Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 3

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Виконала

Студентка групи ІА-31:

Трегуб К. В.

Перевірив:

Мягкий М. Ю.

Київ 2025

Зміст

Оглавление

Мета:

Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Посилання на репозиторій: https://github.com/trehubkira05/BeautySalonBooking

Хід роботи:

Тема : Сервіс бронювання послуг для салону краси

1) Ознайомитись з короткими теоретичними відомостями.
2) Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу

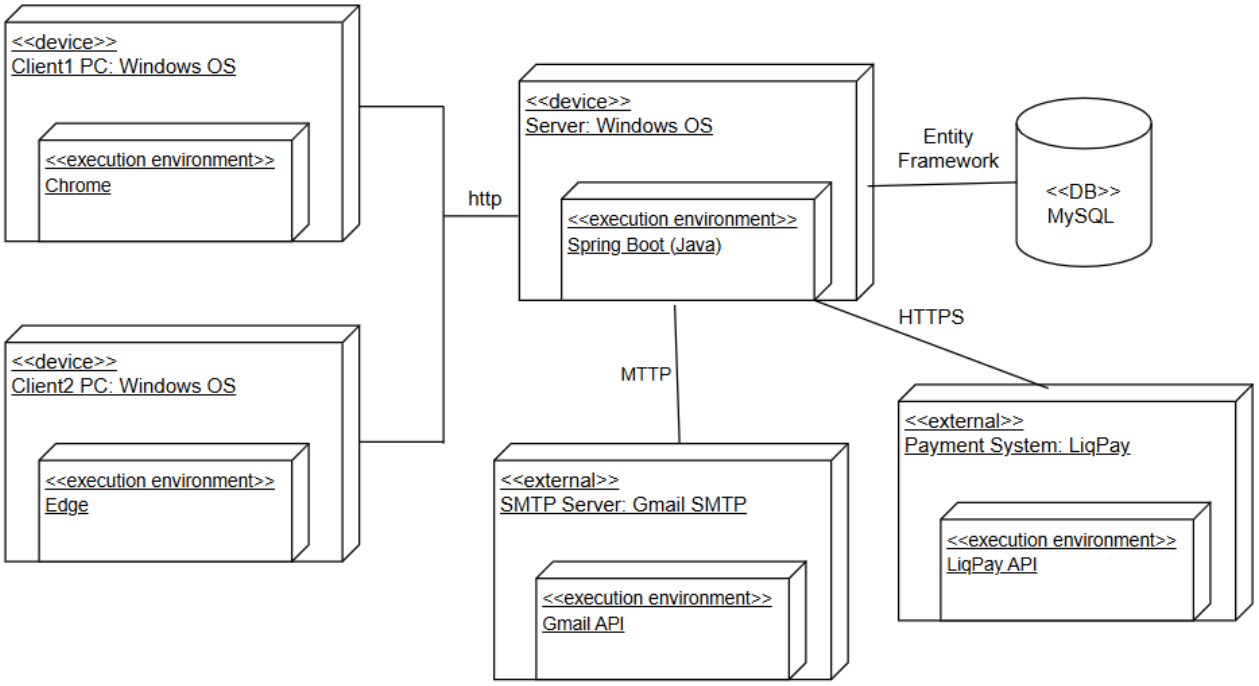

Рис.1 – Діаграма розгортання

На діаграмі розгортання (Рис. 1) зображено фізичну архітектуру веб-застосунку " Сервіс бронювання послуг для салону краси ". Система побудована за клієнт-серверною моделлю та включає такі вузли:

- Клієнтські вузли (Client PC): Представлені як персональні комп'ютери під керуванням Windows OS. Вони взаємодіють із системою через веббраузери (Chrome, Edge), які виступають середовищем виконання для клієнтської частини.
- Серверний вузол (Server): Це центральний вузол, що працює на Windows OS та виконує основну логіку додатку у середовищі Spring Boot. Він відповідає за обробку запитів від клієнтів.

