НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

комп'ютерних наук
(назва кафедри)

_____ Голуб Б. Л.
(підпис) (ПІБ)

"___" _____ 20 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Програмне забезпечення веб-орієнтованої інформаційної системи організації корпоративного харчування»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

с.т викладач _____ Вайганг Г.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

с.т викладач _____ Міловідов Ю.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Лебедєв А. В.
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

комп'ютерних наук

доцент к.т.н. Голуб Б. Л.

(науковий ступінь, вчене звання) (підпис) (ПІБ)

"____" _____20_____р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Лебедєву Андрію Валерійовичу

(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи: Програмне забезпечення веб-орієнтованої інформаційної системи організації корпоративного харчування
Затверджена наказом ректора НУБіП України від "_16_" грудня 2024 р. №2249 "С"
Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)
Вихідні дані до бакалаврської кваліфікаційної роботи
Розробка веб-сервісу для організації корпоративного харчування. Аналіз впливу цифрових рішень на ефективність замовлення та доставки обідів у компаніях.

Перелік питань, які потрібно розробити:

1. Аналіз сучасних систем корпоративного харчування.
2. Проектування інформаційного та програмного забезпечення вебсервісу.
3. Розробка системи з урахуванням ролей компанії, клієнта та постачальника.
4. Тестування та підготовка системи до впровадження.

Дата видачі завдання "_____"_____20___ р.

Керівник бакалаврської кваліфікаційної роботи _____ Міловідов Ю.О.

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Лебедєв А. В.

(підпис) (прізвище та ініціали студента)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – інтерфейс прикладного програмування (Application Programming Interface).

JWT – токен авторизації у форматі JSON Web Token.

ORM – технологія об’єктно-реляційного відображення (Object-Relational Mapping).

REST – архітектурний стиль побудови веб-сервісів (Representational State Transfer).

CRUD – базові операції роботи з даними: створення, читання, оновлення, видалення (Create, Read, Update, Delete).

DRF – Django REST Framework – бібліотека для створення REST API у Django.

UI – користувацький інтерфейс (User Interface).

UX – користувацький досвід (User Experience).

SPA – односторінковий вебзастосунок (Single Page Application).

JSON – текстовий формат обміну структурованими даними (JavaScript Object Notation).

SQL – мова структурованих запитів для взаємодії з реляційними базами даних (Structured Query Language).

ВСТУП

У сучасному діловому середовищі організація харчування працівників є важливою складовою корпоративної культури, соціального пакету та загальної турботи компанії про добробут своїх співробітників. Забезпечення якісного, своєчасного та зручного обіду позитивно впливає на продуктивність праці, рівень задоволеності персоналу, а також знижує часові втрати, пов’язані з пошуком альтернативних способів харчування.

Традиційні підходи до замовлення обідів у компаніях зазвичай базуються на ручному введенні замовлень, передачі даних у вигляді електронних таблиць або навіть усних домовленостей із постачальниками. Це створює зайве навантаження на адміністративний персонал, призводить до помилок у комунікації, дублювання замовлень або їх втрати, а також унеможливлює централізований контроль витрат. У зв’язку з цим постає потреба у створенні сучасного цифрового рішення, яке дозволить автоматизувати весь процес — від реєстрації учасників системи, створення меню та подій обіду, до оформлення та обробки замовлень і збору статистичних даних.

Актуальність теми зумовлена зростаючими вимогами до ефективності внутрішніх бізнес-процесів у компаніях, що впроваджують принципи цифрової трансформації. Ідея централізованого вебсервісу, який дозволяє компаніям взаємодіяти з постачальниками їжі, а працівникам — швидко замовляти обіди, є своєчасною, затребуваною і має високий потенціал для масштабування в корпоративному середовищі.

Метою даної роботи є розробка вебсервісу організації корпоративного харчування, який забезпечує зручну, безпечну та ефективну взаємодію між трьома основними типами користувачів: компаніями (адміністраторами харчування),

працівниками компаній (клієнтами) та постачальниками їжі. Сервіс має забезпечувати повний цикл управління обідами: створення подій, приєднання працівників до компаній, формування меню, прийом та обробка замовлень, підтвердження співпраці між компанією та постачальником, а також аналітику замовлень.

Методи та технології, що використовувались у процесі розробки, включають сучасний вебфреймворк Django, розширення Django REST Framework для створення API, JWT (JSON Web Token) для реалізації механізмів автентифікації та авторизації, а також OpenAPI (Swagger) для документування та інтеграційної зручності. Фронтенд реалізовано окремо, із застосуванням React та бібліотеки Axios, що забезпечує взаємодію з бекендом через REST API.

Апробація результатів роботи відбувалася під час тестування розробленого програмного забезпечення в умовах навчального середовища, у вигляді виконання лабораторних і курсових завдань, консультацій з викладачами та публічного захисту. Окремі ідеї та напрацювання було представлено у вигляді презентації на факультетських заходах із проєктної діяльності.

Структура пояснювальної записки охоплює логіку проектування веб сервісу та реалізації її компонентів. Зміст розділів подано у такій послідовності:

Розділ 1: системний аналіз предметної області, визначення функціональних і нефункціональних вимог, постановка задачі.

Розділ 2: проектування програмної системи — моделювання сутностей, розробка логічної моделі БД, діаграми класів, компонентів і кооперацій.

Розділ 3: вибір СУБД, опис інформаційної бази, програмна реалізація модулів, алгоритмізація логіки.

Розділ 4: тестування та впровадження, опис інсталяційного пакету, вимоги до середовища.

