основний інтерфейс для користувачів, надає всі необхідні інструменти для реєстрації клієнтів на прийом, перегляду черги, виклику клієнтів до робочих місць і завершення сеансів обслуговування. Використовуючи RESTful API для взаємодії з сервером, клієнтська частина забезпечує динамічне оновлення інтерфейсу без необхідності перезавантаження сторінки, що підвищує зручність використання системи.

Серверна частина забезпечує обробку всіх запитів від користувачів і виконує важливу бізнес-логіку. Вона здійснює обробку даних, передає їх до бази даних для зберігання та повертає результат у вигляді JSON-формату, який обробляється клієнтською частиною. Окрім того, серверна частина реалізує важливі функції безпеки, забезпечуючи захист персональних даних користувачів, а також підтримує високий рівень надійності та стабільності системи.

База даних, як основа для зберігання даних, ефективно організовує інформацію про клієнтів, працівників, послуги та черги, що дозволяє швидко здійснювати доступ до необхідної інформації.

7 ER-ДІАГРАМА (ТАБЛИЦЯ БАЗ ДАНИХ)

База даних інформаційної системи управління чергою є основою, яка забезпечує збереження, обробку та доступ до даних, що використовуються системою. Її структура побудована з урахуванням функціональних і нефункціональних вимог до системи, забезпечуючи високу продуктивність, масштабованість і надійність. База даних підтримує основні процеси системи, такі як реєстрація клієнтів, створення черг, розподіл клієнтів між робочими місцями, керування працівниками та формування звітів.

7.1 Сутності

Основними сутностями у розробленій системі є:

– попередній запис (Appointment);

– клієнт (Client);

– працівник (Employee);

– одноразовий пароль (Otp_code_entity);

– черга (Queue);

– звіт (Report);

– послуга (Service_entity);

– робоче місце (Workplace);

– послуга-робоче місце (ServiceWorkplace).

Сутність попереднього запису зберігає дані про клієнтів, які заздалегідь зареєструвалися для отримання послуг. Дана сутність містить посилання на клієнта та послугу і час на який клієнт зареєстрував попередній запис.

Клієнт рахується однією з ключових сутностей. Вона забезпечує зберігання інформації про фізичних осіб, які користуються системою. Сутність містить поля для ідентифікації клієнта, його номер телефону і роль яка допомагає системі визначити його функціонал.

Сутність працівник використовується для зберігання інформації про співробітників системи. Вона містить логін, пароль, роль працівника та посилання на його робоче місце. Кожен працівник має одну з двох доступних ролей: працівник або адміністратор. Завдяки цьому працівники отримають свій відповідний функціонал, а також дані згідно власного робочого місця.

Для авторизації клієнтів за допомогою одноразових паролів (OTP) використовується сутність otp_code_entity. Вона зберігає OTP-коди, час їх закінчення, а також телефонні номери, до яких вони прив’язані. Це забезпечує безпечний доступ до системи клієнтам.

Сутність queue є центральною для реалізації функціоналу системи. Вона зберігає інформацію про створені талони, статус обслуговування, час створення талону та початку обслуговування. Вона також пов’язана з послугою, клієнтом і робочим місцем.

Сутність report використовується для аналізу роботи системи. Вона зберігає ключові статистичні показники, такі як середній, мінімальний та максимальний час очікування, а також загальну кількість талонів за визначений період.

Сутність service_entity містить інформацію про послуги, які доступні для клієнтів. Вона включає назву послуги та її опис. Ця інформація використовується для створення талонів і попередніх записів.

Сутність workplace представляє фізичні робочі місця, де працівники обслуговують клієнтів. Вона включає унікальний ідентифікатор і назву робочого місця.

Сутність serviceWorkplace містить інформацію про зв’язок конкретної послуги і робочого місця. Ця інформація використовується для створення талонів черги.

7.2 Опис таблиць

Детальна структура бази даних описана в таблицях 7.1 – 7.9, де наведено назви колонок, поля та типи даних, внутрішні і також зовнішні ключі. Схему бази даних наведено на діаграмі Г.

Таблиця 7.1 – Поля сутності appointment

Назва поля	Тип даних
Id (PK)	BIGINT
service_entity_id (FK)	BIGINT
client_id (FK)	BIGINT
appointment_time	DATETIME

Таблиця 7.2 – Поля сутності client

Назва поля	Тип даних
id (PK)	BIGINT
username	VARCHAR(255)
phone_number	VARCHAR(255)
role	VARCHAR(255)

Таблиця 7.3 – Поля сутності otp_code_entity

Назва поля	Тип даних
id (PK)	BIGINT
phone_number	VARCHAR(255)
otp_code	VARCHAR(255)
expiration_time	DATETIME

Таблиця 7.4 – Поля сутності report

Назва поля	Тип даних
id (PK)	BIGINT
average_waiting_time	BIGINT
max_waiting_time	BIGINT
min_waiting_time	BIGINT
total_tickets	BIGINT

Таблиця 7.5 – Поля сутності service_entity

Назва поля	Тип даних
id (PK)	BIGINT
service_name	VARCHAR(255)
service_description	VARCHAR(255)

Таблиця 7.6 – Поля сутності workplace

Назва поля	Тип даних
id (PK)	BIGINT
workplace_name	VARCHAR(255)

Таблиця 7.7 – Поля сутності employee

Назва поля	Тип даних
id (PK)	BIGINT
username	VARCHAR(255)
password	VARCHAR(255)
role	VARCHAR(255)
workplace_id (FK)	BIGINT

Таблиця 7.8 – Поля сутності queue

Назва поля	Тип даних
id (PK)	BIGINT
workplace_id (FK)	BIGINT
ticket_number	INT
status	VARCHAR(255)
service_id (FK)	BIGINT
created_at	DATETIME(6)
started_at	DATETIME(6)

Таблиця 7.9 – Поля сутності serviceWorkplace

Назва поля	Тип даних
id (PK)	BIGINT
workplace_id (FK)	BIGINT
service_id (FK)	BIGINT

Висновки до розділу 7

У даному розділі була детально описана структура бази даних, яка є основою для роботи системи управління чергами. Розроблена база даних організована таким чином, щоб ефективно підтримувати всі основні функціональні можливості системи: від реєстрації клієнтів та створення черг до управління працівниками та формування звітів. Всі сутності та їх взаємозв’язки спроектовані з урахуванням вимог до масштабованості, безпеки та ефективності зберігання даних.

Основні сутності, такі як клієнт (Client), працівник (Employee), послуга (Service_entity), черга (Queue), робоче місце (Workplace), та послуга-робоче місце (ServiceWorkplace) взаємодіють між собою через чітко визначені зовнішні ключі, що забезпечує цілісність і зв'язок даних. Важливою частиною системи є сутність OTP-код (Otp_code_entity), яка використовує одноразові паролі для забезпечення

безпеки клієнтів при авторизації, що відповідає вимогам безпеки персональних даних.

Завдяки детально спроектованій базі даних система здатна обробляти великі обсяги інформації без втрати ефективності, зберігаючи дані про черги, клієнтів, працівників і послуги, а також підтримувати необхідні зв'язки між сутностями. Кожна сутність, зокрема звіти (Report), допомагає адміністраторам аналізувати ефективність роботи системи за різними параметрами, що є важливою складовою для подальшого удосконалення процесів управління чергами.

Розроблена база даних підтримує високу масштабованість, що дозволяє системі ефективно працювати навіть при великій кількості клієнтів, працівників і послуг. Завдяки використанню реляційної моделі зберігання даних і зв'язків між сутностями, система забезпечує правильне управління чергами та попереднім записом.

Таким чином, структура бази даних гарантує не лише коректне виконання всіх функцій системи, але й дозволяє забезпечити гнучкість для розширення її можливостей в майбутньому.

8 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

Для розробки інформаційної системи управління чергами державних установ використано багаторівневу архітектуру, яка включає веб-ресурси, бібліотеки та структурований вихідний код. Код організовано на основі рівнів доступу до даних, бізнес-логіки та веб-компонентів. Архітектура базується на принципах абстракції та модульності, що сприяє простоті підтримки та розширення функціональності системи.

Серверна частина системи описується через класифікацію її основних компонентів. Детальну діаграму класів розробленої системи наведено в додатку Д.

8.1. Компоненти рівня доступу до даних

Рівень доступу до даних виконує операції збереження, пошуку та обробки інформації в базі даних. Він містить:

– сутності — класи, що описують модель даних і об'єкти предметної області;

– репозиторії — компоненти, які надають абстрактний інтерфейс для роботи з базою даних і забезпечують зручність доступу до основного сховища.

Цей рівень ізолює бізнес-логіку від деталей реалізації збереження даних, що забезпечує гнучкість та масштабованість системи.

8.1.1 Сутності

Перелік сутностей розробленої системи зображений на рисунку 8.1.

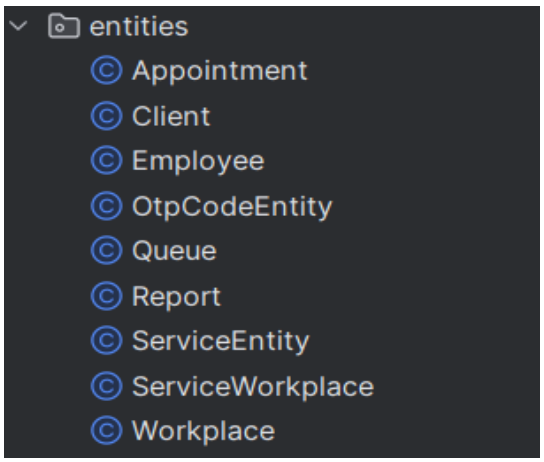

Рисунок 8.1 – Сутності системи

Клас Appointment описує запис клієнта на обслуговування в певному сервісі. Він містить наступні поля:

– id: унікальний ідентифікатор запису;

– serviceEntity: посилання на сервіс, на який здійснюється запис (пов'язано з класом ServiceEntity);

– client: посилання на клієнта, який записується (пов'язано з класом Client);

– appointmentTime: час запису на обслуговування.

Даний клас має також наступні анотації:

– @Entity — вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних;

– @Table — зазвичай вказується назва таблиці, з якою пов’язана сутність (у нашому випадку це за замовчуванням таблиця Appointment);

– @Id — визначає поле id як первинний ключ сутності;

– @GeneratedValue(strategy = GenerationType.IDENTITY) — визначає стратегію генерації значень первинного ключа;

– @ManyToOne і @JoinColumn — встановлюють зв’язок між класами Appointment і іншими сутностями, такими як ServiceEntity та Client, через зовнішні ключі.

Клас Client описує клієнта, який може записатися на обслуговування. Він містить наступні поля:

– id: унікальний ідентифікатор клієнта;

– username: ім'я користувача для входу в систему (для працівників та адміністраторів);

– phoneNumber: телефонний номер клієнта (для користувачів);

– role: роль користувача (наприклад, "user", "employee", "admin").

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних.

– @Id: визначає поле id як первинний ключ сутності.

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

Клас Employee описує працівника системи, зокрема його дані для доступу та прив'язку до робочого місця. Він містить наступні поля:

– id: унікальний ідентифікатор працівника.

– username: ім'я користувача працівника.

– password: пароль для доступу.

– role: роль працівника (наприклад, адміністратор або звичайний працівник).

– workplace: посилання на робоче місце, де працює працівник (пов'язано з класом Workplace).

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних.

– @Id: визначає поле id як первинний ключ сутності.

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

– @ManyToOne та @JoinColumn: визначають зв'язок між працівником та робочим місцем через зовнішній ключ.

Клас OtpCodeEntity описує одноразові паролі (OTP), які використовуються для автентифікації клієнтів. Він містить наступні поля:

– id: унікальний ідентифікатор OTP.

– phoneNumber: номер телефону для якого генерується OTP.

– otpCode: сам OTP код.

– expirationTime: час закінчення дії OTP.

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних.

– @Id: визначає поле id як первинний ключ сутності.

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

Клас Queue описує чергу на обслуговування в системі, включаючи інформацію про сервіс, робоче місце, статус черги і часи створення та початку обслуговування. Він містить наступні поля:

– id: унікальний ідентифікатор черги.

– serviceEntity: посилання на сервіс, для якого створена черга (пов'язано з класом ServiceEntity).

– workplace: посилання на робоче місце, де обслуговується клієнт (пов'язано з класом Workplace).

– ticketNumber: номер квитка у черзі.

– status: статус черги (наприклад, "ACTIVE", "IN_PROGRESS", "COMPLETED").

– createdAt: час створення квитка.

– startedAt: час початку обслуговування.

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних.

– @Id: визначає поле id як первинний ключ сутності.

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

– @ManyToOne та @JoinColumn: визначають зв’язок між чергою і іншими сутностями (наприклад, ServiceEntity та Workplace) через зовнішні ключі.

– @Enumerated(EnumType.STRING): визначає, що значення статусу буде зберігатися як рядок.

Клас Report описує статистичні дані про черги, зокрема часи очікування. Він містить наступні поля:

– id: унікальний ідентифікатор звіту.

– totalTickets: загальна кількість квитків.

– averageWaitingTime: середній час очікування.

– maxWaitingTime: максимальний час очікування.

– minWaitingTime: мінімальний час очікування.

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних.

– @Id: визначає поле id як первинний ключ сутності.

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

Цей клас використовується для створення звітів та збору статистичних даних, які допомагають аналізувати ефективність обслуговування.

Клас ServiceEntity використовується для для збереження даних про доступні послуги в системі. Він містить наступні поля:

– id: унікальний ідентифікатор сервісу.

– serviceName: назва сервісу.

– serviceDescription: опис сервісу.

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних.

– @Id: визначає поле id як первинний ключ сутності.

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

Клас Workplace використовується для управління даними робочих місць, де надаються послуги. Він містить наступні поля:

– id: унікальний ідентифікатор робочого місця.

– workplaceName: назва робочого місця.

Даний клас має наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних;

– @Id: визначає поле id як первинний ключ сутності;

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа.

Клас ServiceWorkplace моделює зв’язок між послугами (ServiceEntity) та робочими місцями (Workplace). Даний клас містить наступні поля:

– id: унікальний ідентифікатор зв’язку;

– service: зв’язок з сутністю ServiceEntity;

– workplace: зв’язок з сутністю Workplace.

Також даний клас містить наступні анотації:

– @Entity: вказує, що клас є сутністю JPA і буде мапуватися на таблицю в базі даних;

– @Id: визначає поле id як первинний ключ сутності;

– @GeneratedValue(strategy = GenerationType.IDENTITY): визначає стратегію генерації значень первинного ключа;

– @Table: name = "service_workplace": задає ім'я таблиці в базі даних.

– @uniqueConstraints: забезпечує, що кожна пара service_id і workplace_id є унікальною.

Кожен з вище описаних класів надає методи getter та setter для системи, щоб та могла модифікувати значення полів.

8.1.2 Репозиторії

У проєкті використовується низка репозиторіїв для взаємодії з базою даних. Перелік інтерфейсів репозиторіїв зображено на рисунку 8.2.

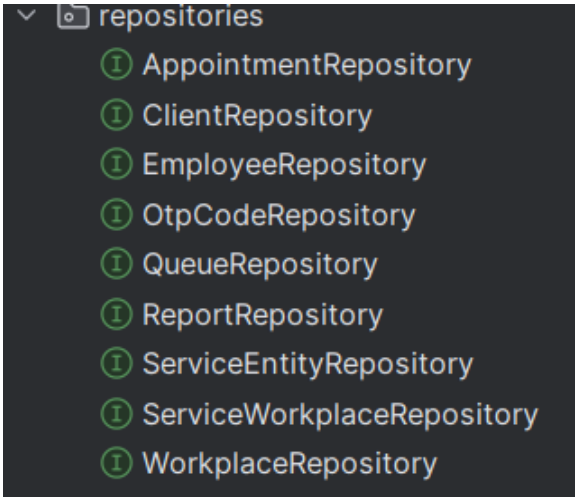

Рисунок 8.2 – Репозиторії системи

Усі ці репозиторії є інтерфейсами з анотацією @Repository, що позначає їх як частини репозиторного рівня Spring. Вони наслідують JpaRepository<Сутність, Тип> для надання базових функцій, таких як створення, зчитування, оновлення та видалення (CRUD). Кожен репозиторій працює з певною сутністю, а тип даних визначає формат первинного ключа.