- Вузол бази даних (Database): Фізично відокремлений сервер, на якому розгорнуто систему управління базами даних MySQL для надійного зберігання всіх даних системи.
- Для надсилання email-нагадувань система взаємодіє з зовнішнім SMTP-сервером (Gmail SMTP), використовуючи протокол SMTP.
- "Payment System" - відповідає за обробку онлайн-оплат. Сервер додатку взаємодіє з ним через захищений HTTPS API, передаючи дані транзакції та отримуючи підтвердження платежу.

3) Розробити діаграму компонентів для проєктованої системи

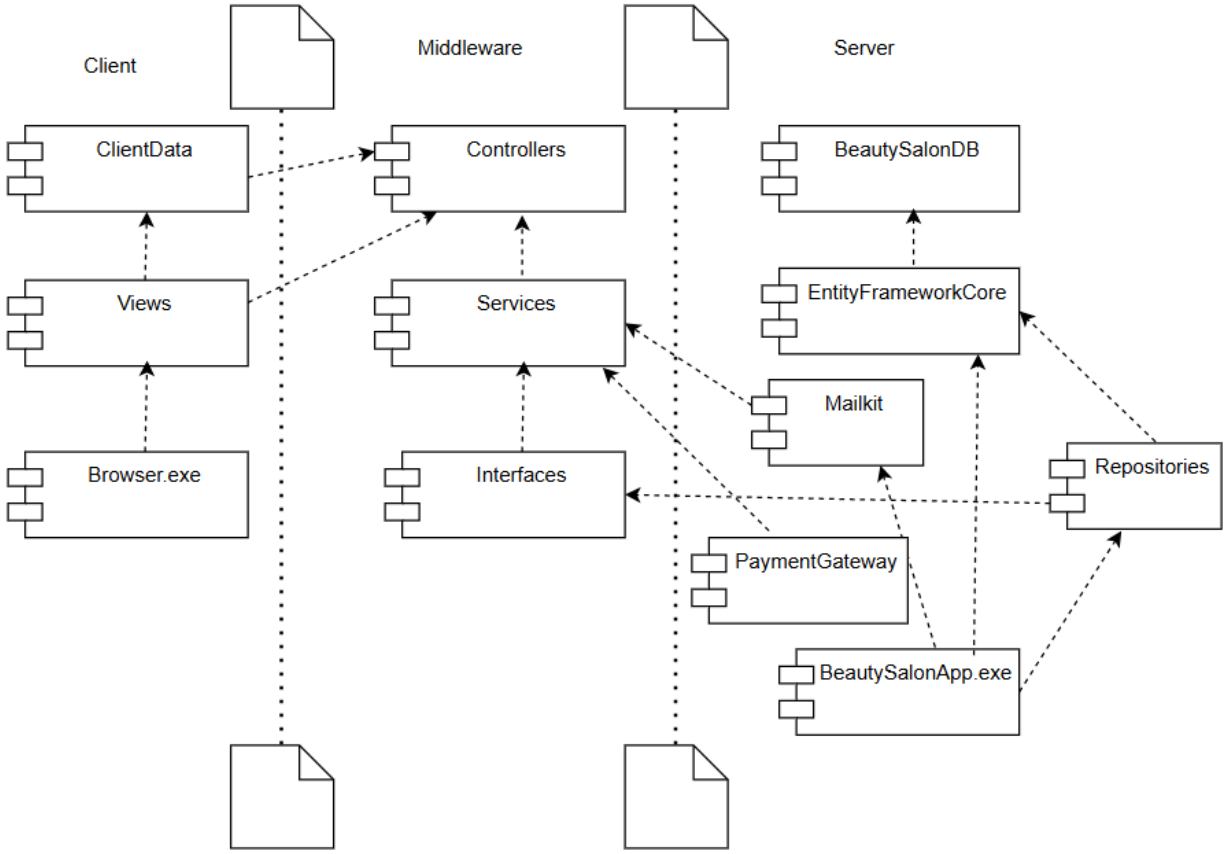

Рис.2 – Діаграма компонентів

На діаграмі зображено основні компоненти програмної системи. У лівій частині подані клієнтські файли запуску (.exe) та бібліотеки (.dll), які відповідають за інтерфейс користувача і взаємодію з веб-додатком. У правій частині показано серверні компоненти (.exe та .dll), які реалізують бізнес-логіку, роботу з базою даних через Entity Framework та інтеграцію з зовнішнім email-сервісом та системою оплати. У центральній частині розташовані модулі middleware, які містять загальні інтерфейси, З контракти API та сервіси, що забезпечують узгоджену взаємодію між клієнтом і сервером. Таким чином, клієнтська і серверна частини системи обмінюються інформацією через спільний набір інтерфейсів і даних, визначених у middleware.

4) Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі

1. Створення бронювання

Передумови:
Клієнт авторизований у системі; салон має доступні послуги та вільних майстрів у розкладі.

Постумови:
Створено нове бронювання, яке збережене у системі та прив’язане до конкретного майстра, дати й часу.

Взаємодіючі сторони:
Клієнт, Сервіс бронювання послуг салону краси.

Короткий опис:
Цей сценарій визначає процес створення нового бронювання клієнтом.

Основний потік подій:
1. Клієнт переглядає список послуг.
2. Клієнт вибирає бажану послугу.
3. Система відображає доступних майстрів і вільні часові слоти.
4. Клієнт обирає майстра та дату з часу.
5. Система формує попереднє бронювання.
6. Клієнт підтверджує створення бронювання.
7. Система зберігає бронювання і повідомляє про успіх.

Винятки:
- Обраний майстер недоступний у вибраний час — система повідомляє про неможливість бронювання, клієнт повертається до вибору часу.
- Відсутні активні послуги — система повідомляє про помилку.