1. Системний аналіз предметної області

1.1 Опис предметної області

Сфера, яку охоплює розроблене програмне забезпечення, стосується організації та автоматизації процесів корпоративного харчування — тобто забезпечення працівників обідами у межах робочого середовища. У сучасному світі, де комфорт і продуктивність персоналу прямо впливають на успіх бізнесу, компанії все частіше звертають увагу не лише на заробітну плату, а й на створення зручних умов праці. І одним із таких важливих елементів стає організоване, збалансоване харчування.

На практиці далеко не всі компанії мають у своєму розпорядженні власні їдальні чи кухні. Часто доводиться звертатися до сторонніх кейтерингових сервісів або договірних постачальників. Проте організувати цей процес вручну — складно: погодження списків, комунікація через месенджери, Excel-таблиці, роздруківки замовлень, телефонні дзвінки — усе це займає час, створює плутанину й підвищує ризик помилок. Менеджери втрачають години на обробку повторюваних задач, а працівники — терпіння.

У результаті компанії стикаються з рядом типових проблем: дублювання замовлень, невдоволення від поганої комунікації, порушення бюджету, відсутність аналітики та контролю. Відтак виникає потреба у сучасному інструменті, який візьме ці завдання на себе — швидко, просто і без зайвих турбот.

Цифровий вебсервіс, що пропонується в межах цієї роботи, якраз і покликаний закрити цю нішу. Він дозволяє компаніям централізовано керувати всіма аспектами харчування: обирати постачальників, створювати "події обіду", приєднувати працівників, слідкувати за замовленнями та витратами. Працівники, зі свого боку, мають зручний особистий кабінет, де можуть обирати страви з меню

й оформлювати замовлення в кілька кліків. А постачальники отримують інструменти для управління своїм меню, підтвердження співпраці з компаніями й обробки замовлень.

Сервіс розділяє ролі трьох основних учасників — працівника, компанії та постачальника — й надає кожному саме той функціонал, який йому потрібен. Все це відбувається в єдиному інтерфейсі, у реальному часі, без потреби у паперових документах чи нескінченних дзвінках.

Таким чином, запропоноване рішення сприяє цифровій трансформації повсякденного процесу, який досі часто залишався в «ручному» форматі. Це не просто програмний інструмент — це новий підхід до турботи про працівника і до організації внутрішніх процесів у компанії.

1.1.1 Ключові учасники системи

Розробленеий веб-сервіс передбачає взаємодію кількох типів користувачів, кожен з яких виконує певну роль у загальному процесі організації корпоративного харчування. Для ефективного функціонування системи були виділені три ключові групи учасників, які відображають реальні ролі в бізнес-процесах: компанії, працівники (клієнти) та постачальники. Вони мають чітко визначені функціональні обов’язки та повноваження в системі, що дозволяє забезпечити логічну розділеність відповідальностей і мінімізувати конфлікти при обміні даними.

Компанії (роботодавці)

Компанії виступають основними замовниками послуг з організації харчування. Вони створюють власний профіль у системі, в якому вказують загальну інформацію про організацію, контактні дані, логотип, локацію та інші атрибути.

Після реєстрації компанія має змогу:

- додавати до себе працівників (через підтвердження запитів);
- створювати події харчування на певні дати (наприклад, "обід на понеділок");
- встановлювати обмеження на максимальну вартість обіду на одного співробітника;
- обирати постачальників, з якими готова співпрацювати;
- переглядати статуси замовлень по працівниках та загальну статистику витрат.

Компанія виконує роль адміністратора — вона є ініціатором взаємодії, формує правила гри, а також відповідає за фінансову та логістичну складову процесу. Впровадження цифрової системи дозволяє компанії значно зменшити адміністративне навантаження, яке раніше покладалось на офіс-менеджерів або керівників відділів.

Клієнти (працівники компаній)

Кожен клієнт є звичайним користувачем — працівником певної компанії, який бажає скористатися послугою харчування. У системі клієнт проходить реєстрацію, вказує свою роль, а далі подає запит на приєднання до компанії, в якій він працює. Після підтвердження з боку компанії, він отримує доступ до доступних подій та меню.

Клієнт може:

переглядати список доступних обідів на конкретну дату;

обирати страви з меню, сформованого постачальниками;

оформляти замовлення та змінювати його до дедлайну;

бачити статус свого замовлення (прийнято, пакується, транспортується тощо);

отримувати історію замовлень і переглядати свій ліміт.

Таким чином, система дає працівнику відчуття контролю та зручності — він більше не мусить звертатися до менеджера, чекати Excel-таблиці чи дзвонити постачальнику. Усе здійснюється самостійно через вебінтерфейс.

Постачальники (ресторани, кейтерингові служби)

Постачальники в системі відповідають за створення меню та фізичне приготування/доставку їжі. Вони створюють профіль, додають категорії страв (наприклад, "супи", "салати", "десерти") та самі страви з описом, фото і ціною. Кожна страва може бути використана для формування обіду в межах подій, створених компаніями.

Крім того, постачальник:

- отримує запити на співпрацю від компаній і може підтвердити або відхилити їх;
- переглядає замовлення, які були сформовані для його страв;
- оновлює статуси кожного замовлення, щоб клієнт і компанія бачили прогрес;
- має доступ до статистики замовлень — які страви популярні, скільки порцій продано, які компанії найактивніші тощо.

Цей тип користувача також отримує прямий зиск від використання системи, адже вона зменшує навантаження на операторів call-центрів, знижує ймовірність помилок у замовленнях та покращує планування об’ємів.

Загальна логіка взаємодії

У центрі системи — обідня подія (LunchEvent), яка створюється компанією на певну дату. Кожному підтвердженому клієнту в рамках цієї події створюється замовлення, яке він може наповнити стравами з меню обраних постачальників. Постачальники готують і доставляють страви, оновлюючи статуси, а компанія контролює бюджет та аналітику.