AppointmentRepository використовується для управління записами на прийом. Цей репозиторій містить наступні методи:

– findByClientId(Long clientId) — повертає список записів, пов'язаних із заданим клієнтом;

– findByAppointmentTimeAndServiceEntity(LocalDateTime appointmentTime, ServiceEntity serviceEntity) — знаходить записи за часом і типом послуги;

– findByAppointmentTime(LocalDateTime appointmentTime) — пошук усіх записів за певною датою і часом.

ClientRepository дозволяє виконувати операції з даними клієнтів. Цей репозиторій містить наступні методи:

– findByUsername(String username) — знаходить клієнта за його іменем користувача;

– findByPhoneNumber(String phoneNumber) — знаходить клієнта за номером телефону.

EmployeeRepository забезпечує роботу з інформацією про працівників та має наступні методи:

– findByUsername(String username) — пошук працівника за його іменем користувача;

– removeEmployeeByUsername(Employee employee) — видалення працівника;

– findByUsernameAndPassword(String username, String password) — перевірка облікових даних працівника.

OtpCodeRepository обробляє OTP-коди (одноразові паролі), його методи наступні:

– findByPhoneNumber(String phoneNumber) — пошук OTP-коду, прив'язаного до конкретного номера телефону.

QueueRepository застосовується для роботи із записами в черзі. У даному репозиторії використовуються наступні методи:

– findByWorkplaceIdAndStatus(Long workplaceId, Queue.QueueStatus status) — отримання записів для конкретного робочого місця із заданим статусом;

– findFirstByWorkplaceIdAndStatus(Long workplaceId, Queue.QueueStatus queueStatus) — знаходить перший запис у черзі для певного робочого місця;

– findByStatus(Queue.QueueStatus status) — повертає список записів із заданим статусом.

– findByCreatedAtBetween(LocalDateTime startDate, LocalDateTime endDate) — знаходить записи, створені в зазначений часовий інтервал.

ServiceWorkplaceRepository використовується для роботи з існуючими зв’язками послуг і робочих місць. Даний репозиторій містить наступні методи:

– findByServiceId(Long serviceId) – Отримує список усіх зв’язків, пов’язаних із конкретною послугою (service_id);

– findByWorkplaceId(Long workplaceId): Отримує список усіх зв’язків, пов’язаних із конкретним робочим місцем (workplace_id);

– findByServiceIdAndWorkplaceId(Long serviceId, Long workplaceId): шукає один зв'язок між послугою та робочим місцем;

– findAllByServiceIdAndWorkplaceId(Long serviceId, Long workplaceId): пошук усіх зв’язків послуг з робочими місцями.

Репозиторії ReportRepository, ServiceEntityRepository, WorkplaceRepository не мають додаткових методів і виконують стандартні CRUD операції.

8.2 Компоненти рівня бізнес-логіки

Компоненти рівня бізнес-логіки забезпечують реалізацію ключових бізнес-функцій у програмному забезпеченні. Вони відповідають за виконання бізнес-правил, обробку даних, їх валідацію, а також інтеграцію та координацію дій між різними частинами системи. Головна мета цих компонентів – створити зручний та структурований спосіб взаємодії між рівнями представлення та даних.

Бізнес-логіка обробляється сервісними компонентами, які забезпечують розподіл завдань між окремими класами. Сервіси виступають посередниками між даними та інтерфейсами, гарантують модульність та полегшують підтримку системи.

8.2.1 Основні сервіси

Перелік основних сервісів розробленої системи зображений на рисунку 8.3.

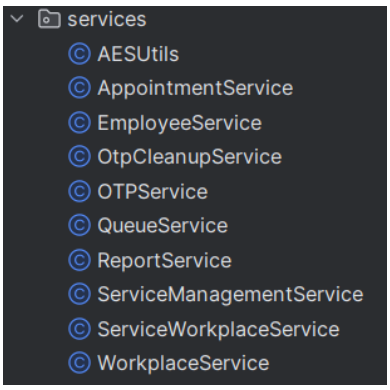

Рисунок 8.3 – Сервіси системи

Сервіс AppointmentService відповідає за управління записами клієнтів на послуги. Основні методи цього класу:

– getAppointmentsByClient(Long clientId) — отримує всі записи клієнта за його ID. Метод взаємодіє з репозиторієм AppointmentRepository та повертає список записів для конкретного клієнта;

– updateAppointment(Long appointmentId, Long clientId, LocalDateTime newAppointmentTime) — дозволяє редагувати час запису. Перед зміною перевіряється, чи належить запис вказаному клієнту;

– deleteAppointment(Long appointmentId, Long clientId) — видаляє запис, перевіряючи, чи належить він вказаному клієнту;

bookAppointment(ServiceEntity service, Client client, LocalDateTime appointmentTime) — дозволяє клієнту забронювати час для послуги. Проводиться перевірка, чи є це будній день, чи вибраний час не є минулим, а також чи не заброньовано цей час для послуги.

Сервіс EmployeeService керує операціями з Employee (співробітниками) і має наступні функції:

– findAll() — повертає список усіх співробітників;

– findById(Long id) — знаходить співробітника за ID;

– save(Employee employee) — зберігає нового співробітника;

– delete(Long id) — видаляє співробітника за ID.

Сервіс OTPService призначений для генерації та перевірки одноразових паролів (OTP). Його основні функції:

– generateOTP() — генерує 6-значний OTP;

– encryptAndLogOTP(String otpCode) — шифрує OTP перед збереженням в базу даних і логуванням;

– saveOtp(String phoneNumber, String otpCode) — зберігає OTP в зашифрованому вигляді в базі даних з часом дії 5 хвилин;

– verifyOtp(String phoneNumber, String otp) — перевіряє OTP, розшифровуючи його та порівнюючи з введеним значенням. Якщо OTP співпадає та не втратив термін дії, воно видаляється з бази.

– OtpCleanupService — це сервіс, що відповідає за очищення зберігання застарілих одноразових паролів (OTP) в базі даних. Оскільки OTP мають обмежений термін дії, необхідно регулярно видаляти їх після того, як цей термін закінчився. Основні функції:

– deleteExpiredOtps() — цей метод автоматично виконується кожні 5 хвилин завдяки використанню анотації @Scheduled. Він шукає всі OTP, що зберігаються в базі даних, та перевіряє, чи не закінчився їхній термін дії. Якщо термін дії OTP вже завершений, ці OTP видаляються з бази даних. Для визначення терміну дії використовуються поля expirationTime в об'єктах OtpCodeEntity. Видалення відбувається через метод deleteAll(), що ефективно очищає базу від старих записів.

AESUtils — це допоміжний клас для роботи з шифруванням та дешифруванням даних за допомогою алгоритму AES[17] (Advanced Encryption Standard). Він містить методи для генерації ключів, шифрування та дешифрування OTP кодів. Основні функції даного класу наступні:

– generateKey() — цей метод генерує випадковий 128-бітний секретний ключ для шифрування та дешифрування даних, використовуючи алгоритм AES. Ключ генерується за допомогою KeyGenerator, який налаштовується на 128 біт (AES-128);

– encrypt() — метод, який шифрує OTP код з використанням заданого секретного ключа. Він приймає OTP код у вигляді рядка та ключ для шифрування, потім використовує Cipher для перетворення даних у

зашифрований вигляд. Результатом є зашифрований OTP, який кодується у формат Base64, щоб його можна було безпечно зберігати;

– decrypt() — метод, який дешифрує OTP код, використовуючи той самий секретний ключ. Спочатку дані декодуються з Base64, після чого за допомогою Cipher виконується зворотна операція шифрування, і результат повертається як звичайний рядок.

Сервіс QueueService керує чергою та обслуговуванням клієнтів. Основні методи:

– autoMoveAppointmentsToQueue() — кожні 1 хвилину (через @Scheduled) автоматично переносить записи на поточний день у чергу;

– moveAppointmentsToQueue(LocalDate date) — переносить записи на вказану дату в чергу. Перевіряється наявність робочих місць та генерується новий талон;

– getCurrentQueue(Long workplaceId) — отримує поточну чергу для вказаного робочого місця;

– getInProgressTickets() — отримує талони, що знаходяться в процесі виконання;

– getCurrentClient(Long workplaceId) — отримує клієнта, який обслуговується на вказаному робочому місці;

– createTicket(Long serviceId, Long workplaceId) — створює новий талон на послугу для робочого місця;

– updateTicket(Long ticketId, Long newServiceId, Long newWorkplaceId) — оновлює інформацію про талон (послугу або робоче місце);

– deleteTicket(Long ticketId) — видаляє талон;

– callNextClient(Long workplaceId) — переносить клієнта з черги в статус "в обробці";

– transferClient(Long queueId, Long newWorkplaceId) — переносить клієнта з одного робочого місця на інше;

– completeSession(Long queueId) — завершення сесії, зміна статусу талона на "завершено";

– findLeastLoadedWorkplaceForService(Long serviceId) – вибирає робоче місце з мінімальним навантаженням для заданої послуги;

– isWorkplaceOverloaded(Long workplaceId, int limit) – перевіряє, чи не перевищено встановлений ліміт талонів для робочого місця.

Сервіс ReportService відповідає за створення звітів та має наступні функції:

– generateReport(LocalDateTime startDate, LocalDateTime endDate) — генерує звіт за вказаний період, підраховує загальну кількість талонів, середній, максимальний та мінімальний час очікування клієнтів;

– deleteReport(Long reportId) — видаляє звіт за ID;

Сервіс ServiceManagementService займається керуванням послугами та відповідає за наступні функції:

– getAllServices() — повертає список усіх послуг;

– addService(ServiceEntity serviceEntity) — додає нову послугу;

– updateService(Long id, ServiceEntity serviceEntity) — оновлює дані про послугу за її ID;

– deleteService(Long id) — видаляє послугу за ID.

Сервіс WorkplaceService відповідає за управління робочими місцями. Його функції є наступними:

– findAll() — повертає список усіх робочих місць;

– save(Workplace workplace) — додає нове робоче місце;

– update(Long id, Workplace workplace) — оновлює інформацію про робоче місце;

– delete(Long id) — видаляє робоче місце;

– findById(Long id) — знаходить робоче місце за ID.

Сервіс ServiceWorkplaceService відповідає за управління зв'язками між послугами та робочими місцями. Основні функції сервісу:

– getWorkplacesForService(Long serviceId): повертає список робочих місць, які можуть виконувати обрану послугу;

– getServicesForWorkplace(Long workplaceId): повертає список послуг, які доступні для виконання на вказаному робочому місці.

– linkServiceToWorkplace(Long serviceId, Long workplaceId): додає зв'язок між послугою та робочим місцем. Також дана функція перевіряє чи вже існує зв’язок і якщо він відсутній то створює новий і зберігає його у базі даних;

– unlinkServiceFromWorkplace(Long serviceId, Long workplaceId): видаляє всі зв'язки між вказаною послугою та робочим місцем.

8.2.2 Додаткові сервіси

Для реалізації бізнес-логіки та підтримки основних функцій системи було впроваджено низку додаткових сервісів. Кожен із них відіграє важливу роль у забезпеченні роботи системи, від обробки даних і безпеки до генерації користувацького інтерфейсу. Додаткові сервіси, які дозволяють реалізувати розроблену бізнес-логіку:

– сервіс рендерингу HTML;

– сервіс доступу до даних;

– веб сервіс;

– сервіс безпеки.

Сервіс рендерингу HTML реалізовано за допомогою шаблонізатора Thymeleaf, який інтегрується через залежність spring-boot-starter-thymeleaf. Цей сервіс дозволяє:

– створювати динамічні html-сторінки, адаптовані під дані, отримані із серверу;

– відокремлювати логіку представлення від бізнес-логіки, що спрощує підтримку і розвиток проєкту;

– легко інтегрувати сторінки з іншими сервісами, такими як веб-сервіс та сервіс доступу до даних.

Завдяки Thymeleaf, система забезпечує сучасний і гнучкий користувацький інтерфейс, що зручно відображає динамічну інформацію, зокрема стан черги, дані користувачів чи OTP-коди.

Доступ до даних системи реалізовано через Spring Data JPA, підключеного за допомогою залежності spring-boot-starter-data-jpa. Основні функції цього сервісу включають:

– роботу з базою даних за допомогою orm (object-relational mapping), що спрощує інтеракцію між java-класами та таблицями бази даних;

– легке виконання crud-операцій завдяки репозиторіям (jparepository);

– можливість створення складних запитів за допомогою спеціальних методів чи jpql (java persistence query language).

База даних системи побудована на MySQL, для підключення до якої використовується бібліотека mysql-connector-j. Вона гарантує стабільний і швидкий доступ до даних, таких як інформація про користувачів, OTP-коди чи стан черг.

Обробка HTTP-запитів реалізована через Spring Web, підключеного за допомогою залежності spring-boot-starter-web. Цей сервіс відповідає за:

– розробку restful api, що дозволяє клієнтам взаємодіяти із сервером, відправляючи запити та отримуючи відповіді у форматі json або xml;

– обробку get, post, put, delete запитів для управління даними в системі;

– реалізацію маршрутизації, яка забезпечує правильне спрямування запитів до відповідних контролерів.

Цей сервіс також є основою для інтеграції клієнтської частини (користувацький інтерфейс) із сервером, забезпечуючи зв’язок між користувачами та логікою системи.

Захист даних і доступу до ресурсів забезпечується через Spring Security, інтегрований у проєкт за допомогою залежності spring-boot-starter-security. Основні можливості сервісу включають:

– аутентифікацію користувачів за допомогою різних механізмів, таких як логін і пароль;

– авторизацію доступу до ресурсів на основі ролей і прав користувачів;

– захист від поширених атак, зокрема csrf (cross-site request forgery) та xss (cross-site scripting).

Сервіс безпеки інтегрується з іншими компонентами системи, зокрема з доступом до даних та веб-сервісами, забезпечуючи комплексний захист всієї системи.

Ці сервіси працюють у тісному взаємозв'язку, утворюючи єдину екосистему:

– веб-сервіс приймає http-запити та передає їх до контролерів;

– контролери взаємодіють із сервісом доступу до даних для отримання або запису інформації;

– отримані дані обробляються і передаються до сервісу рендерингу html для формування динамічного інтерфейсу;

– сервіс безпеки забезпечує контроль доступу до запитів і даних на кожному етапі.

Таким чином, ці сервіси утворюють міцну архітектурну основу, яка дозволяє системі бути ефективною, безпечною та легкою у підтримці.

8.3 Компоненти рівня представлення

Рівень представлення відповідає за взаємодію користувачів із системою. У реалізації програмних систем, що базуються на архітектурі MVC (Model-View-Controller)[11], цей рівень представлений контролерами, які виступають інтерфейсом між клієнтськими запитами та бізнес-логікою. Окрім серверної частини, яка реалізована через контролери на Java, важливим компонентом є клієнтська логіка, реалізована за допомогою JavaScript.

8.3.1 Контроллери

Контролери приймають HTTP-запити, виконують базову перевірку даних, викликають необхідні сервіси для обробки запитів та повертають результати у відповідному форматі (HTML, JSON тощо). У даному проєкті контролери також реалізують REST API для забезпечення зручної інтеграції клієнтських інтерфейсів і зовнішніх систем із бізнес-логікою.