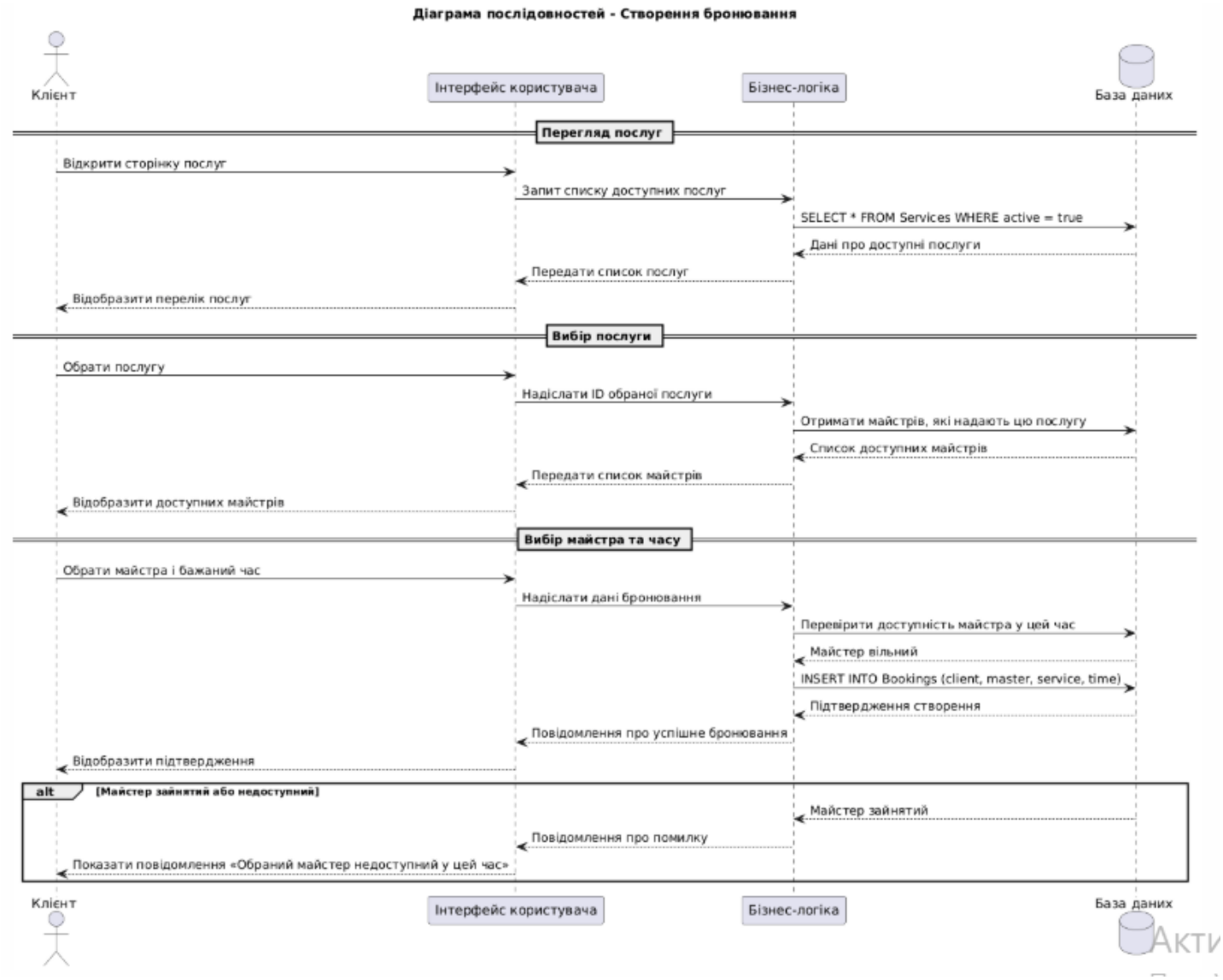

Рисунок 3.1. Діаграма послідовності «Створення бронювання»

Основний потік:

1. Перегляд доступних послуг

2. Вибір послуги
Клієнт обирає бажану послугу зі списку.
Інтерфейс користувача показує клієнту доступних майстрів і час для запису.

3. Вибір майстра та часу
Клієнт обирає конкретного майстра та зручний час для відвідування.

4. Перевірка доступності
Альтернатива 1 (Помилка):
Якщо майстер зайнятий у цей час, бізнес-логіка надсилає повідомлення про помилку в інтерфейс користувача, який інформує клієнта про неможливість бронювання.

Альтернатива 2 (Успішне бронювання):
Якщо майстер вільний, бізнес-логіка створює новий запис у таблиці бронювань
База даних підтверджує створення запису.
Бізнес-логіка надсилає підтвердження в інтерфейс користувача, а той повідомляє клієнта про успішне бронювання.

2. Керування розкладом майстра

Передумови:
Адміністратор авторизований у системі; у базі є зареєстровані майстри та перелік послуг.

Постумови:
Розклад майстра створено або оновлено; зміни збережено у системі.

Взаємодіючі сторони:
Адміністратор, Система бронювання послуг салону краси.

Короткий опис:
Цей сценарій визначає процес створення або редагування розкладу роботи майстра адміністратором салону.

Основний потік подій:
1. Адміністратор відкриває розділ «Майстри».
2. Система відображає список зареєстрованих майстрів.
3. Адміністратор обирає потрібного майстра.
4. Система відображає поточний розклад роботи майстра.
5. Адміністратор додає або змінює робочі дні та часові проміжки.
6. Адміністратор підтверджує зміни.
7. Система оновлює розклад і повідомляє про успішне збереження.

Винятки:
- Майстер не знайдений — система повідомляє про помилку.
- Часові проміжки перетинаються або некоректні — система повідомляє про неможливість збереження і пропонує виправити дані.

Рисунок 3.2. Діаграма послідовності «Керування розкладом роботи майстра»

Основний потік:

1. Відкриття розділу «Майстри»

2. Вибір конкретного майстра

Адміністратор обирає потрібного майстра зі списку.
Отриманий розклад повертається адміністратору через інтерфейс користувача.

3. Редагування розкладу

Адміністратор додає або змінює робочі дні та часові проміжки для обраного майстра.