Таким чином:

- система передбачає реєстрацію користувачів трьох типів;
- підтримує гнучку модель "компанія ↔ постачальники ↔ клієнти";
- забезпечує повну автоматизацію ланцюга від вибору страви до доставки;
- надає прозору систему обліку, контролю та взаємодії в корпоративному середовищі.

Всі ці процеси охоплюють повний цикл взаємодії, пов'язаний із замовленням та доставкою обідів у межах однієї або декількох компаній, забезпечуючи сучасну, зручну, масштабовану та контрольовану інфраструктуру для корпоративного харчування.

1.2 Аналіз вимог до веб-сервісу

1.2.1 Функціональні вимоги

Таблиця 1.1

Таблиця з функціональними вимогами до веб-сервісу

Функціонал	**Опис**
Реєстрація та автентифікація користувачів	Система повинна підтримувати реєстрацію користувачів трьох типів: клієнт, компанія , постачальник. Користувачі повинні мати можливість авторизуватись через JWT.
Управління профілем користувача	Кожен тип користувача має окремий профіль із відповідними атрибутами (логотип, опис, контакти тощо).
Взаємодія між компаніями та клієнтами	Клієнт може подати запит на приєднання до компанії. Компанія може затвердити чи відхилити запит. Затверджені клієнти отримують доступ до заходів компанії.
Взаємодія між компаніями та постачальниками	Компанія може надіслати запит на співпрацю з постачальником. Постачальник може підтвердити чи відхилити співпрацю.

Таблиця 1.1(закінчення)

Управління обідніми заходами (LunchEvent)	Компанія може створити обідній захід на певну дату. При створенні заходу для кожного підтвердженого клієнта створюється порожнє замовлення (Order).
Управління меню постачальника :	Постачальник може створювати категорії та підкатегорії страв. Шкіра має назву, опис, ціну, зображення.
Оформлення замовлень клієнтами	Клієнти можуть додавати страви до свого замовлення. Шкіра має статус виконання (прийнято, пакується, в дорозі тощо).
Звіти та аналітика	Компанії можуть переглядати щомісячну статистику замовлень та сум
Пошук та перегляд профілів	Клієнти можуть шукати компанії. Компанії можуть шукати постачальників. Постачальники можуть переглядати компанії, які надіслали запити на співпрацю.

1.2.2 Нефункціональні вимоги

Таблиця 1.2

Таблиця з нефункціональними вимогами до веб-сервісу

Категорія	**Вимога**	**Опис**
Безпека	Авторизація через JWT	Усі запити до захищених API виконуються лише з дійсним токеном доступу.
Документованість	Swagger-документація API	Усі ендпоінти задокументовані через drf_yasg (OpenAPI 3.0).
Масштабованість	Підтримка великої кількості запитів	Архітектура сервісу дозволяє працювати з великою кількістю користувачів.

1.3 Моделювання предметної області

Моделювання предметної області полягає в аналізі основних об’єктів (сутностей) системи, їх атрибутів та зв’язків між ними. У системі організації корпоративного харчування головними є три типи користувачів: клієнти, компанії, постачальники, які взаємодіють через створення замовлень на обіди.

Ключові сутності системи:

1. Користувач (User): базова сутність, яка представляє всі типи зареєстрованих осіб у системі. Користувач має унікальне ім'я, електронну пошту, пароль, а

також роль — client, companyабо provider. В залежності від ролі, користувач отримує доступ до відповідного функціоналу.

2. Компанія (Company): юридична особа, яка організовує харчування для своїх співробітників. Компанія створює обідні події (Lunch Events), обирає постачальників, встановлює бюджет, затверджує запити клієнтів на приєднання.

3. Постачальник (Provider): суб'єкт, який формує меню, готує їжу, бере участь у подіях компаній.

4. Обідній захід (LunchEvent): подія, організована компанією на певну дату, в рамках якої працівники можуть робити замовлення з доступного меню. LunchEvent є точкою входу для створення замовлень.

5. Замовлення (Order): набір страв, обраних клієнтом в межах конкретного обіднього заходу. Кожне замовлення складається з кількох OrderItem — окремих позицій меню із зазначеним кількістю та статусом доставки.

6. Категорія страв (DishCategory) та Страва (Dish): логічна структура меню. Страви групуються у категорії (наприклад, "Перші страви", "Напої"). Шкіра має назву, опис, вартість, зображення і належить до певної категорії постачальника.

7. JoinRequest: зв'язує клієнта з компанією. Клієнт подає запит на приєднання до обраної компанії, після чого його запит може бути ухвалений адміністрацією компанії.

8. CompanyProvider: реалізує зв'язок між компанією та постачальником. Компанія може подати запит на співпрацю з певним постачальником, який підтверджує або відхиляє співпрацю. Зв'язок є взаємним та унікальним для кожної парі.