Перелік контроллерів розробленої системи зображений на рисунку 8.4.

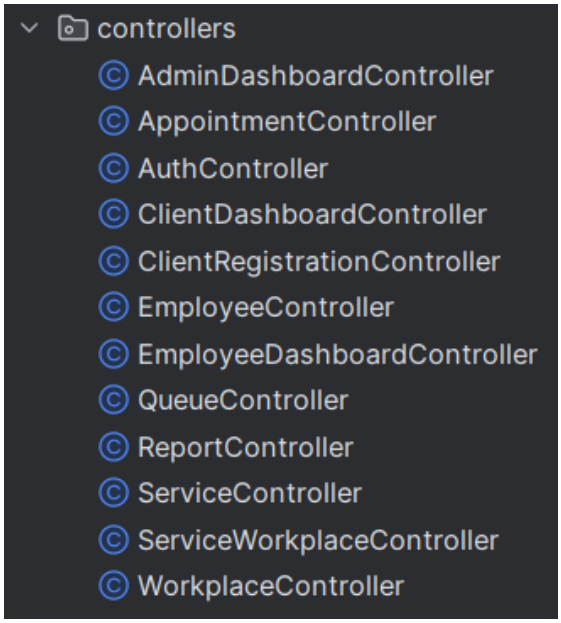

Рисунок 8.4 – Контроллери системи

Контролер AdminDashboardController відповідає за рендеринг панелі адміністратора, на якій адміністратор може побачити загальні дані про систему, такі як статистика, активні записи тощо. Основною його функцією є перевірка прав доступу користувача, щоб переконатися, що тільки авторизовані адміністративні користувачі можуть отримати доступ до цієї сторінки.

Функціонал даного контроллера наступний:

– перевірка ролі користувача (адміністратор);

– отримання сторінки панелі адміністратора;

– отримання сторінки керування послугами;

– отримання сторінки керування працівниками;

– отримання сторінки керування робочими місцями;

– отримання сторінки керування звітами;

– перенаправлення на сторінку входу, якщо користувач не авторизований.

Контролер AppointmentController забезпечує функціональність запису користувачів на прийом до фахівців. Користувачі можуть створювати нові записи, перевіряти доступні часи і скасувати вже заплановані зустрічі.

Функціонал даного контроллера наступний:

– перевірка ролі користувача (клієнт);

– отримання сторінки попереднього запису;

– створення, редагування та скасування записів;

Контролер AuthController відповідає за аутентифікацію користувачів. Він забезпечує можливість реєстрації, входу в систему та обробки сесій користувачів.

Функціонал даного контроллера наступний:

– створення сессії для користувача;

– автентифікація відповідного користувача;

– відправлення та перевірка OTP коду для клієнта;

– повернення відповідних сторінок, як логін клієнта, працівника, сторінку верифікації за OTP кодом а також головну сторінку системи;

– вихід користувача з системи.

Контролер ClientDashboardController відповідає за рендеринг панелі клієнта. Після входу користувач може переглядати свої записи, скасувати або змінити час прийому, а також отримувати іншу персоналізовану інформацію.

Функціонал даного контроллера наступний:

– перевірка ролі користувача (клієнт);

– отримання сторінки клієнта;

– відображення персоналізованих даних для клієнта (наприклад, майбутні записи);

– перенаправлення на сторінку входу, якщо користувач не авторизований.

Контролер ClientRegistrationController призначений для реєстрації нових клієнтів у системі. Користувач може створити обліковий запис, який дозволить йому записуватися на прийом та взаємодіяти з іншими частинами системи.

Функціонал даного контроллера наступний:

– отримання форми реєстрації клієнта;

– реєстрація нових клієнтів;

– перенаправлення на сторінку входу після успішної реєстрації.

Контролер EmployeeController відповідає за управління профілями працівників. Адміністратори можуть створювати, редагувати чи видаляти працівників системи, а також призначати їх на певні робочі місця.

Функціонал даного контроллера наступний:

– отримання списку всіх працівників;

– створення, редагування та видалення профілів працівників;

– призначення працівників на робочі місця;

– взаємодія з сервісами для обробки даних про працівників.

Контролер EmployeeDashboardController забезпечує доступ працівників до персоналізованої панелі, де вони можуть переглядати свої заплановані прийоми, обробляти чергу клієнтів та виконувати інші функції, пов'язані з їхньою роботою.

Функціонал даного контроллера наступний:

– перевірка ролі користувача (працівник);

– відображення списку поточних записів та черги;

– перенаправлення на сторінку входу, якщо користувач не авторизований.

Контролер QueueController керує чергами клієнтів. Користувачі можуть записуватися в чергу, а працівники — викликати клієнтів по черзі, змінювати чергу чи завершувати сеанси.

Функціонал даного контроллера наступний:

– створення, оновлення та видалення талону черги;

– перегляд поточних клієнтів у черзі;

– виклик наступного клієнта з черги;

– перегляд поточного клієнта, викликаного з черги;

– переміщення поточного клієнта між робочими місцями;

– завершити сессію для поточного клієнта.

Контролер ReportController забезпечує генерацію звітів на основі даних, що зібрані в системі. Адміністратори можуть генерувати звіти по різних параметрах, таких як кількість записів, середній час очікування, та інші метрики.

Функціонал даного контроллера наступний:

– генерація звітів на основі вказаних параметрів (дати, часу);

– підготовка csv-файлів для завантаження;

– видалення звітів після завантаження.

Контролер ServiceController відповідає за управління послугами, які доступні для запису. Адміністратори можуть додавати, оновлювати чи видаляти послуги, а також переглядати всі доступні послуги.

Функціонал даного контроллера наступний:

– отримання списку всіх послуг;

– додавання нових послуг;

– оновлення існуючих послуг;

– видалення послуг.

Контролер ServiceWorkplaceController відповідає за управління зв'язками між послугами та робочими місцями. Основна мета цього контролера — забезпечити інтерактивність між послугами та робочими місцями, включаючи можливість отримання списків, прив'язки та відв'язки.

Функціонал даного контролера наступний:

– прив'язка послуг до робочих місць;

– відв'язка послуг від робочих місць;

– отримання доступних послуг для конкретного робочого місця;

– отримання доступних робочих місць для конкретної послуги.

Контролер WorkplaceController керує робочими місцями, на яких виконуються послуги. Адміністратори можуть додавати нові робочі місця, оновлювати їх характеристики та видаляти їх із системи.

Функціонал даного контроллера наступний:

– отримання списку всіх робочих місць.

– додавання нових робочих місць.

– оновлення та видалення робочих місць.

8.3.2 Клієнтська логіка (JavaScript)

JavaScript код на стороні клієнта бере на себе завдання динамічного оновлення інтерфейсу, обробки подій користувача та виконання асинхронних запитів до серверної частини. Це дозволяє системі бути більш інтерактивною та зручною для користувача. Завдяки використанню JavaScript, користувачі можуть отримувати оновлення даних без необхідності перезавантаження сторінки, а також виконувати операції в реальному часі, такі як взаємодія з чергами, записами на прийом або управлінням робочими місцями[19].

Перелік JavaScript файлів розробленої системи зображений на рисунку 8.5.

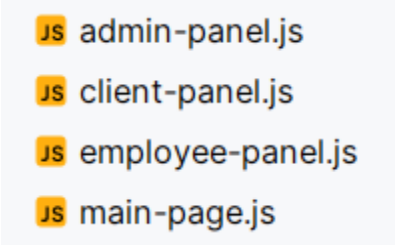

Рисунок 8.5 – Список JavaScript файлів системи

Admin-panel.js забезпечує керування глобальними параметрами системи, які доступні адміністратору, а саме:

– управління підменю та модальними вікнами за допомогою функцій openSubMenu(menuId), openModal(modalId) і closeModal(modalId);

– отримання даних (робочих місць, працівників, послуг) із сервера за допомогою функції fetchData(url, successCallback);

– робота з послугами (завантаження списку послуг з серверу та заповнення списку у відповідному select елементі, додавання послуг через POST-запит, оновлення послуг через PUT-запит, видалення послуг через DELETE-запит);

– робота з працівниками (завантаження списку працівників з серверу та заповнення списку у відповідному select елементі, додавання працівників через POST-запит, оновлення працівників через PUT-запит, видалення працівників через DELETE-запит);

– робота з робочими місцями (завантаження списку робочих місць з серверу та заповнення списку у відповідному select елементі, додавання робочих місць через POST-запит, оновлення робочих місць через PUT-запит, видалення робочих місць через DELETE-запит);

– робота зі зв’язками між послугами та робочими місцями, а саме додавання зв’язків між послугами та робочими місцями в систему, або їх редагування чи видалення.

– генерація та керування звітами (формування звіту через POST-запит, завантаження та видалення звіту через DELETE-запит).

Client-panel.js забезпечує керування параметрами системи, які доступні для клієнта, а саме:

– управління модальними вікнами за допомогою функцій openModal(modalId) і closeModal(modalId);

– отримання даних (послуг і створених попередніх записів) із сервера за допомогою функції fetchData(url, successCallback);

– робота з попередніми записами (бронювання через POST-запит, перезапис через PUT-запит, видалення через DELETE-запит);

– фільтрація вибору дати та часу для попереднього запису (за допомогою функції updateAvailableTimes).

Employee-dashboard.js відповідає за роботу з чергою клієнтів і обробку форм на сторінці, а саме:

– управління модальними вікнами за допомогою функцій openModal(modalId) і closeModal(modalId);

– завантаження даних на сторінку (за допомогою події DOMContentLoaded ініціалізує форми та основні блоки сторінки, Запускає періодичне оновлення даних (черга, клієнти) кожні 5 секунд і викликає функції для заповнення випадаючих списків із послугами та робочими місцями);

– отримання даних (ID поточного робочого місця, дані черги для поточного робочого місця, поточну чергу клієнтів та заповнює відповідні випадаючі списки, дані про поточного клієнта для обслуговування) із сервера за допомогою функцій fetchWorkplaceId(), populateDropdowns(), loadQueue(), populateCurrentQueue(), populateCurrentClient();

– отримання повідомлень за допомогою функції showNotification(message);

– робота з талонами (додавання через POST-запит, редагування через PUT-запит, видалення через DELETE-запит);

– створення обробнику для кнопки "Виклик наступного клієнта" (надсилає запит на сервер для виклику наступного клієнта, після чого оновлює відображення поточного клієнта та черги після виклику);

– створення обробнику для кнопки "Завершити сеанс" (завершує обслуговування поточного клієнта на робочому місці, після чого оновлює дані про чергу та клієнтів).

Main-page.js забезпечує динамічне оновлення черги клієнтів у режимі реального часу за допомогою технології Server-Sent Events (SSE). Він також відповідає за обробку звукових сповіщень для клієнтів. Даний JavaScript файл має наступні функції:

– оновлює відображення черги в елементі з ID in-progress-queue за допомогою функції updateQueueTable(queue);

– надає доступ до інших панелей залежно від ролі користувача;

– відображає повідомлення для клієнта та програє звукове сповіщення за допомогою функції playSoundAlert(client).

Також у всіх javaScript файлах наявна функція sanitizeHTML(text) для екранування даних, що захищає від XSS-атак.

Висновки до розділу 8

У цьому розділі було розглянуто основні компоненти архітектури програмного забезпечення, які формують ядро бізнес-логіки системи. Кожен із цих рівнів виконує важливу роль в забезпеченні функціональності, зручності взаємодії з користувачем та ефективності обробки даних.

Розробка та інтеграція компонентів рівня доступу до даних, бізнес-логіки та представлення дозволяє створити зручну та ефективну систему. Кожен із рівнів виконує свою унікальну роль: рівень доступу до даних відповідає за роботу з базою даних, рівень бізнес-логіки — за обробку і реалізацію бізнес-правил, а рівень представлення надає зручний інтерфейс для взаємодії з користувачем. Спільно вони формують систему, яка забезпечує ефективне управління записами, чергами та іншими процесами, гарантуючи зручність, надійність та масштабованість.

9 РОЗРОБЛЕННЯ ІНТЕРФЕЙСА КОРИСТУВАЧА

Розроблення інтерфейсу користувача для системи управління чергою було виконано з використанням HTML[21], CSS[22], JavaScript[20] та серверних шаблонів. Основна увага в ході розроблення інтерфесу користувача була присвячена простоті використання, інтуїтивності та адаптивності. Описані нижче сторінки охоплюють функціонал як для клієнтів, так і для працівників, а діаграми діяльності клієнта, працівника і адміністратора зображені на діаграмах Е, Ж та И відповідно.

9.1 Головна сторінка

Головна сторінка системи була спроектована так, щоб слугувати центральним елементом для всіх користувачів. На цій сторінці представлено інтерфейс, що дозволяє переглядати актуальний стан черги в реальному часі. Таблиця з чергою автоматично оновлюється за допомогою JavaScript і відображає список поточних клієнтів та відповідних робочих місць.

Дизайн сторінки є простим, зі світлим фоном та контрастними шрифтами для легкості читання. У верхній частині сторінки розміщено навігаційну панель із кнопками для вибору ролі: клієнта або працівника. Ці кнопки ведуть до відповідних сторінок входу.

Вигляд головної сторінки зображено на рисунку 9.1.

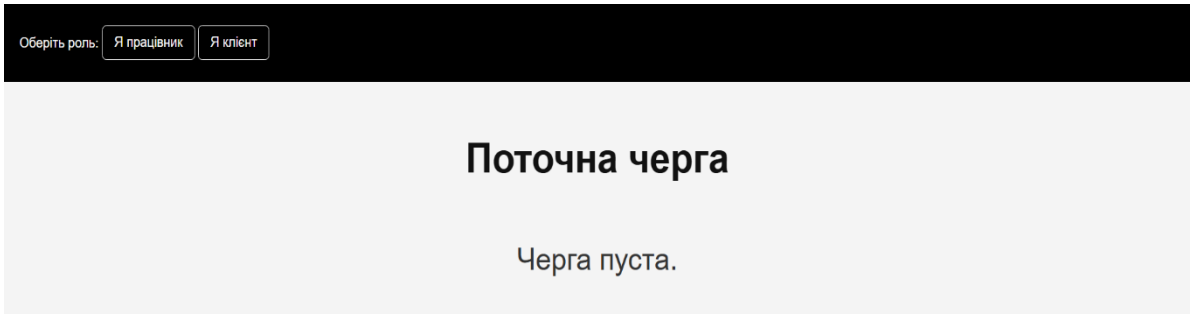

Рисунок 9.1 – Головна сторінка системи

9.2 Сторінка логіну клієнта

Сторінка входу для клієнта забезпечує можливість введення номера телефону, що є ключовим елементом для ідентифікації користувача в системі. Інтерфейс цієї сторінки є мінімалістичним, з фокусом на полі введення номера та кнопці для надсилання запиту на отримання OTP (одноразового пароля).

Додано валідацію на клієнтській стороні для забезпечення коректного формату введеного номера телефону відповідно до міжнародних стандартів. Також передбачено кнопку для переходу до сторінки реєстрації клієнта, що робить систему доступною для нових користувачів.

Вигляд сторінки логіну клієнта зображено на рисунку 9.2.