4. Альтернативні варіанти:

Якщо дані коректні — система зберігає зміни у базі даних, після чого адміністратор отримує повідомлення про успішне оновлення розкладу.

Якщо виявлено помилки (наприклад, часові перетини або некоректний формат даних) — система повідомляє про неможливість збереження і пропонує виправити введену інформацію.

5) На основі спроєктованих діаграм розгортання та компонентів доопрацювати

програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинна бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

Реалізація системи

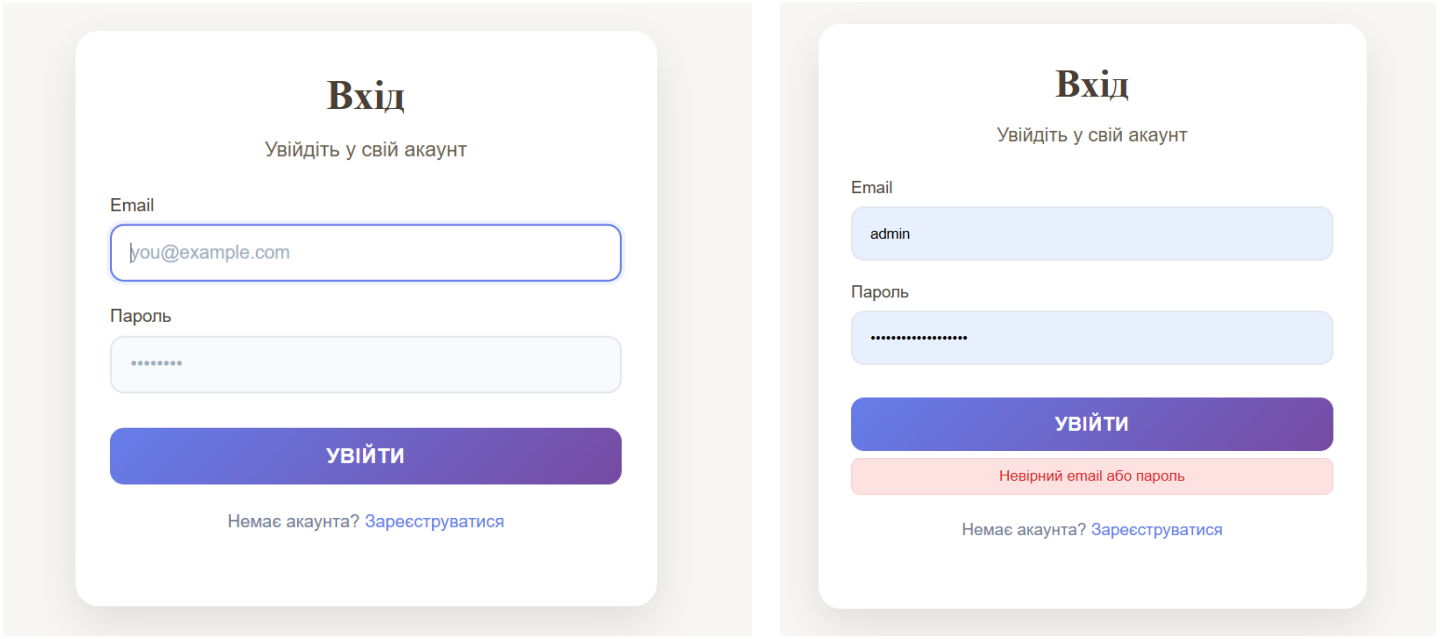

Рисунок 4.1. Форма Авторизації

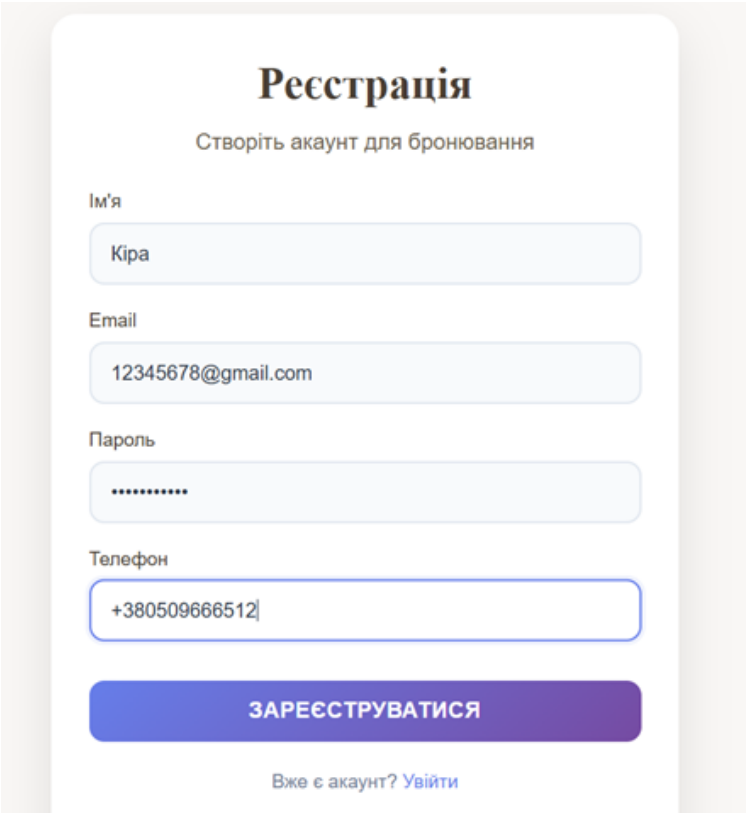

Рисунок 4.2. Форма Реєстрації

```
mysql> select * from users;
+--------------------------------------+-------+--------------------+----------------------+---------------+--------+
| user_id                              | name  | email              | password             | phone         | role   |
+--------------------------------------+-------+--------------------+----------------------+---------------+--------+
| 550e8400-e29b-41d4-a716-446655440000 | Admin | admin@beauty.com   | wT4@pJ8u#Vz!cE9rQy2x | +380501234567 | ADMIN  |
| c439c474-b4f9-4d0f-9735-7b87eb1ba8a7 | K?pa  | 12345678@gmail.com | qwerty1234-          | +380509666512 | CLIENT |
+--------------------------------------+-------+--------------------+----------------------+---------------+--------+
```