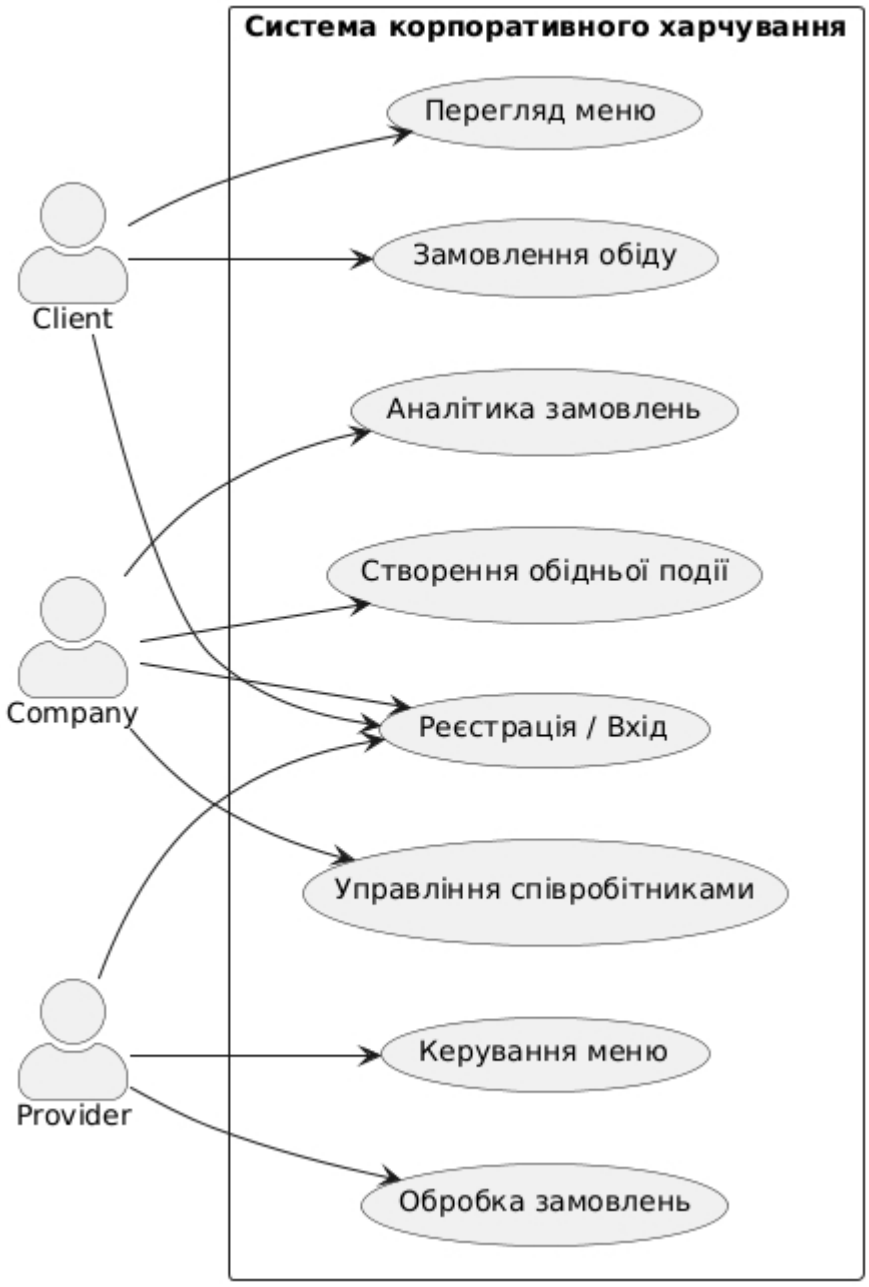

Рис.1.1 - Діаграма прецедентів

Всі виявлені сутності реалізовані у вигляді об'єктно-орієнтованих моделей Django , які взаємодіють між собою через відношення зовнішніх ключів або унікальних об'єднаних ключів (для зв'язків "багато-до-багатьох"). Зовнішній інтерфейс реалізовано за допомогою REST API , що забезпечує чітку розмежованість логіки фронтенду та бекенду та дозволяє масштабувати систему в майбутньому — наприклад, інтегрувати мобільний додаток чи сторонні сервіси.

1.4 Огляд інформаційних джерел та існуючих рішень

Для побудови ефективного вебсервісу організації корпоративного харчування було проведено аналіз наявних інформаційних джерел, прикладних рішень та конкурентних систем. Метою аналізу є виявлення сильних і слабких сторін існуючих продуктів для формулювання вимог до власного проєкту.

1.4.1 Аналіз існуючих вебрішень

Таблиця 1.3

Таблиця з аналізом існуючих рішень

Сервіс	**Опис та функціонал**	**Особливості**
ezCater	Онлайн-платформа для замовлення корпоративного кейтерингу. Компанії можуть створювати замовлення на події.	Потужна база постачальників, автоматизація замовлень, відсутність підтримки ролі клієнта.
OfficeLunch	Простий сервіс для обідів в офіс. Працівники обирають меню із фіксованих варіантів, компанія сплачує рахунок.	Обмежене налаштування меню, орієнтовано на один тип страв, відсутній механізм ролей.
FoodStorm	Програмне забезпечення для кейтерингу з CRM-функціями, управлінням доставкою та звітністю.	Комерційний продукт для постачальників, слабка інтеграція з компаніями-клієнтами.

Таблиця 1.3 (закінчення)

Lunch.ua (Україна)	Український сервіс обідів в офіс. Обирається меню з шаблонів на тиждень.	Мінімальна інтеграція з IT-середовищем, слабка підтримка персоналізації, не SaaS.

1.4.2 Основні недоліки існуючих рішень

Відсутність гнучкої ролевої моделі: більшість систем не підтримують одночасно клієнтів, компанії та постачальників як окремих учасників.

Обмежені можливості кастомізації меню або подій: замовлення часто прив’язані до шаблонів чи одного варіанту страв.

Закритість до інтеграції: частина систем не має API або документації.

Обмежений фокус: або лише для постачальників, або лише для компаній (без залучення працівників).

1.4.3 Мотивація створення власного рішення

На основі огляду виявлено, що жодне з існуючих рішень не задовольняє одночасно такі вимоги:

- Підтримка трьох окремих ролей (client, company, provider)
- Гнучке управління обідами як подіями
- Індивідуальні замовлення клієнтів на кожен захід
- Внутрішня статистика для компаній
- Відкритий REST API для подальшої інтеграції з мобільними/веб-клієнтами

Це стало підґрунтям для створення власного вебсервісу, який забезпечить повний цикл взаємодії між компанією, постачальником і працівником в одному середовищі.