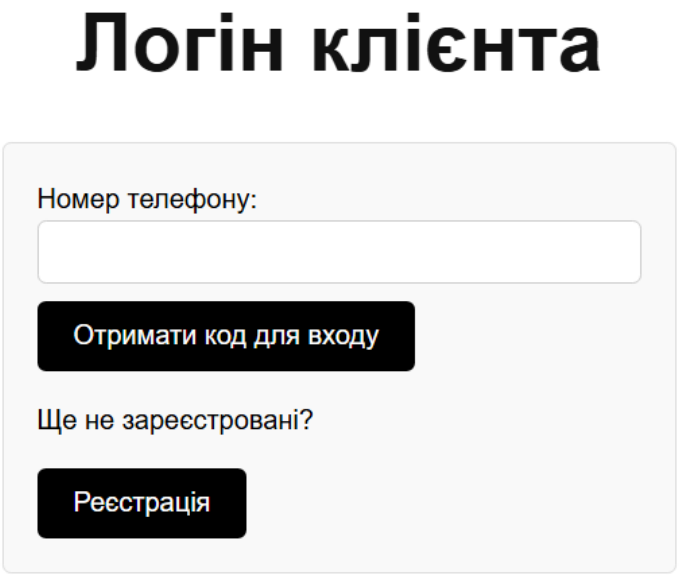

Рисунок 9.2 – Сторінка логіну клієнта

9.3 Сторінка логіну працівника

Сторінка входу для працівника орієнтована на забезпечення безпечного доступу до системи для персоналу. Її структура містить поля для введення імені користувача та пароля, які перевіряються на клієнтській стороні за допомогою

регулярних виразів. Цей підхід гарантує, що введені дані відповідають встановленим стандартам.

Дизайн сторінки є лаконічним, але достатньо функціональним для швидкого входу працівників у систему. Дані передаються на сервер через метод POST, що забезпечує додатковий рівень безпеки. Вигляд сторінки логіну працівника зображено на рисунку 9.3.

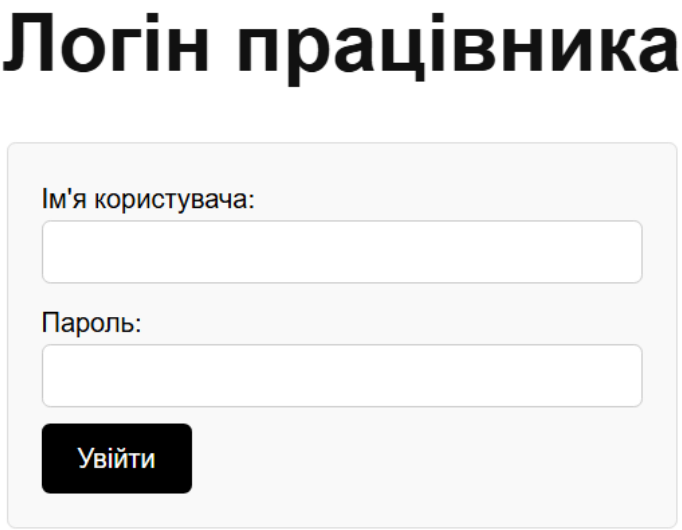

Рисунок 9.3 – Сторінка логіну працівника

9.4 Сторінка OTP верифікації

Сторінка підтвердження OTP створена спеціально для клієнтів, які вже отримали одноразовий пароль після введення номера телефону. Її головною метою є перевірка цього коду для завершення процесу ідентифікації.

Валідація коду здійснюється на клієнтській стороні, що дозволяє уникнути помилок введення ще до надсилання даних на сервер. Інтеграція з бекендом відбувається через серверні шаблони, які динамічно генерують сторінку, підставляючи номер телефону для подальшого зв’язку з попередніми діями користувача.

Дизайн зберігає мінімалістичний підхід: одне поле для введення коду OTP, приховане поле для передачі номера телефону та кнопка для підтвердження. Вигляд сторінки верифікації OTP зображена на рисунку 9.4

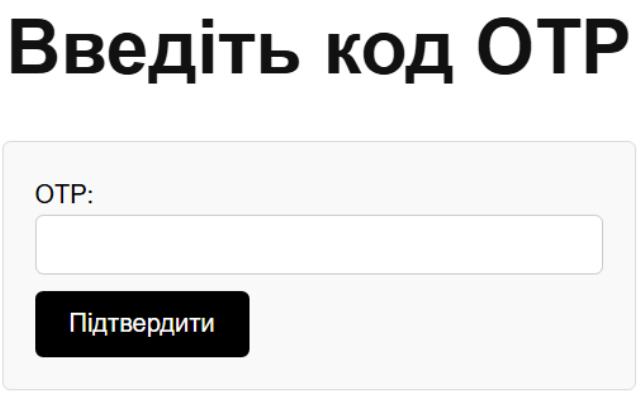

Рисунок 9.4 – Сторінка підтвердження OTP

9.5 Сторінка реєстрації клієнта

Для забезпечення доступу до послуг системи клієнти мають можливість зареєструватися за допомогою веб-форм. У HTML-документі форма розроблена з урахуванням основних потреб користувача: введення імені та номера телефону.

Для поліпшення якості даних передбачено використання атрибутів валідації (required та pattern), що гарантують відповідність введених даних встановленим форматам.

Вигляд сторінки реєстрації клієнта зображена на рисунку 9.5

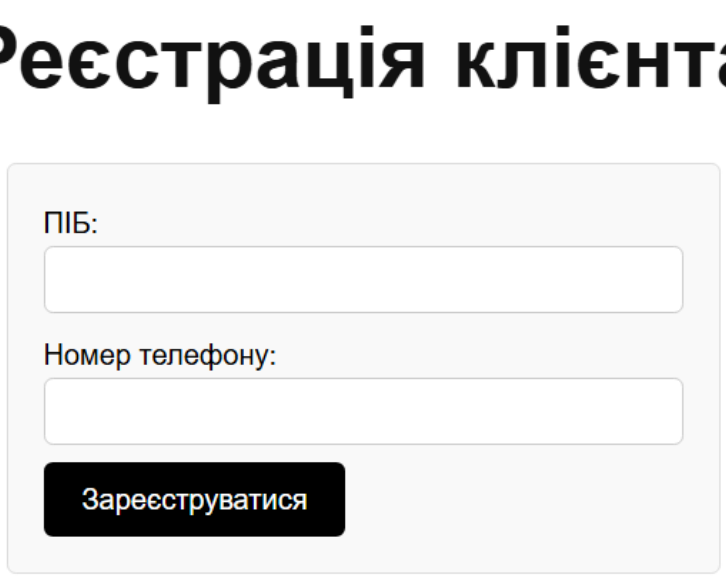

Рисунок 9.5 – Сторінка форми реєстрації клієнта

9.5 Особистий кабінет клієнта

Особистий кабінет надає клієнтам можливість переглядати створені записи та бронювати нові послуги. Модальне вікно для створення запису містить елементи для вибору послуги, дати й часу.

Інтерфейс адаптований для різних пристроїв, що робить його зручним для мобільних користувачів. Динамічне завантаження списків послуг і доступних часових слотів реалізовано через скрипти, які взаємодіють із сервером.

Вигляд описаного функціоналу та особистого кабінету зображений на рисунках 9.6 – 9.9.

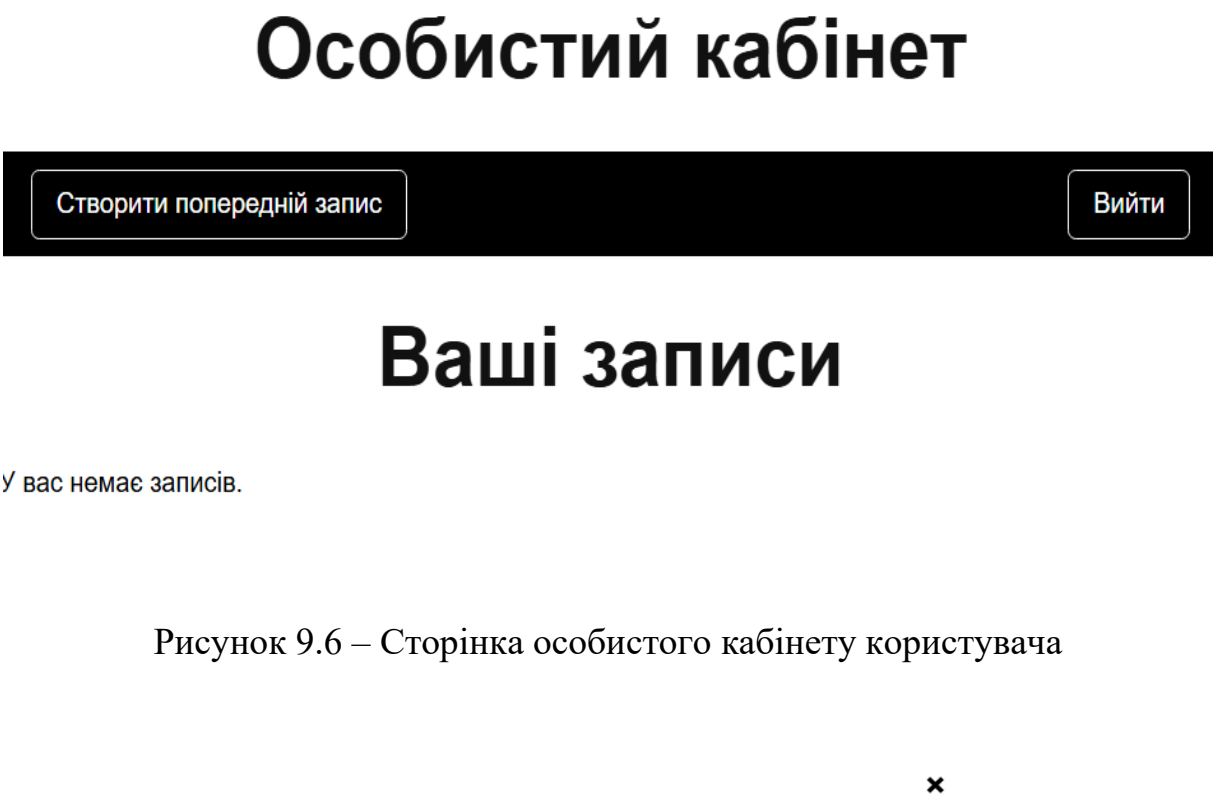

Рисунок 9.6 – Сторінка особистого кабінету користувача

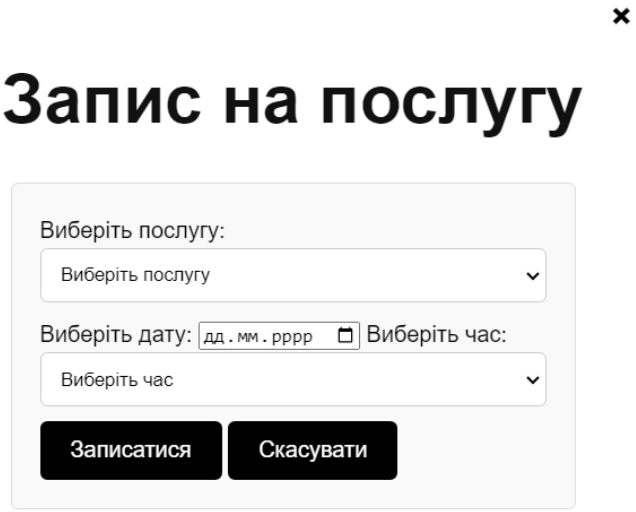

Рисунок 9.7– Сторінка запису на послугу

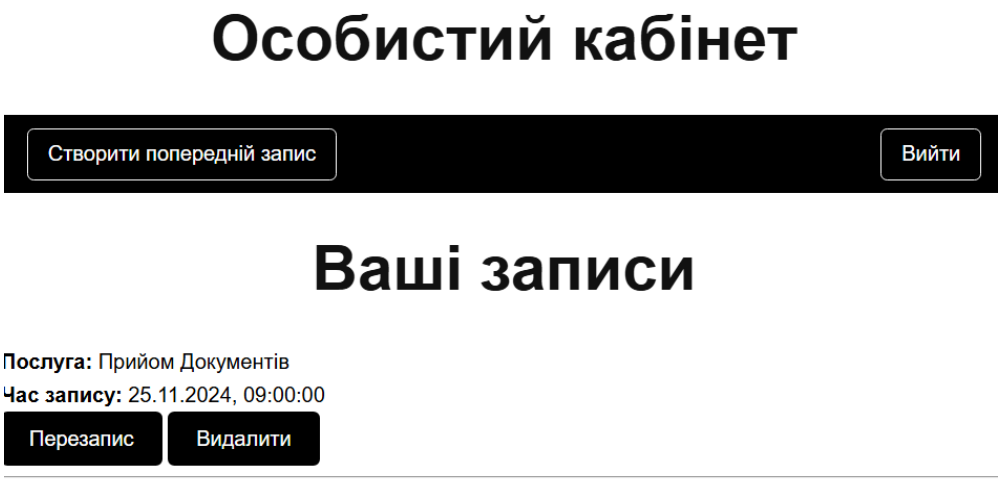

Рисунок 9.8 – Особистий кабінет користувача після створення запису

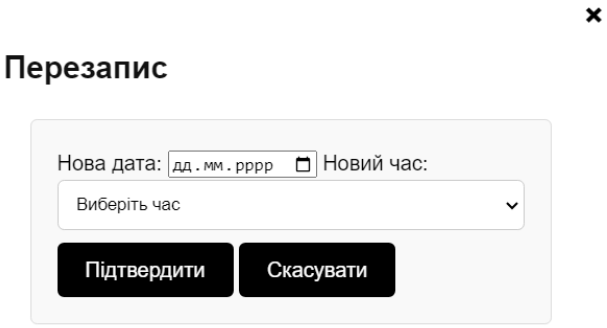

Рисунок 9.9 – Сторінка перезапису на попередню послугу

9.6 Адміністративна панель

Адміністративна панель є центральною точкою управління для адміністраторів системи. У панелі реалізовано декілька функціональних підменю для керування послугами, робочими місцями, працівниками, зв’язками послуг та робочих місць та генерації звітів, а також реалізована можливість вийти з обілкового запису.

Кожна функція представлена у вигляді інтерактивного модального вікна, яке викликається кнопками з відповідних підменю. Динамічність панелі забезпечується через JavaScript, що дозволяє завантажувати та оновлювати дані в

реальному часі без перезавантаження сторінки. Вигляд сторінки та модальних вікон зображено на рисунках 9.10-9.27.