Рисунок 4.3. Таблиця в SQL з доданим користувачем

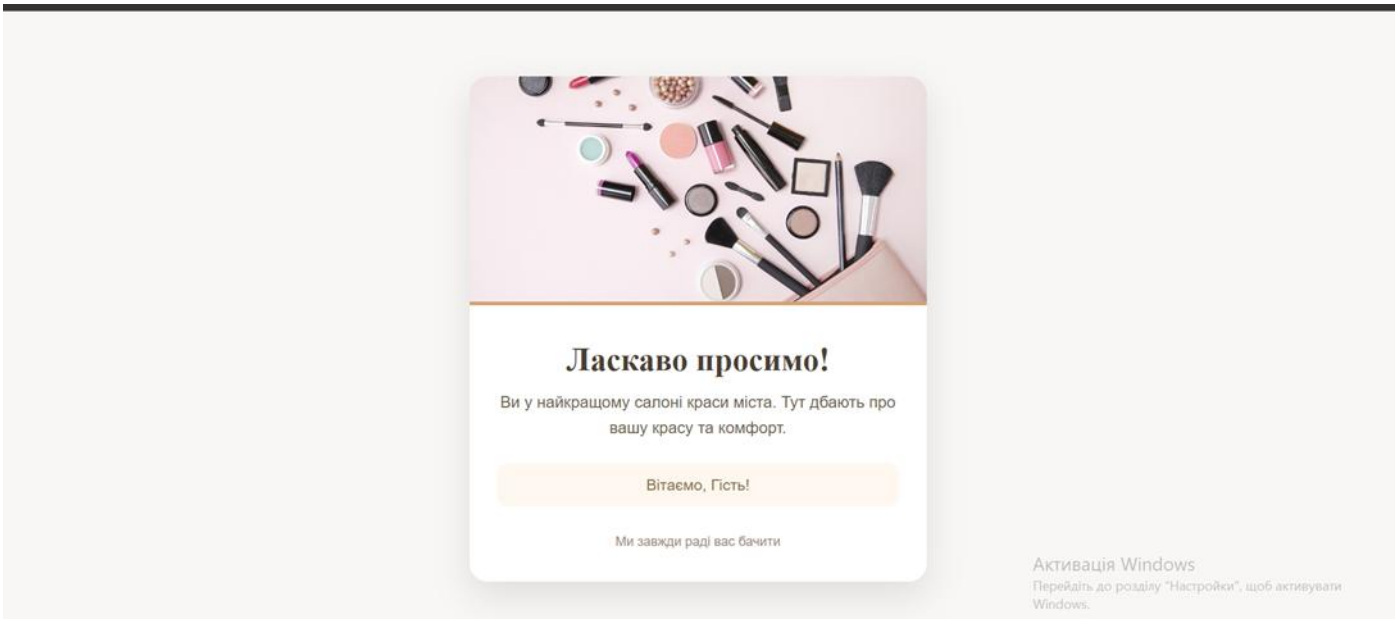

Рисунок 4.4. Успішна авторизація

Висновок

Отже, у процесі виконання лабораторної роботи було досягнуто визначеної мети: освоєно створення діаграм розгортання та компонентів для проектування системи, які дають змогу зобразити її фізичну архітектуру, розподіл програмних модулів між вузлами та їхні взаємозв'язки. Крім того, опановано розробку діаграм взаємодії, зокрема послідовних діаграм, на основі сценаріїв, розроблених у попередній лабораторній роботі. Це допомогло глибше зрозуміти логіку взаємодії об'єктів у системі та порядок виконання операцій, що є ключовим кроком у моделюванні програмного забезпечення та підготовці до його реалізації

Контрольні питання

1) Діаграма розгортання — UML-діаграма, що відображає фізичну структуру системи: на яких пристроях (вузлах) розміщуються програмні компоненти та як вони взаємодіють у реальному середовищі.
2) Види вузлів:

Апаратні вузли — фізичні пристрої (сервер, комп'ютер, смартфон).
Програмні вузли — середовища виконання (операційна система, контейнер, віртуальна машина).

1) Види зв'язків на діаграмі розгортання:

Асоціація — фізичне або логічне з'єднання між вузлами.
Залежність — вказує, що один вузол або компонент використовує ресурси іншого.

2) Елементи діаграми компонентів:

Компоненти, інтерфейси, порти, залежності, пакети, підсистеми.

3) Зв’язки на діаграмі компонентів показують взаємозалежність між компонентами, тобто який компонент використовує або реалізує певний інтерфейс іншого.
4) Види діаграм взаємодії:
Діаграма послідовностей. Діаграма комунікацій.
Діаграма часу.
Діаграма взаємодії огляду.
5) Діаграма послідовностей — показує порядок викликів і повідомлень між об’єктами під час виконання певного сценарію або процесу.
6) Ключові елементи діаграми послідовностей: актори, об’єкти, лінії життя, повідомлення (синхронні та асинхронні), області активації.
7) Зв’язок із діаграмами варіантів використання — діаграми послідовностей деталізують дії, що виконуються для реалізації кожного варіанту використання.
8) Зв’язок із діаграмами класів — об’єкти, що взаємодіють на діаграмі послідовностей, є екземплярами класів, описаних на діаграмі класів, а повідомлення відповідають викликам їх методів.
9) Зв’язок із діаграмами варіантів використання — діаграми послідовностей деталізують дії, що виконуються для реалізації кожного варіанту використання.
10) Зв’язок із діаграмами класів — об’єкти, що взаємодіють на діаграмі послідовностей, є екземплярами класів, описаних на діаграмі класів, а повідомлення відповідають викликам їх методів.