1.5 Постановка завдання

Мета роботи: Розробити вебсервіс для організації корпоративного харчування, який дозволяє компаніям взаємодіяти з постачальниками, а працівникам — зручно замовляти обіди через єдину платформу.

Завдання дослідження:

1. Провести аналіз предметної області та визначити ключові сутності.
2. Побудувати інформаційну модель системи.
3. Розробити REST API з підтримкою трьох ролей користувачів.
4. Реалізувати авторизацію та автентифікацію за допомогою JWT.
5. Реалізувати можливість створення обідніх подій, формування замовлень та взаємодії між користувачами.
6. Забезпечити ведення статистики та фільтрації даних.
7. Задокументувати API з використанням Swagger (OpenAPI).

Таким чином, результатом виконання роботи має стати функціональний вебсервіс, готовий до використання як частина внутрішньокорпоративної інфраструктури або як SaaS-рішення.

2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Для побудови вебсервісу корпоративного харчування було розроблено логічну модель бази даних у вигляді ER-діаграми (Entity-Relationship), яка відображає основні сутності системи, їх атрибути та взаємозв’язки.

Згідно з принципами нормалізації, модель повинна відповідати третій нормальній формі (ЗНФ), що забезпечує відсутність надлишкових залежностей та збереження цілісності даних.

У межах даного проєкту побудована модель є реляційною, всі сутності мають чітко визначені ключі, а всі зв’язки реалізуються засобами зовнішніх ключів (Foreign Key), що підтверджує відповідність моделі нормалізованій структурі.

Таблиця 2.1

Таблиця основних сутностей системи та їх призначення

Сутність	**Ключові атрибути**	**Зв’язки**	**Функції**
auth_user	username, email, password, role, company_id	Пов’язана 1:1 з companies_company або providers_provider; 1:M з orders_order, joinrequest	Основна таблиця користувачів. Зберігає логін, роль, до якої компанії належить, та іншу особисту інформацію.

Таблиця 2.1 (продовження)

companies_company	user_id, description, limit_one_lunch, address	1:M з joinrequest, lunchevent, companyprovider; 1:1 з auth_user	Профіль компанії. Організовує заходи харчування, встановлює обмеження на обід.
providers_provider	user_id, description, logo, address	1:M з dishcategory, companyprovider; 1:1 з auth_user	Профіль постачальника. Додає страви та меню, співпрацює з компаніями.
companies_joinrequest	client_id, company_id, is_approved, created_at	Багато клієнтів → одна компанія	Запити клієнтів на приєднання до компанії. Затверджуються адміністрацією компанії.
companies_companyprovider	company_id, provider_id, is_confirmed	Багато компаній ↔ багато постачальників	Представляє співпрацю між компанією та постачальником. Може бути підтверджена або ні.

Таблиця 2.1 (продовження)

companies_lunchevent	company_id, date, is_confirmed, created_at	1:M з orders_order	Обідній захід компанії на певну дату. Кожен працівник компанії може оформити замовлення.
orders_order	client_id, event_id, created_at	1:M з orders_orderitem	Замовлення клієнта на конкретний захід. Містить декілька страв (OrderItem).
orders_orderitem	order_id, dish_id, quantity, status	Багато страв → одне замовлення, одна страва → багато замовлень	Одна позиція замовлення: конкретна страва, її кількість і статус (прийнято, пакується, доставлено тощо).
providers_dishcategory	name, provider_id, parent_id	1:M з providers_dish, може мати підкатегорії	Категорії страв (наприклад, "Супи", "Салати"). Підтримує вкладеність (категорія в категорії).

Таблиця 2.1 (закінчення)

providers_dish	name, description, price, category_id	Багато страв у одній категорії	Конкретна страва з ціною, описом та фото. Використовується в замовленнях клієнтів.

Як видно з таблиці 2.1, система містить сутності для моделювання всіх ключових аспектів бізнес-логіки: користувачів, компаній, замовлень, обідів та постачальників.