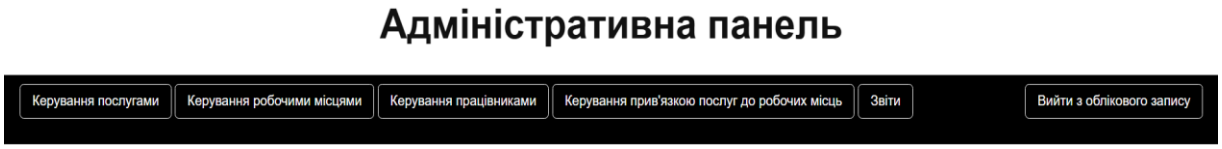

Рисунок 9.10 – Сторінка панелі адміністратора

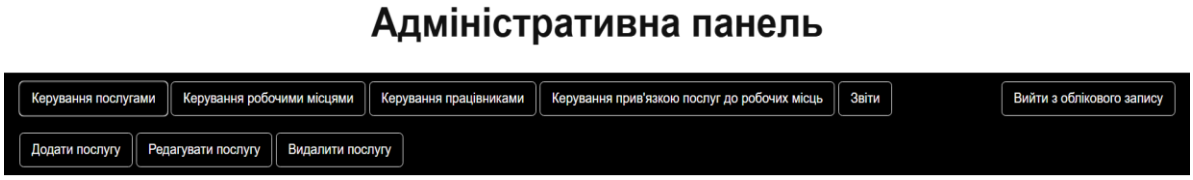

Рисунок 9.11 – Сторінка з відкритим підменю керування послугами

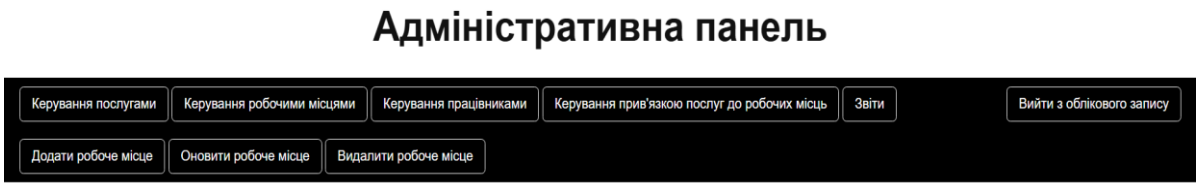

Рисунок 9.12 – Сторінка з відкритим підменю керування робочими місцями

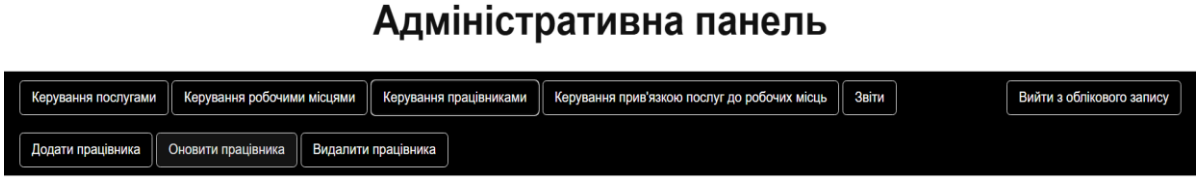

Рисунок 9.13 – Сторінка з відкритим підменю керування працівниками

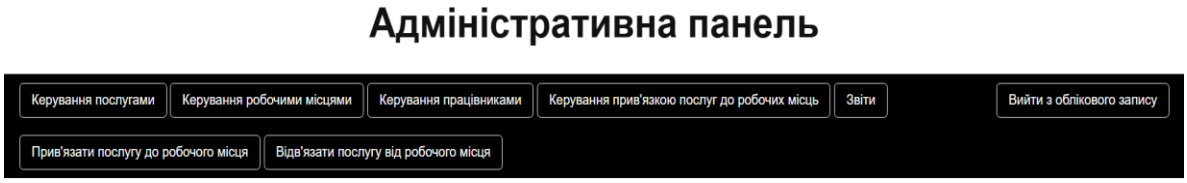

Рисунок 9.14 – Сторінка з відкритим підменю керування прив’язкою послуг до робочих місць

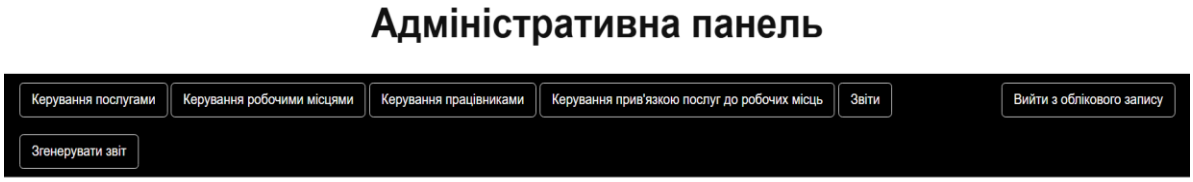

Рисунок 9.15 – Сторінка з відкритим підменю звіти

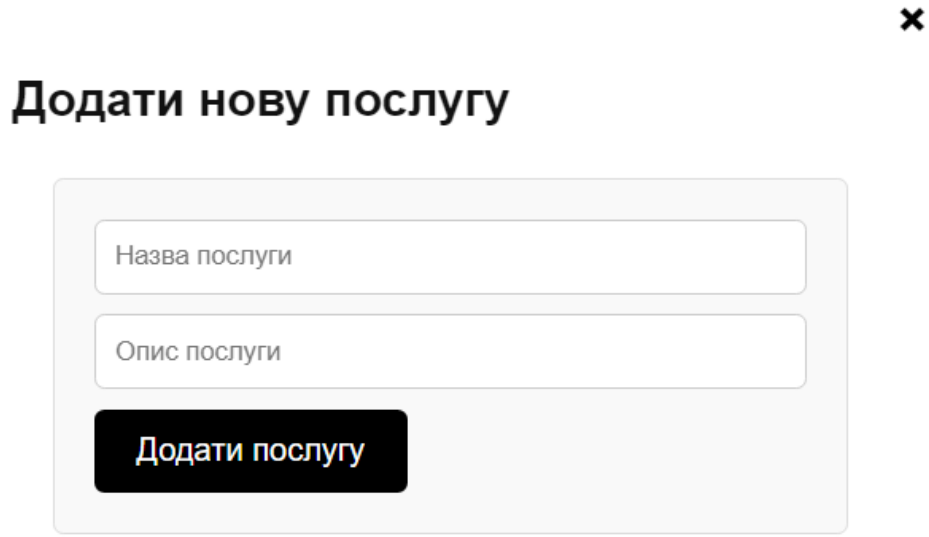

Рисунок 9.16 – Сторінка з модальним вікном додати нову послугу

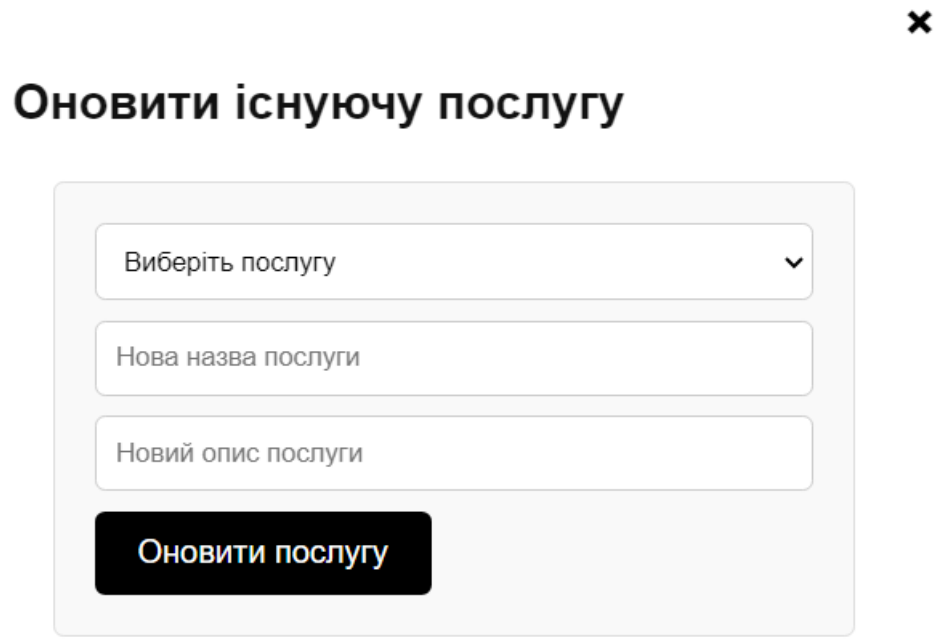

Рисунок 9.17 – Сторінка з модальним вікном оновити існуючу послугу

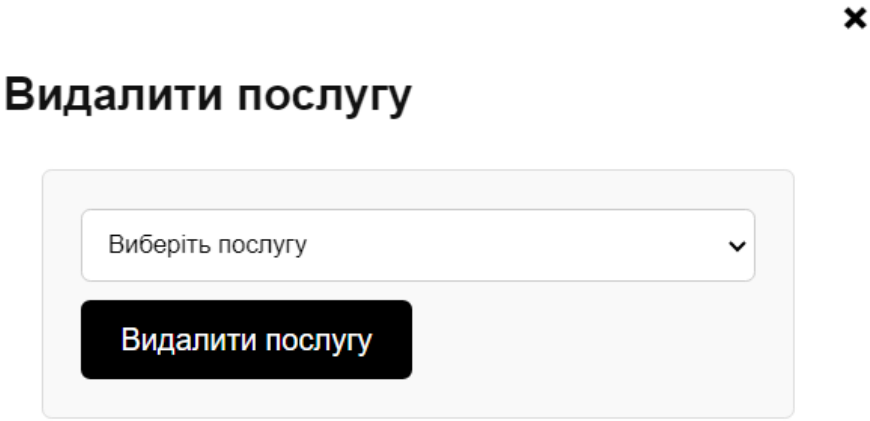

Рисунок 9.18 – Сторінка з модальним вікном видалити послугу

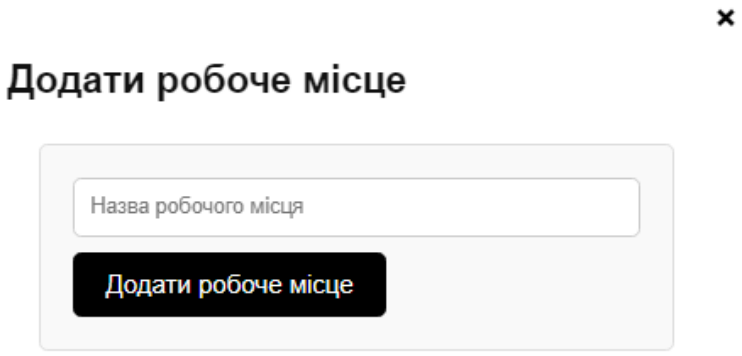

Рисунок 9.19 – Сторінка з модальним вікном додати робоче місце

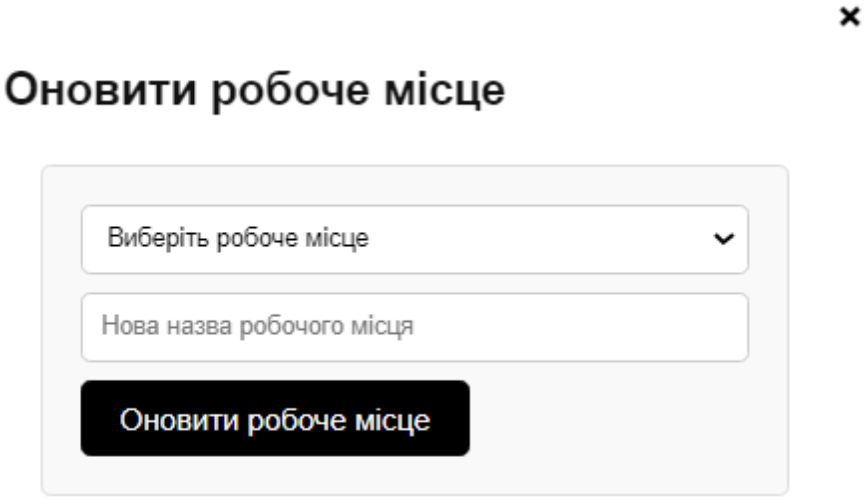

Рисунок 9.20 – Сторінка з модальним вікном оновити робоче місце

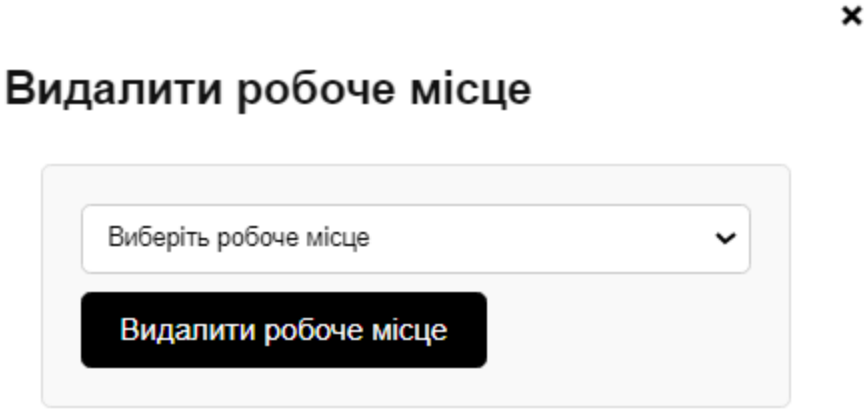

Рисунок 9.21 – Сторінка з модальним вікном видалити робоче місце

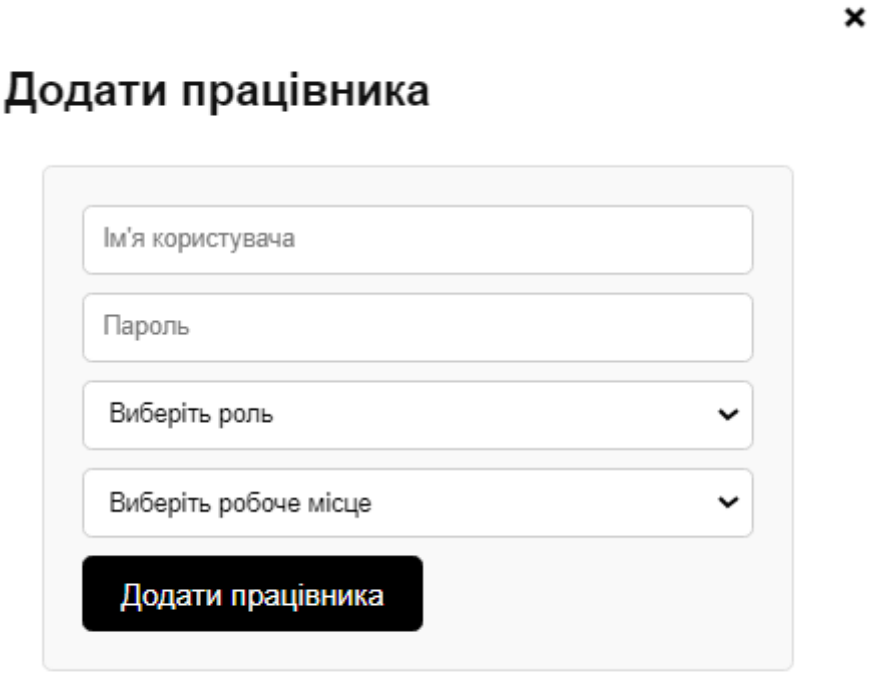

Рисунок 9.22 – Сторінка з модальним вікном додати працівника

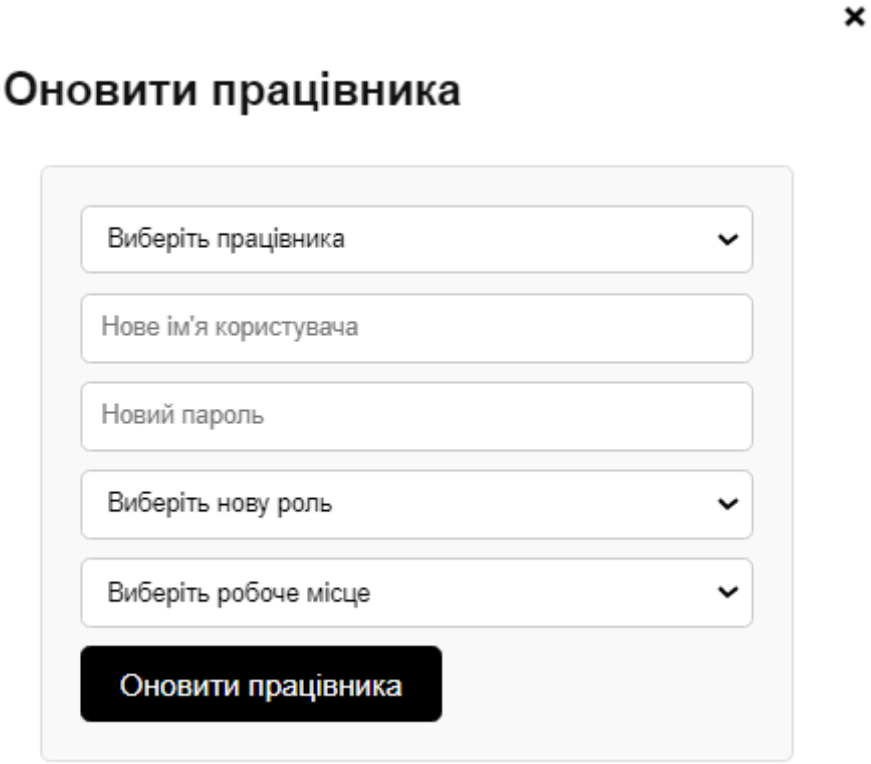

Рисунок 9.23 – Сторінка з модальним вікном оновити працівника

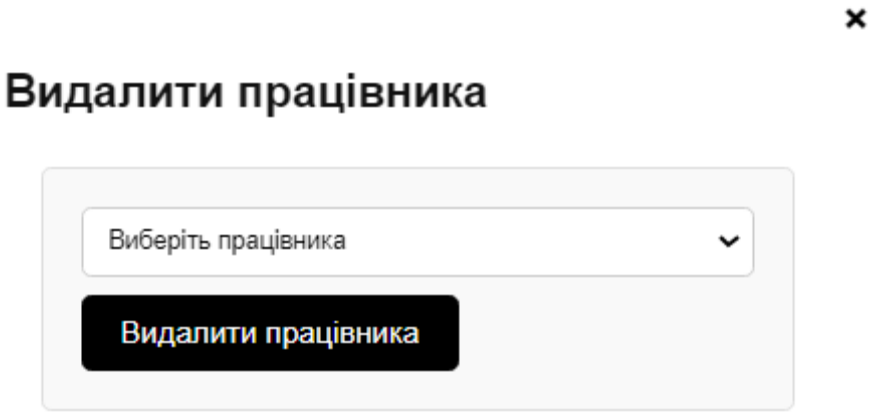

Рисунок 9.24 – Сторінка з модальним вікном видалити працівника

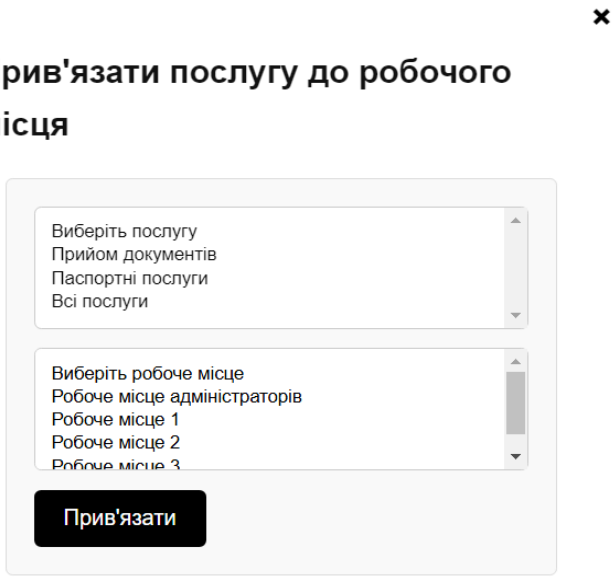

Рисунок 9.25 – Сторінка з модальним вікном прив’язати послугу до робочого місця

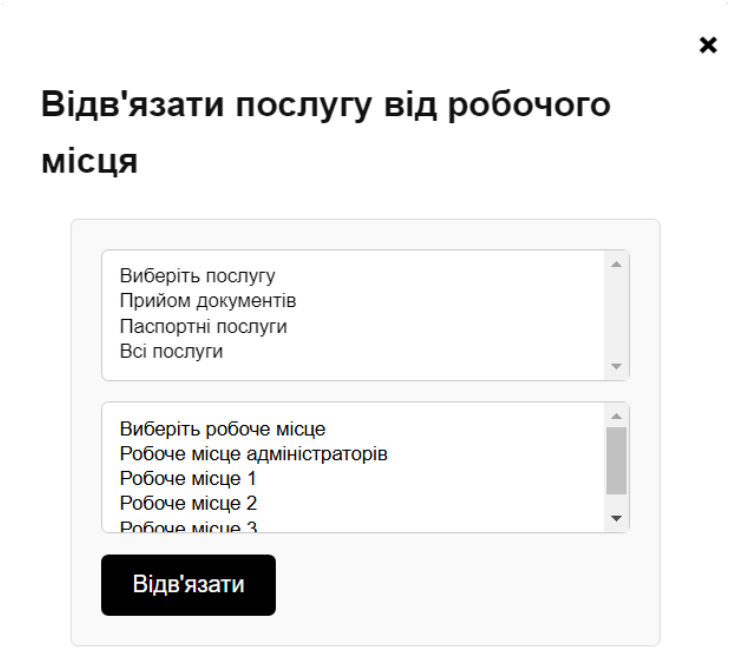

Рисунок 9.26 – Сторінка з модальним вікном відв’язати послугу від робочого місця

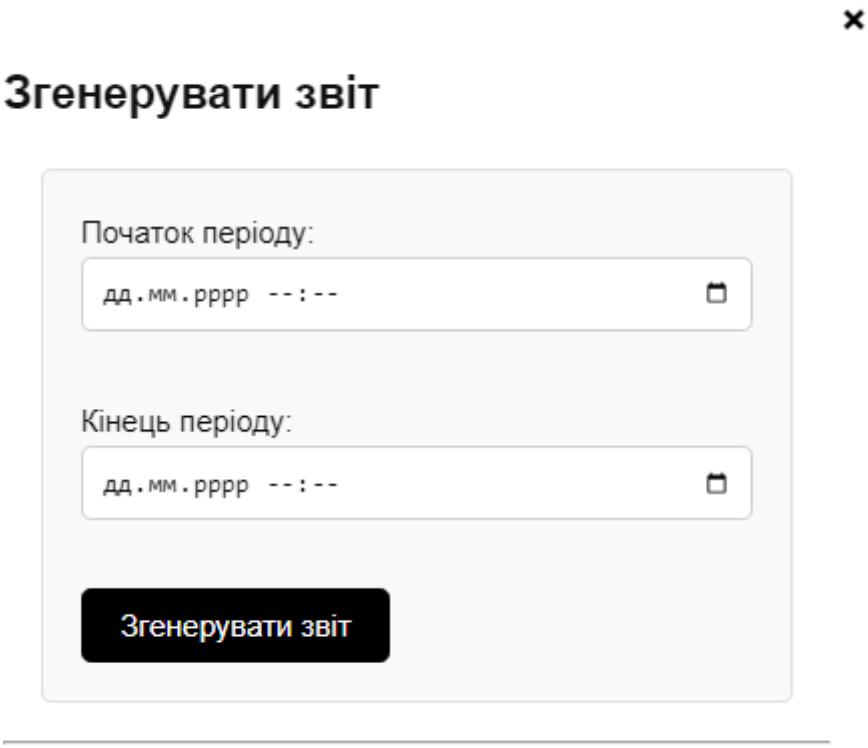

Рисунок 9.27 – Сторінка з модальним вікном згенерувати звіт

9.7 Дошка працівника

Інтерфейс дошки дозволяє створювати, оновлювати або видаляти талони клієнтів, а також викликати наступного клієнта чи завершувати обслуговування. Усі дії супроводжуються візуальним підтвердженням через модальні вікна, які відкриваються на вимогу. Додатково передбачена можливість передачі клієнта між робочими місцями, що сприяє оптимізації роботи.

Вигляд функціоналу та сторінки дошки працівника зображено на рисунках 9.28-9.32.

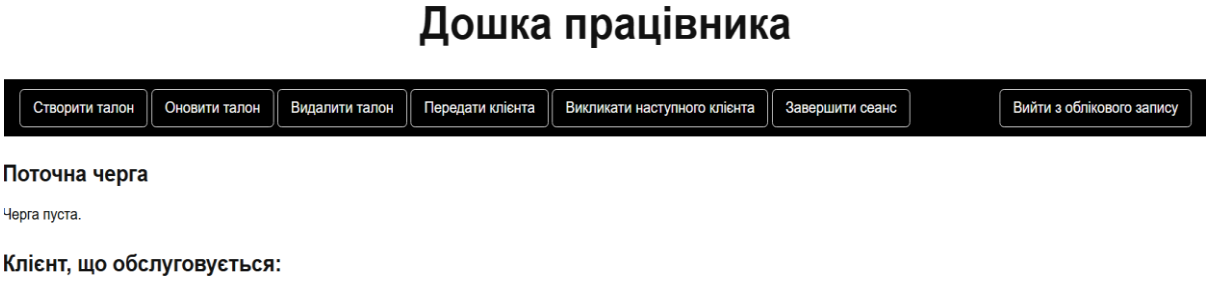

Рисунок 9.28 – Сторінка панелі працівника

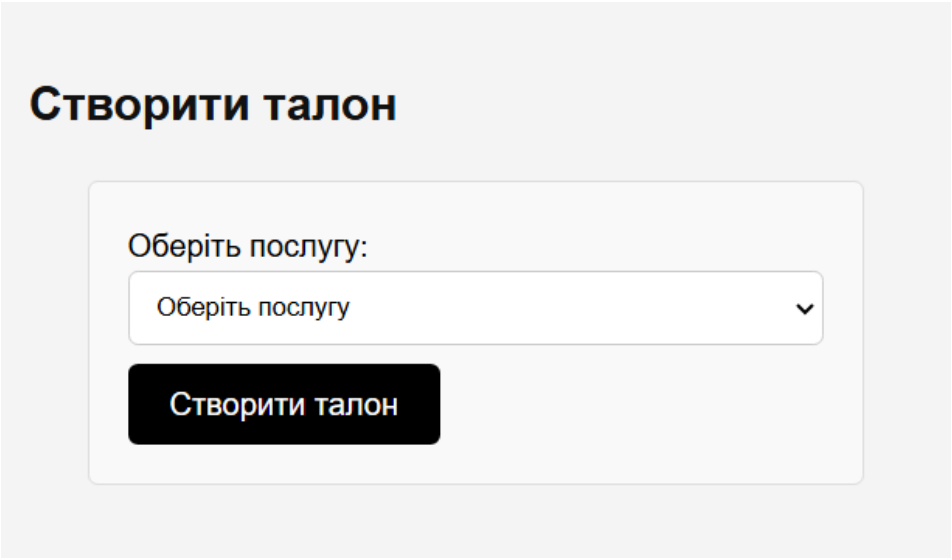

Рисунок 9.29 – Сторінка з модальним вікном створення талону

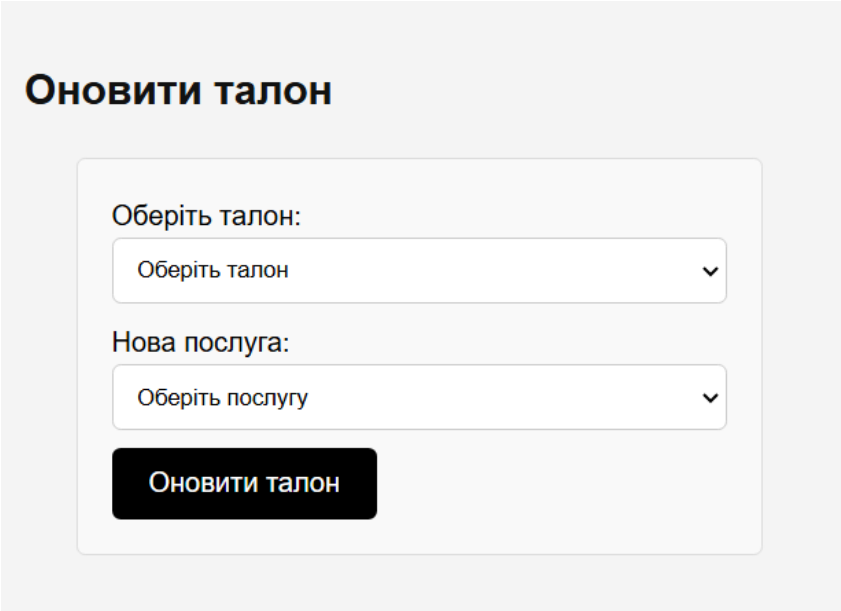

Рисунок 9.30 – Сторінка з модальним вікном оновлення талону

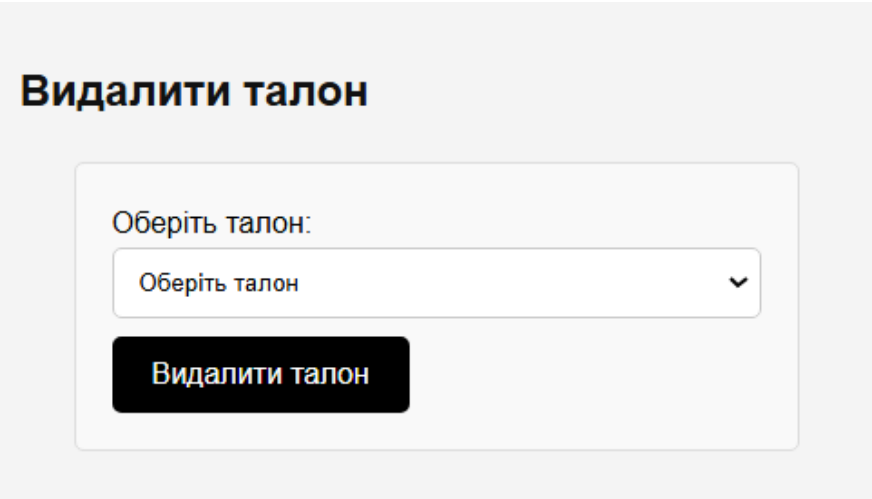

Рисунок 9.31 – Сторінка з модальним вікном видалення талону

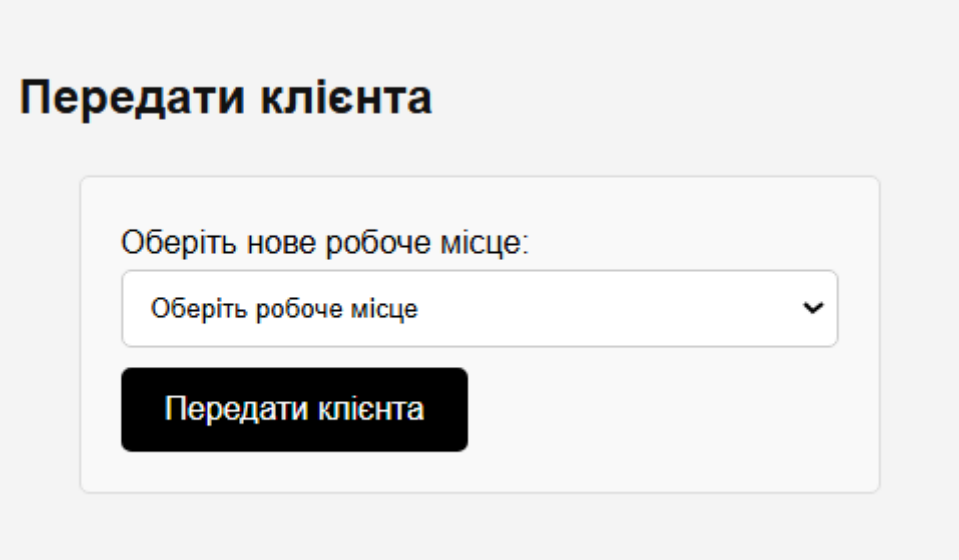

Рисунок 9.32 – Сторінка з модальним вікном передачі талона іншому робочому місцю

Висновок до розділу 9

Було розроблено інтерфейс користувача інформаційної системи управління чергою. Ключова мета була створити інтуїтивний, зручний та адаптивний дизайн який відповідає сучасним вимогам веб-додатків. Розроблений інтерфейс охопив усі потреби різних категорій користувачів таких як клієнти, працівники системи та адміністратори системи. Розроблений інтерфейс є сучасним, ергономічним та орієнтованим на потреби користувачів, що сприяє підвищенню ефективності роботи системи управління чергою.

10 ТЕСТУВАННЯ СИСТЕМИ

Критично важиливим етапом у процесі розробки програмного забезпечення є тестування системи. Даний етап дозволяє переконатись у відповідності функціоналу системи згідно поставлених вимог, забезпечити надійність роботи системи та виявити помилки.