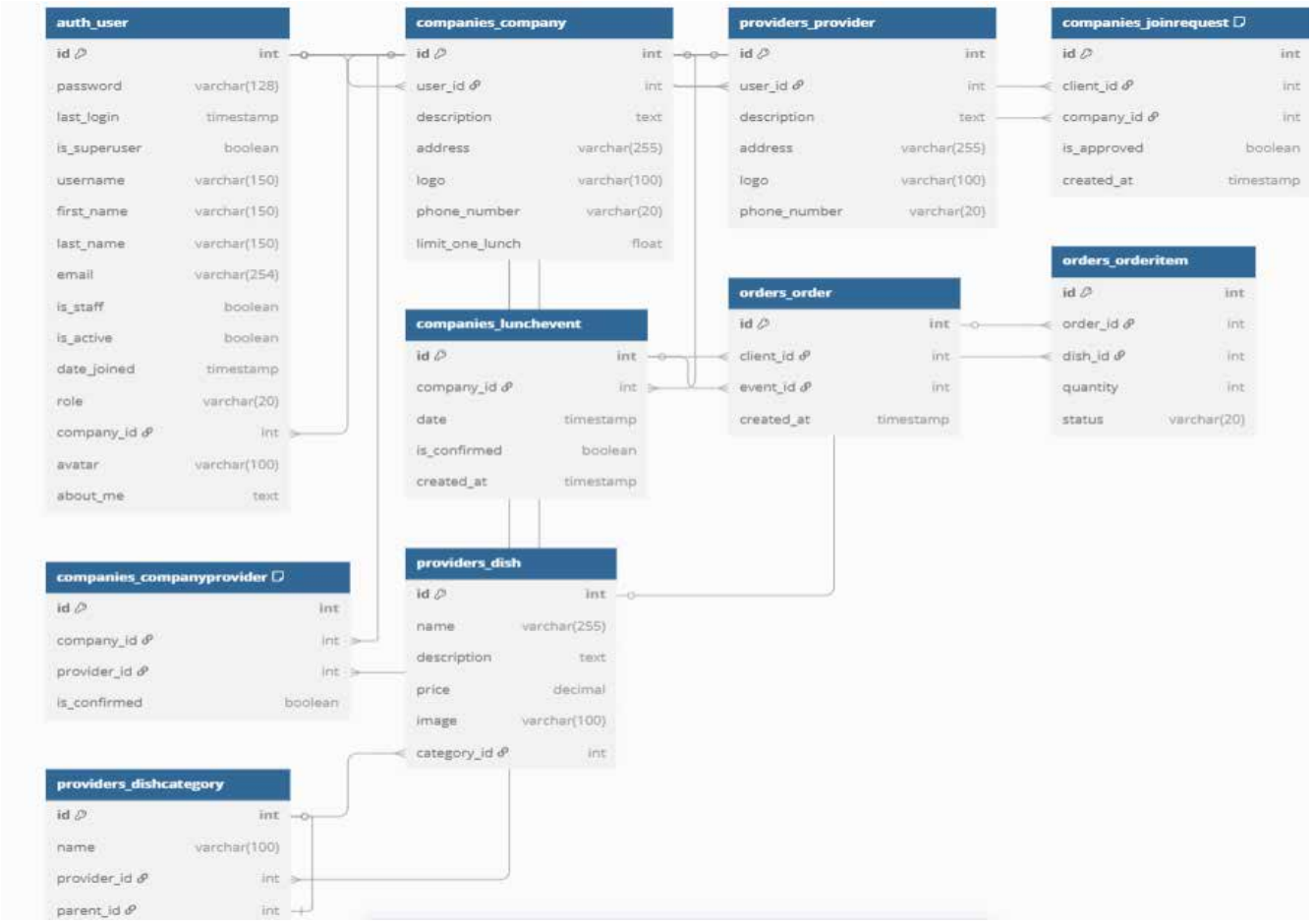

Рис. 2.1 - ER - діаграма бази даних web-сервісу

На основі побудованої логічної моделі даних була розроблена діаграма класів, яка відображає основні сутності системи як об’єкти з атрибутами та зв’язками. Класи в об’єктно-орієнтованому підході відповідають сутностям у реляційній моделі, а зв’язки між ними реалізують асоціації.
Це дозволяє перейти від моделювання структури даних до побудови архітектури прикладного програмного забезпечення з використанням принципів об’єктно-орієнтованого проєктування.

2.2 Діаграма класів та кооперацій

Діаграма класів є основним інструментом для об'єктно-орієнтованого моделювання. Вона відображає основні сутності системи (класи), їхні атрибути, методи (опціонально) та зв’язки між ними.

У вебсервісі організації корпоративного харчування класи відповідають основним моделям бази даних: користувачам, компаніям, постачальникам, замовленням, обідами тощо. Взаємозв’язки між ними відображають логіку системи:

- користувач може належати до компанії;
- компанія може створювати події (обіди);
- постачальники мають меню, яке компанії можуть обирати;
- клієнти формують замовлення на основі подій.

Діаграма дозволяє візуально проаналізувати структуру системи, визначити залежності, а також перевірити відповідність моделі реальним бізнес-процесам.

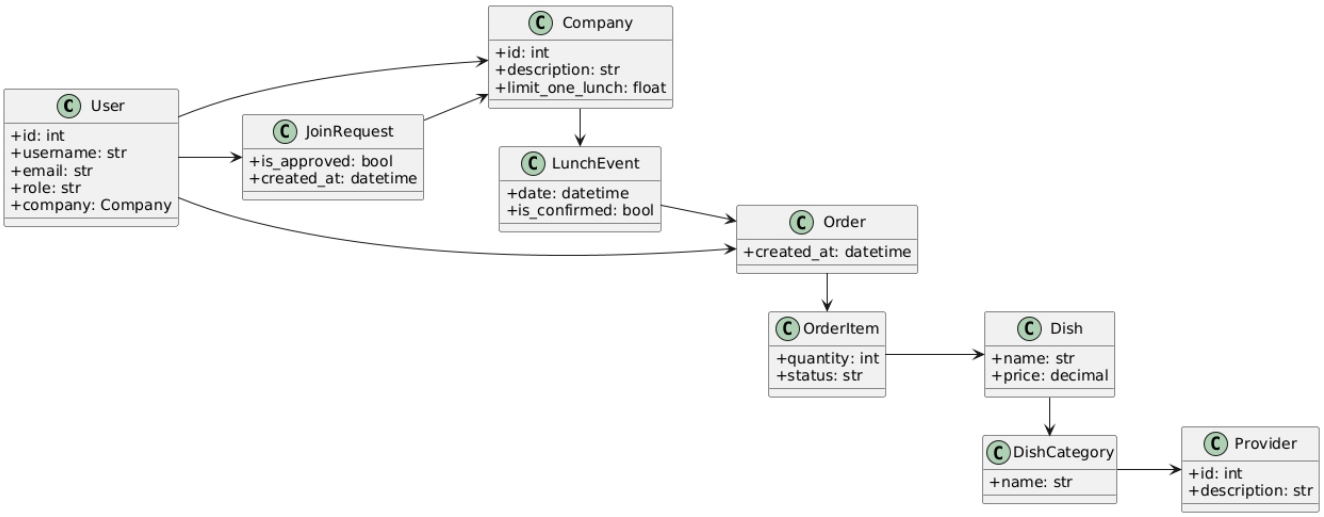

Рис. 2.2 – Діаграма класів та кооперацій системи.