10.1 Перевірка функціоналу OTP коду

Для перевірки працездатності однієї з головних функцій системи – перевірка OTP-коду – необхідне підключення стороннього SMS API для відправлення коду на фізичний телефон. Однак подібного API немає у відкритому доступі, чим блокується дана можливість тестування.

Однак було сторено штучний функціонал подібного API для відправки коду на телефон. Послідовність даного функціоналу зображений на додатку К.

Після того як клієнт відправляє запит на сервер про генерацію коду, сервер генерує 6-значний OTP, шифрує OTP перед збереженням в базу даних і логуванням, відправляє на базу даних і завдяки логуванню ми імітуємо відправку на телефон. Після вводу коду на сайті інтерфейс користувача повертає дані на сервер, після чого він їх шифрує та порівнює з зашифрованим кодом в базі даних. Якщо код збігається сервер повертає інтерфейсу користувача сторінку особистого кабінету клієнта, що означає що авторизація пройшла успішно.

10.2 Тест-кейси

Для перевірки відповідності функціоналу системи специфікаціям було розроблено тест-кейси для кожного модуля. У таблицях нижче наведено приклади тест-кейсів із описом їхньої реалізації та очікуваними результатами.

Таблиця 10.1 – Модуль логіну працівника

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-001	Вхід працівника з валідними даними	Обліковий запис існує	1. Відкрити сторінку логіну 2. Ввести валідні логін і пароль 3. Натиснути «Увійти»	Успішний вхід. Перехід на панель працівника
ТС-002	Вхід працівника з невалідними даними	Обліковий запис існує	1. Відкрити сторінку логіну. 2. Ввести некоректний логін чи пароль. 3. Натиснути «Увійти»	Відображення помилки «Невірний логін або пароль»
ТС-003	Спроба входу працівника з незаповненими полями	-	1. Відкрити сторінку логіну 2. Натиснути «Увійти», залишивши поля порожніми	Відображення повідомлення «Поле є обов’язковим»

Таблиця 10.2 – Модуль логіну клієнта

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
TC-004	Вхід клієнта з валідним номером телефону	Клієнт зареєстрований	1. Відкрити сторінку логіну 2. Ввести валідний номер телефону 3. Натиснути «Отримати OTP»	Код OTP надіслано на телефон
TC-005	Вхід клієнта з невалідним номером телефону	-	1. Відкрити сторінку логіну 2. Ввести некоректний номер 3. Натиснути «Отримати OTP»	Відображення помилки «Некоректний формат»
TC-006	Вхід із незаповненим полем номера телефону	-	1. Відкрити сторінку логіну 2. Натиснути «Отримати OTP», залишивши	Відображення повідомлення «Поле є обов’язковим»

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
			поле порожнім	

Таблиця 10.3 – Модуль реєстрації клієнта

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-007	Реєстрація нового клієнта з валідними даними	Клієнт не зареєстрований	1. Відкрити сторінку реєстрації 2. Ввести ім’я та валідний номер телефону 3. Натиснути «Реєстрація»	Клієнт зареєстрований. Перехід на сторінку OTP
ТС-008	Реєстрація клієнта з уже існуючим номером телефону	Клієнт зареєстрований	1. Відкрити сторінку реєстрації 2. Ввести існуючий номер телефону 3. Натиснути «Реєстрація»	Відображення помилки «Користувач вже існує»

Таблиця 10.4 – Модуль верифікації OTP-коду

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
TC-009	Верифікація валідного OTP	Клієнт отримав код OTP	1. Відкрити сторінку OTP 2. Ввести коректний код 3. Натиснути «Підтвердити»	Успішна верифікація. Перехід до особистого кабінету
TC-010	Верифікація некоректного OTP	Клієнт отримав код OTP	1. Відкрити сторінку OTP 2. Ввести некоректний код OTP 3. Натиснути «Підтвердити»	Відображення помилки «Код Невірний»
TC-011	Спроба підтвердження з порожнім полем.	-	1. Відкрити сторінку OTP. 2. Натиснути «Підтвердити», не вводячи код.	Відображення повідомлення «Поле є обов’язковим».

Таблиця 10.5 – Модуль адміністративної панелі

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-012	Додавання нової послуги	-	1. Перейти до розділу «Послуги» 2. Натиснути "Додати" 3. Заповнити назву та опис 4. Натиснути «Зберегти»	Нова послуга додана
ТС-013	Редагування існуючої послуги	Послуга існує	1. Перейти до розділу «Послуги» 2. Обрати послугу 3. Натиснути «Редагувати» 4. Внести зміни 5. Зберегти	Послуга оновлена
ТС-014	Видалення послуги	Послуга існує	1. Перейти до розділу «Послуги» 2. Обрати послугу 3. Натиснути «Видалити»	Послуга видалена

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-015	Додавання нового працівника	-	1. Перейти до розділу «Працівники» 2. Натиснути "Додати" 3. Заповнити дані 4. Натиснути «Зберегти»	Працівник доданий
ТС-016	Редагування даних працівника	Працівник існує	1. Перейти до розділу «Працівники». 2. Обрати працівника 3. Натиснути «Редагувати» 4. Внести зміни 5. Зберегти	Дані працівника оновлені
ТС-017	Видалення працівника	Працівник існує	1. Перейти до розділу «Працівники» 2. Обрати працівника 3. Натиснути «Видалити»	Працівник видалений
ТС-018	Додавання нового	-	1. Перейти до розділу	Робоче місце додане

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
	робочого місця		«Робочі місця» Натиснути «Додати» 2. Заповнити дані 3. Натиснути «Зберегти»	
ТС-019	Редагування робочого місця	Робоче місце існує	1. Перейти до розділу «Робочі місця». 2. Обрати робоче місце 3. Натиснути «Редагувати» 4. Внести зміни 5. Зберегти	Робоче місце оновлено
ТС-020	Видалення робочого місця	Робоче місце існує	1. Перейти до розділу «Робочі місця» 2. Обрати робоче місце 3. Натиснути «Видалити»	Робоче місце видалено

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-021	Прив’язка послуги до робочого місця	Послуга і робоче місце існує	1. Перейти до розділу «прив'язка послуг до робочих місць» 2. Обрати послуги і робочі місця 3. Натиснути прив'язати	Послугу прив’язано до робочого місця
ТС-022	Відв’язка послуги від робочого місця	Послуга і робоче місце існує	1. Перейти до розділу «відв'язка послуг від робочих місць» 2. Обрати послуги і робочі місця 3. Натиснути відв'язати	Послугу відв’язано від робочого місця
ТС-023	Генерація звіту за обраний період	Дані для звіту існують	1. Перейти до розділу «Звіти». 2. Обрати період	Звіт створений. Звіт відображено на екрані або завантажено

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
			3. Натиснути «Згенерувати»	
ТС-024	Логаут адміністратора	Адміністратор увійшов	1. Натиснути «Вийти з облікового запису»	Сессія адміністратора завершена. Перехід на головну сторінку

Таблиця 10.6 – Модель панелі працівника

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-025	Створення талона клієнта	Клієнт у черзі.	1. Натиснути на кнопку «створити талон» 2. Обрати послугу та робоче місце 3. Натиснути «Створити талон».	Талон створено
ТС-026	Редагування талону клієнта	Талон перебуває в поточній черзі	1. Натиснути на кнопку «оновити талон»	Талон оновлено

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
			2. Обрати нові дані для талону 3. Натиснути «Оновити талон»	
ТС-027	Видалення талогу клієнта	Талон перебуває в поточній черзі	1. Натиснути на кнопку «видалити талон» 2. Обрати талон зі списку 3. Натиснути «Видалити талон»	Талон видалено
ТС-028	Викликати талон клієнта	Талон перебуває в поточній черзі	1. Натиснути на кнопку «Викликати наступного клієнта»	Перший талон в поточній черзі змінив статус з «очікує» на «обслуговується»
ТС-029	Передати клієнта іншому робочому місцю	Талон перебуває поточній черзі, талон має статус	1. Натиснути на кнопку «передати клієнта»	Талон поточного робочого місця передано

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
		«обслуговується»	2. Обрати робоче місце якому передається клієнт Натиснути «Передати»	іншому робочому місцю.
ТС-030	Завершити сеанс з поточним клієнтом	Талон перебуває поточній черзі, талон має статус «обслуговується»	1. Натиснути «завершити сеанс»	Талон змінив статус з «обслуговується» на «закінчений» та видаляється з черги
ТС-031	Вихід з облікового запису працівника	Працівник увійшов у систему	1. Натиснути «Вийти з обілкового запису»	Завершення сессії працівника Перехід на головну сторінку

Таблиця 10.7 – Модель особистого кабінету клієнта

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
ТС-032	Створення попереднього запису	Клієнт увійшов у систему.	1. Перейти до розділу «Записатись на послугу».	Запис створено.

ID	Опис	Передумови	Кроки виконання	Очікуваний результат
		Обрана час і дата доступні	Обрати послугу. 2. Обрати дату і час. 3. Натиснути на кнопку «Записатись»	
ТС-033	Оновити попередній запис на іншу дату і годину	Клієнт має існуючий запис	1. Обрати послугу. 2. Натиснути "Перезапис". 3. Вказати нову годину і дату. 4. Натиснути на кнопку «підтвердити.	Запис оновлено
ТС-034	Видалити попередній запис	Клієнт має існуючий запис	1. Обрати послугу 2. Натиснути «Видалити»	Запис видалено
ТС-035	Вихід з облікового запису клієнта	Клієнт увійшов у систему	1. Натиснути «Вийти з обілкового запису»	Завершення сессії клієнта. Перехід на головну сторінку

Для забезпечення повноти тестування була створена матриця трасування вимог, яка дозволяє зіставити тест-кейси з відповідними функціональними вимогами. Таблиця складається з ID вимоги, опису вимогу, пов’язаних модулей, ID тест-кейсу і результату тесту. Матриця трасування вимог зображена за допомогою таблиці 10.8.

Таблиця 10.8 – Матриця трасування вимог

ID Вимоги	Опис Вимоги	Пов’язані Модулі	ID Тест-Кейсу	Результат Тесту
REQ-001	Система має дозволяти працівникам входити в систему.	Модуль логіну працівника	TC-001, TC-002, TC-003	Pass
REQ-002	Система має дозволяти клієнтам входити в особистий кабінет.	Модуль логіну клієнта	TC-004, TC-005, TC-006	Pass
REQ-003	Клієнт має можливість зареєструватися в системі.	Модуль реєстрації клієнта	TC-007, TC-008, TC-009, TC-010, TC-011	Pass
REQ-004	Верифікація OTP-коду під час входу або реєстрації.	Модуль верифікації OTP-коду	TC-009, TC-010, TC-011	Pass
REQ-005	Адміністратор може керувати послугами.	Модуль адміністративної панелі	TC-012, TC-013, TC-014	Pass
REQ-006	Адміністратор може керувати працівниками.	Модуль адміністративної панелі	TC-015, TC-016, TC-017	Pass

ID Вимоги	Опис Вимоги	Пов’язані Модулі	ID Тест-Кейсу	Результат Тесту
REQ-007	Адміністратор може керувати робочими місцями.	Модуль адміністративної панелі	TC-018, TC-019, TC-020	Pass
REQ-008	Адміністратор може керувати прив’язками послуг і робочих місьц	Модуль адміністративної панелі	TC-021, TC-022	Pass
REQ-009	Адміністратор може генерувати звіти.	Модуль адміністративної панелі	TC-023	Pass
REQ-010	Користувач може вийти з облікового запису	Модуль адміністративної панелі, модуль панелі працівника модуль особистого кабінету користувача	TC-024, TC-031, TC-035	Pass
REQ-011	Працівник може керувати талонами черги.	Модуль панелі працівника	TC-025, TC-026, TC-027	Pass
REQ-012	Працівник може	Модуль панелі працівника	TC-028	Pass

ID Вимоги	Опис Вимоги	Пов’язані Модулі	ID Тест-Кейсу	Результат Тесту
	викликати талон з черги			
REQ-013	Працівник може передати клієнта іншому робочому місцю	Модуль панелі працівника	TC-029	Pass
REQ-014	Працівник може завершити сеанс з поточним клієнтом	Модуль панелі працівника	TC-030	Pass
REQ-015	Клієнт може створювати попередній запис.	Модуль особистого кабінету користувача	TC-032	Pass
REQ-016	Клієнт може редагувати або видаляти записи.	Модуль особистого кабінету користувача	TC-033, TC-034	Pass

Висновки до розділу 10

У даному розділі було описано різні тестування для перевірки коректності роботи системи згідно поставлених вимог.

Проведене модульне тестування підтвердило коректність роботи окремих частин системи. Для кожного модуля були розроблені детальні тест-кейси, що забезпечили всебічну перевірку функціональності, включаючи позитивні й негативні сценарії. Матриця трасування забезпечила повний зв'язок між вимогами, тест-кейсами та відповідними модулями системи. Це дозволило гарантувати, що кожна вимога була врахована і перевірена.

11 СТАРТАП ПРОЄКТ

11.1 Опис ідеї проєкту

Ідея даного проєкту полягає у реалізації інформаційної системи управління чергами державних установ. Вона дозволяє здійснювати управління чергами в режимі реального часу, бронювати послуги, отримувати сповіщення про статус черги та виконувати операції адміністрування. Короткий опис ідеї проєкту зображений в таблиці 11.1.

Таблиця 11.1 – Опис ідеї проєкту

Зміст ідеї	Напрям застосування	Вигоди для користувача
Надання програмного забезпечення державним установам	1. Центри надання адміністративних послуг	Зменшення часу очікування, прозорість обслуговування.
	2. Медичні заклади	Доступ до розкладу лікарів, сповіщення про прийом
	3. Банківські установи	Автоматизація процесів, зниження навантаження персоналу

Рішення для впровадження цієї системи є універсальним і може бути застосоване в різних сферах, таких як медичні заклади, банківські установи та центри надання адміністративних послуг.

Для аналізу техніко-економічних характеристик порівнювалися ключові показники проекту з аналогічними конкурентними рішеннями. Визначення характеристик ідеї проєкту зображені в таблиці 11.2.

Таблиця 11.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Конкурент Qmate	Конкурент Qmatic	Конкурент WaitWhile			
1	Інтеграція з API сторонніх сервісів	±	±	+	+		+	
2	Зручність користувацького інтерфейсу	+	+	+	+			+
3	Підтримка аналітичних інструментів	±	+	±	+		+	
4	Вартість впровадження	+	±	+	+			+

Розроблена система має сильні сторони в зручності інтерфейсу і вартості впровадження, але потребує оптимізації у підтримці аналітичних інструментів і інтеграції API сторонніх сервісів

11.2 Технологічний аудит ідеї проєкту

Технологічний аудит ідеї проекту показує, що більшість необхідних технологій для реалізації проекту вже доступні. У таблиці 11.3 було наведено інформацію про технологічну здійсненність ідеї проекту.

Таблиця 11.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Веб-інтерфейс	HTML, CSS, JavaScript	Наявні	Доступні
2	Інтеграція з SMS сповіщеннями	API провайдерів	Потрібно розробити	Потребує ліцензії
3	Інструменти аналітики	Google Analytics, Firebase	Потрібно розробити	Доступні

Згідно даної таблиці варто звернути увагу що декілька ідей проєкту варто розробити, після чого проєкт можна втілювати повністю у любий напрям.

11.3 Аналіз ринкових можливостей запуску стартап-проекту

Аналіз ринку показує високий потенціал для запуску стартапу в даній галузі, де зростає попит на автоматизацію процесів обслуговування. Однак на ринку вже присутня сильна конкуренція з кількома великими гравцями, що вимагає від нас розробки унікальних функцій та постійного вдосконалення продукту. Інформація про аналіз ринкових можливостей запуску стартап-проекту зображена у таблицях 11.4 – 11.13 відповідно.