2.3 Діаграма пакетів

Діаграма пакетів демонструє логічну структуру проєкту з точки зору розподілу на функціональні модулі (пакети). Вона використовується для представлення архітектури програмного забезпечення на більш високому рівні абстракції.

У межах Django-проєкту для корпоративного харчування можна виокремити такі основні пакети:

- users – логіка реєстрації, автентифікації та керування ролями користувачів;
- companies – компанії, заявки на приєднання, організація обідів;
- providers – постачальники та їхнє меню;
- orders – логіка замовлень та позицій замовлення;
- authentication – обробка JWT-токенів;
- api – інтеграція з фронтендом, обробка запитів.

Діаграма показує, як ці пакети взаємодіють між собою, які між ними існують залежності, що дозволяє зрозуміти структуру та межі відповідальності кожного модуля.

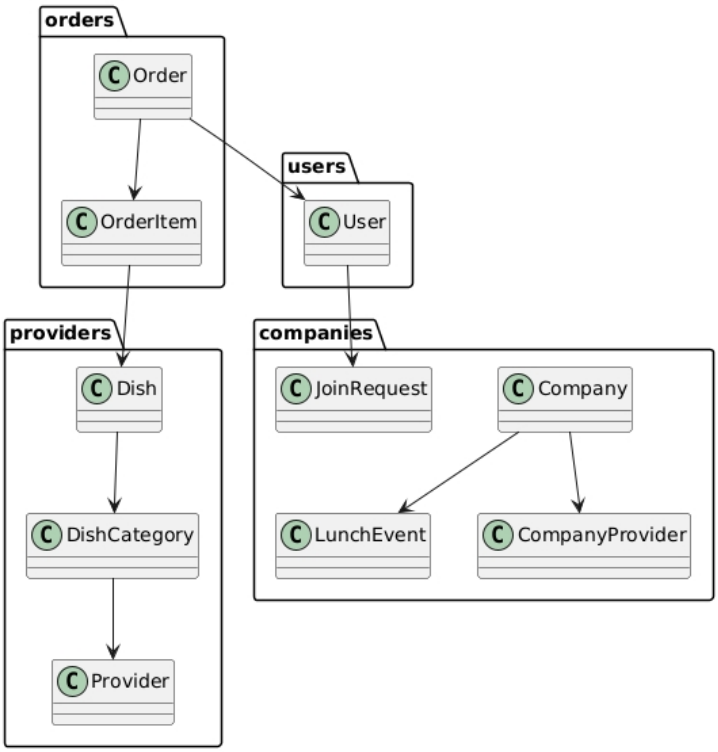

Рис. 2.3 – Діаграма пакетів програмного забезпечення.

2.4 Діаграма компонентів

Діаграма компонентів відображає фізичну архітектуру системи – з яких частин вона складається на рівні виконуваних модулів (компонентів) та як ці компоненти взаємодіють між собою.

У даному проєкті виділено три основні компоненти:

- Клієнтська частина (React SPA) – відповідає за інтерфейс користувача, надсилає запити до серверного API.
- Серверна частина (Django REST Framework) – обробляє запити, керує логікою бізнес-процесів, виконує авторизацію, обробку замовлень.

- База даних (PostgreSQL) – зберігає всю інформацію: користувачів, компанії, страви, замовлення, події тощо.

Кожен компонент має чітко визначені інтерфейси. Клієнт звертається до REST API через HTTP. API звертається до бази даних через ORM. Це забезпечує розділення відповідальностей та гнучкість масштабування.

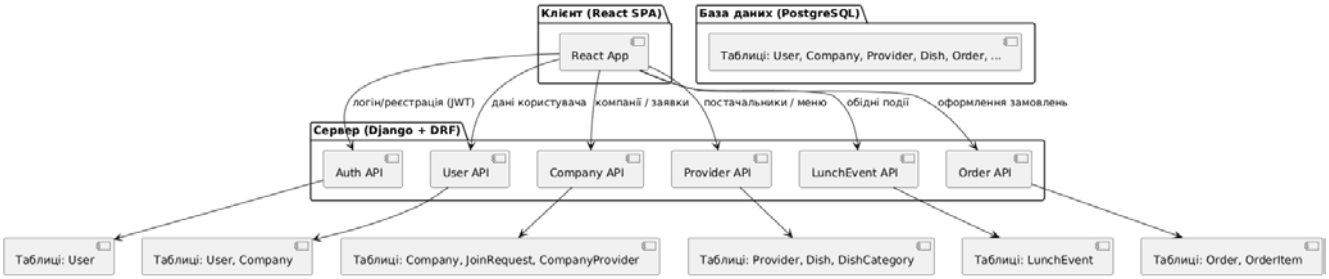

Рис. 2.4 – Діаграма компонентів системи.

3. РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

У рамках розробки вебсервісу організації корпоративного харчування було обрано систему управління базами даних PostgreSQL. Це потужна об’єктно-реляційна СУБД з відкритим кодом, яка широко використовується для розробки масштабованих вебзастосунків.

Основні переваги PostgreSQL:

- підтримка складних типів даних (JSONB, масиви, геодані);
- транзакційність та надійність зберігання даних;
- можливість масштабування та оптимізації запитів;
- повна сумісність з Django ORM.

Завдяки інтеграції PostgreSQL з Django ORM (Object-Relational Mapping) було забезпечено високий рівень абстракції при роботі з базою даних, що прискорило розробку і зменшило кількість помилок у SQL-запитах.

3.2 Розробка інформаційної бази

Розробка інформаційної бази здійснювалась на основі логічної моделі, представленої у вигляді ER-діаграми (див. розділ 2.1). Кожна сутність була реалізована у вигляді Django-моделі (класа), яка автоматично трансформується у відповідну таблицю під час виконання команди makemigrations та migrate.