Таблиця 11.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5-7
2	Загальний обсяг продаж, грн/ум.од	Невідомо
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Висока конкуренція
5	Специфічні вимоги до стандартизації та сертифікації	Юридичні
6	Середня норма рентабельності в галузі (або по ринку), %	50%

Таблиця 11.5 – Характеристика потенційних клієнтів стартап-проекту

п/п	Потреба	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги до товару
1	Зменшення часу очікування, Зручність обслуговування	Державні установи, Медичні заклади	-	Простота інтеграції, захист даних, Надійність, стабільність системи

Таблиця 11.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкуренція	На ринку вже є великий вибір подібних рішень, що ускладнює вихід на ринок	Розробка унікальних функцій, акцент на інноваціях та адаптивності
2	Зміни в законодавстві	Потенційні зміни в регулюваннях (наприклад, нові	Постійне відстеження змін, своєчасна
3	Високі витрати на впровадження	Початкові витрати на запуск та маркетинг можуть бути великими	Ініціація фазового запуску продукту, залучення

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
			інвесторів, пошук партнерів
4	Залежність від сторонніх постачальників	Потрібність інтеграції з іншими сервісами та платформами (API) може бути ризикованою правила GDPR чи локальні закони)	Пошук альтернативних постачальників або розробка власних рішень адаптація та впровадження відповідних змін у продукт

Таблиця 11.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростаючий попит на автоматизацію	Враховуючи сучасні тенденції в автоматизації та оптимізації бізнес-процесів	Швидке впровадження інновацій, акцент на автоматизації управління чергами
2	Популярність мобільних додатків	Зростаючий попит на мобільні рішення для бізнесу та споживачів	Розробка мобільних додатків для доступу до черг і обслуговування

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
3	Збільшення уваги до клієнтського досвіду	Покращення обслуговування клієнтів через технологічні рішення	Використання аналітики для поліпшення взаємодії з клієнтами та оптимізації їх досвіду
4	Можливість партнерств з державними установами	Співпраця з урядовими структурами в рамках оптимізації надання адміністративних послуг	Налагодження партнерських відносин для розвитку системи в державному секторі

Таблиця 11.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції	Олігополія з кількома великими компаніями, що мають контроль над ринком	Потрібно створювати унікальні переваги та застосовувати інноваційні стратегії
2. Рівень конкуренції	Високий на локальному рівні, низький на міжнародному	Агресивний маркетинг та фокус на специфічні ніші ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
3. Конкуренція за видами товарів	Міжгалузева конкуренція, конкуренція між різними технологіями управління чергами	Визначити чітке позиціонування продукту, що перевищує конкурентів за функціональністю
4.Інтенсивність конкуренції	Висока на локальному рівні (між певними постачальниками послуг)	Розробка сильних конкурентних переваг на рівні цін та функціональності

Таблиця 11.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Qmate Qmatic WaitWhile	Відсутні	Відсутні	Державні установи	Інші автоматизовані системи

Таблиця 11.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Інтеграція з існуючими системами	Зручність підключення до різних платформ, що дозволяє швидко впроваджувати систему
2	Мобільні додатки для користувачів	Збільшення доступності продукту для клієнтів через мобільні додатки, що значно розширює ринок
3	Цінова політика	Цінова політика є ключовим аспектом для потенційного користувача системи
4	Дизайн інтерфейсу	Зручний та інтуїтивно зрозумілий інтерфейс економить час роботи користувача з системою
5	Швидкодія та безперебійність роботи програмного продукту	Швидкість та безперебійність продукту є ключовим аспектом для потенційного користувача системи
6	Аналітичні інструменти	Можливість надавати детальну статистику для

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
		оптимізації роботи персоналу

Таблиця 11.11 – Порівняльний аналіз сильних та слабких сторін «Інформаційна система управління чергами державних»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	+1	+2	+3
1	Інтеграція з існуючими системами	10					+		
2	Мобільні додатки для користувачів	10							+
3	Цінова політика	14	+						
4	Дизайн інтерфейсу	17	+						
5	Швидкодія та безперебійність роботи програмного продукту	20		+					
6	Аналітичні інструменти	15					+		

Таблиця 11.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Зручний інтерфейс, лояльна цінова політика, доступність веб-додатку на різник платформах	Слабкі сторони: Необхідність адаптації під специфічні вимоги клієнтів
Можливості: залучення інвесторів для зростання стартапу, можливість співпраці з державними установами	Загрози: Висока конкуренція на ринку, високі витрати на маркетинг і просування

Таблиця 11.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Запуск пілотного проекту в окремих державних установах для тестування системи, збору відгуків та доопрацювання функціоналу.	Висока	6 місяців
2	Пошук партнерів серед державних установ для спільного впровадження системи через тендерні закупівлі.	Середня	1 рік

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
3	Проведення маркетингової кампанії для залучення уваги приватного сектору, включаючи бізнес-центри, банки, клініки.	Висока	9 місяців
4	Співпраця з IT-компаніями для інтеграції системи в їхні програмні продукти як додаткової послуги.	Низька	1,5 роки
5	Вихід на міжнародний ринок через участь у профільних виставках та грантових програмах.	Середня	2 роки

Згідно з результатами аналізу, зростаючий попит на автоматизацію в державних установах, медичних закладах та банках є основною можливістю для нашого стартапу. Важливою умовою для успішного виходу на ринок є адаптація до специфічних вимог кожної сфери та постійне удосконалення продукту, щоб підтримати конкурентоспроможність на фоні змін у законодавстві та технологічних вимог.

11.4 Розроблення ринкової стратегії проєкту

Розробка ринкової стратегії для цього стартап-проекту є ключовим етапом на шляху до успішного запуску та впровадження. Важливим завданням є визначення цільових груп потенційних споживачів, а також формулювання стратегій розвитку та конкурентної поведінки. Інформація про детальне розроблення ринкової стратегії проєкту зображена на таблицях 11.14 – 11.17.

Таблиця 11.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Державні установи, які надають адміністративні послуги (ЦНАПи,	Висока	Висока	Низька	Середня

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
	відділи міграції).				
2	Приватні бізнеси, орієнтовані на обслуговування клієнтів (банки, клініки, страхові компанії).	Середня	Середня	Висока	Висока
Які цільові групи обрано: усі					

Таблиця 11.15. – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Запуск пілотного проекту в державних установах	Концентрований маркетинг	Надійність, прозорість роботи, інтеграція з існуючими системами	Інноваційна стратегія

Таблиця 11.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Забирати існуючих у конкурентів	Часткове копіювання (автоматичне оновлення черги, аналітика)	Імітаційна з додаванням інновацій

Таблиця 1.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Прозорість обслуговування	Інноваційна стратегія	Надійність, гнучкість, зручність інтеграції	Інноваційність, довіра, прозорість

11.5 Розроблення маркетингової програми стартап-проекту

Успішний маркетинг стартап-проекту залежить від чіткого розуміння потреб цільових груп та розробки стратегії, яка ефективно комунікує переваги продукту. Важливі аспекти було визначено і описано у таблицях 11.18 – 11.22.

Таблиця 11.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Прозорість та організованість черги	Зменшення часу очікування	Прозорість обслуговування, реальний час оновлення черги
2	Автоматизація управління чергою	Зниження ручної роботи	Інтеграція з існуючими системами, автоматизація процесів
3	Зручність для кінцевих користувачів	Інтуїтивно зрозумілий інтерфейс	Гнучкий дизайн, адаптація для мобільних пристроїв

Таблиця 11.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Задоволення потреби в прозорості та організованості обслуговування, функціональна вигода – зменшення часу очікування.		
	Властивості/характеристики	М/Нм	Вр/Тх/Е/Ор
	1. Зручний інтерфейс	1. Нм	1. Е 2. Вр

Рівні товару	Сутність та складові		
II. Товар у реальному виконанні	2. Низька ціна 3. Інтеграція з сторонніми сервісами	2. М 3. Нм	3. Тх
	Якість: відповідність стандартам безпеки даних.		
	Пакування: цифрова платформа.		
	Марка: впізнавана система для державного сектору.		
III. Товар із підкріпленням	До продажу: демо-версія продукту, інформаційна підтримка.		
	Після продажу: регулярні оновлення, технічна підтримка.		
За рахунок чого потенційний товар буде захищено від копіювання: веб-застосунок, який адаптується під потреби кожного користувача			

Таблиця 11.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	1000-1500 грн/міс	2000-3000 грн/міс	Середній	1500-2500 грн/міс

Таблиця 11.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Вимога прямого контакту та	Продаж, впровадження	Прямий	Прямий збут через власний

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	технічної підтримки	, технічна підтримка		відділ продажів
2	Онлайн закупівлі для малого бізнесу	Продаж, базова консультація	Непрямий	Залучення партнерів для регіональних клієнтів

Таблиця 11.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Орієнтація на якість і ефективність	Галузеві конференції, профільні видання	Прозорість, зручність, надійність	Підвищення обізнаності про продукт	"Ефективність обслуговування у ваших руках"
2	Акцент на швидкості впровадження	Соцмережі, вебінари	Інноваційність, економія часу	Генерація лідів серед приватних бізнесів	"Швидкість та простота з нашою системою"

Висновки до розділу 11

У даному розділі було виконано маркетингові дослідження та аналіз ринкових можливостей стартап проєкту інформаційної системи управління чергами державних установ.

Згідно з проведеним аналізом техніко-економічних характеристик та порівняльним аналізом з конкурентами, система має сильні сторони в зручності інтерфейсу, ціновій політиці та інтеграції з існуючими системами.

Аналіз ринкових можливостей показав, що ринок автоматизації черг на стадії зростання, з високою конкуренцією, проте наявність унікальних функцій та можливість впровадження інновацій дають проекту конкурентні переваги. Визначення цільових груп клієнтів, таких як державні установи та приватні бізнеси, дозволяє точно сформулювати пропозицію для кожної групи, враховуючи їх специфічні потреби та вимоги до продукту. Ринкова стратегія проекту орієнтована на створення продукту з високою зручністю для користувачів та низьким порогом впровадження. Вибір стратегії диференційованого маркетингу дозволяє ефективно охопити кілька сегментів ринку з урахуванням їх специфіки.

Запропонований стартап-проект має всі необхідні передумови для успішного впровадження на ринку, враховуючи технологічну здійсненність, конкурентні переваги та правильно обрану маркетингову стратегію. Реалізація проекту не тільки відповідає актуальним вимогам ринку, але й забезпечує потенціал для подальшого розвитку та масштабування в інших сегментах ринку.

ВИСНОВКИ

У результаті виконаної роботи було розроблено методику створення інформаційної системи управління чергами в державних установах. Застосування адаптивного алгоритму розподілення клієнтів забезпечує рівномірне навантаження на працівників, що дозволяє уникнути перевантаження окремих робочих місць і скорочує середній час очікування. Інтерфейс системи, створений державною мовою, відповідає сучасним вимогам ергономіки, інтуїтивно зрозумілий і зручний для використання як персоналом, так і клієнтами.

Тестування підтвердило ефективність роботи системи в умовах стандартного завантаження, однак виявлено необхідність подальшої оптимізації для роботи в умовах високого навантаження.

Розроблена система повністю відповідає завданням дипломного проєкту та висунутим вимогам. Вона реалізує всі ключові функції, включаючи реєстрацію клієнтів, виклик до робочих місць, автоматичне розподілення клієнтів і моніторинг стану черги. Інтеграція функціоналу в один програмний застосунок забезпечує зручність управління та високу продуктивність.

Запропоновану систему можна впроваджувати в державних установах, таких як Центри надання адміністративних послуг, міграційні служби, паспортні столи та податкові інспекції. Впровадження сприятиме зменшенню адміністративних витрат завдяки автоматизації процесів і покращенню якості надання послуг. Система також здатна масштабуватися для задоволення потреб великих установ і різних категорій клієнтів.

Розробка демонструє значний прогрес у вирішенні проблем управління чергами, особливо в державному секторі. Інтеграція сучасних технологій, таких як адаптивний алгоритм розподілення та система звітності в реальному часі, сприяє підвищенню рівня довіри до державних установ і розвитку цифрових сервісів в Україні. Крім того, використання державної мови в інтерфейсі відповідає вимогам чинного законодавства та забезпечує доступність для користувачів.

У майбутньому доцільно спрямувати зусилля на такі напрями:

– оптимізація алгоритмів розподілення: для підвищення продуктивності системи в умовах пікового навантаження;

– інтеграція з іншими державними системами: наприклад, реєстраційними або платіжними сервісами, що розширить функціонал і забезпечить єдиний користувацький досвід;

– розробка мобільного додатка: для покращення взаємодії з клієнтами, надання можливості попередньої реєстрації та доступу до інформації про стан черги з будь-якого пристрою;

– підтримка нових технологій: включаючи впровадження машинного навчання для прогнозування часу обслуговування та аналізу ефективності роботи.

Впровадження розробленої системи дозволить державним установам значно підвищити ефективність роботи, зменшити час очікування громадян і покращити їхній досвід взаємодії з державними послугами. Запропонований підхід може стати основою для створення нових стандартів обслуговування у державному секторі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Повний посібник з систем керування чергою [електронний ресурс] – https://facit.ai/insights/queue-management-system (дата звернення: 04.09.2024

2. Qmatic Group. [електронний ресурс] – https://www.qmatic.com (дата звернення: 10.09.2024)

3. Servus System Integration [електронний ресурс] – https://www.ssi.com.ua/ (дата звернення: 10.09.2024)

4. Waitwhile Inc. [електронний ресурс] – https://waitwhile.com/ (дата звернення: 10.09.2024)

5. Клієнт-серверна архітектура – Wikipedia URL: https://en.wikipedia.org/wiki/Client%E2%80%93server_model (дата звернення: 1.10.2024)

6. MVC паттерн – Wikipedia URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (дата звернення: 1.10.2024)

7. Spring Framework – [електронний ресурс]: https://spring.io/guides (дата звернення: 3.10.2024)

8. REST API Design Guide – [електронний ресурс]: https://restfulapi.net (дата звернення: 15.10.2024)

9. MySQL– [електронний ресурс]: https://dev.mysql.com/doc (дата звернення: 13.10.2024)

10. Spring MVC Documentation – [електронний ресурс]: https://docs.spring.io/spring-framework/reference/web/webmvc.html (дата звернення: 11.10.2024)

11. Spring MVC CRUD with Example – [електронний ресурс]: https://www.geeksforgeeks.org/spring-mvc-crud-with-example/ (дата звернення: 11.10.2024)

12. Spring Boot Quick Start – [електронний ресурс]: https://spring.io/guides/gs/spring-boot (дата звернення: 08.10.2024)

13. Serving Web Content with Spring MVC – [електронний ресурс]: https://spring.io/guides/gs/serving-web-content (дата звернення: 08.10.2024)

14. Spring Security – [електронний ресурс]: https://spring.io/projects/spring-security (дата звернення: 20.10.2024)

15. Вступ до Jackson Framework – [електронний ресурс]: https://javarush.com/ua/groups/posts/uk.579.vstup-do-jackson-framework (дата звернення: 17.10.2024)

16. Вступ у JSON– [електронний ресурс]: https://www.json.org/json-uk.html (дата звернення: 17.10.2024)

17. Java AES Encryption and Decryption – [електронний ресурс]: https://www.baeldung.com/java-aes-encryption-decryption (дата звернення: 23.10.2024)

18. Registration with Spring Security – [електронний ресурс]: https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt

19. JavaScript Guide – [електронний ресурс]: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 23.10.2024)

20. The Modern JavaScript Tutorial – [електронний ресурс]: https://javascript.info/ (дата звернення: 17.10.2024)

21. HTML Guide on MDN – [електронний ресурс]: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 30.10.2024)

22. CSS Guide on MDN – [електронний ресурс]: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 30.10.2024)