Основні етапи створення бази даних:

1. Проєктування моделей: для кожної сутності створено окремий Python-клас з полями, які відповідають атрибутам.

2. Налаштування зв’язків: використано ForeignKey, OneToOneField та ManyToManyField для реалізації відповідних зв’язків між таблицями.
3. Автоматична генерація схем: Django на основі моделей створює SQL-команди для створення таблиць і зв’язків.
4. Ініціалізація бази даних: шляхом виконання python manage.py migrate створено фізичну структуру бази у PostgreSQL.

Приклад реалізації моделі:

```
class Company(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    description = models.TextField(blank=True)
    address = models.CharField(max_length=255)
    limit_one_lunch = models.FloatField(null=True, blank=True)
```

Цей клас відповідає таблиці companies_company, де кожна компанія прив’язана до окремого користувача (OneToOneField), має опис, адресу та ліміт на обід.

Також у проєкті реалізовано початкове заповнення бази (seed-дані) та підключено адміністративну панель для перегляду вмісту таблиць у середовищі Django Admin.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Для реалізації вебсервісу організації корпоративного харчування було обрано сучасний та ефективний стек технологій, який забезпечує швидку розробку, масштабованість, зручну підтримку та розширення проєкту. Розробка

велась за архітектурою "клієнт-сервер", де фронтенд і бекенд розділені на окремі логічні частини.

Таблиця 3.1

Таблиця компонентів бекенду

Компонент	Інструмент / Технологія	Причина вибору
Мова програмування	Python	Зручний синтаксис, велика кількість бібліотек
Вебфреймворк	Django + Django REST Framework	Швидка розробка, інтеграція з ORM, підтримка REST API
СУБД	PostgreSQL	Потужна реляційна СУБД, добре працює з Django
Авторизація	JWT (JSON Web Tokens)	Безпечна та масштабована авторизація
Документація API	Swagger (drf-yasg)	Автоматична генерація інтерактивної документації API
Розробка БД	Django ORM + міграції	Автоматичне створення SQL, просте внесення змін

Таблиця 3.2

Таблиця компонентів фронтенду

Компонент	Інструмент / Технологія	Причина вибору
Мова/фреймворк	JavaScript / React	Побудова SPA, зручна компонентна структура
Стан керування	React Context	Керування глобальним станом користувача
Запити до API	Axios	Асинхронна комунікація з бекендом
UI-бібліотека	Bootstrap / Material UI	Швидке створення адаптивного інтерфейсу

Завдяки цьому інструментарію було забезпечено:

- чіткий поділ обов’язків між частинами системи;
- масштабованість у майбутньому;
- можливість повторного використання компонентів;
- швидке прототипування та тестування.

3.4 Алгоритмізація та програмування програмних модулів

Алгоритмізація програмних модулів – це побудова блок-схем, які відображають ключову бізнес-логіку системи. Вони дають змогу описати порядок дій для реалізації основних процесів: автентифікації, створення замовлень, організації обідів тощо. Блок-схеми виступають етапом проміжного проєктування

між логічною структурою бази даних і безпосередньою реалізацією програмного коду.

Програмування програмних модулів – це реалізація цих алгоритмів за допомогою відповідної мови програмування та фреймворків. У рамках створення вебсервісу організації корпоративного харчування було використано Python з Django REST Framework для серверної частини та React для клієнтської.

Реєстрація клієнта

Рис. 3.1 Блок-схема модуля реєстрація клієнта

Фрагмент реалізації:

```
class RegisterClientAPIView(APIView):
    def post(self, request):
        serializer = RegisterClientSerializer(data=request.data)
```

```
        if serializer.is_valid():
            serializer.save()
            return Response({'detail': 'Client registered.'})
        return Response(serializer.errors, status=400)
```

Створення обіднього заходу компанією

Рис. 3.2 Блок-схема модуля створення обіднього заходу компанією

Фрагмент реалізації:

```
def perform_create(self, serializer):
    company = get_object_or_404(Company, user=self.request.user)
    event = serializer.save(company=company)
    clients = JoinRequest.objects.filter(company=company,
is_approved=True).values_list('client_id', flat=True)
    for client_id in clients:
```

Order.objects.create(client_id=client_id, event=event)

Зміна статусу позицій замовлення постачальником

Рис. 3.3 Блок-схема модуля створення обіднього заходу компанією

Фрагмент коду:

```
class OrderStatusUpdateView(APIView):
    def patch(self, request, pk):
        order = get_object_or_404(Order, pk=pk,
items__dish__category__provider__user=request.user)
        new_status = request.data.get("status")
        for item in order.items.all():
            item.status = new_status
            item.save()
```

```
return Response({"detail": "Status updated"})
```

Розроблені програмні модулі реалізують основні бізнес-процеси системи: реєстрацію, організацію обідів, оформлення замовлень і взаємодію з постачальниками. Алгоритми, представлені у вигляді блок-схем, дозволяють наочно побачити порядок обробки даних. Їх реалізація в коді на Python забезпечує зручність, розширюваність і відповідність сучасним практикам REST-архітектури.

4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

Впровадження програмної системи передбачає комплекс заходів, спрямованих на перевірку її працездатності, оцінку ресурсних вимог, підготовку середовища для роботи, встановлення компонентів, а також подальшу експлуатацію. У цьому розділі розглянуто етапи тестування, побудову діаграми розміщення, визначено апаратні та програмні вимоги, а також склад інсталяційного пакету.

4.1 Тестування системи

Тестування — це процес виявлення помилок у програмному забезпеченні шляхом виконання коду в різних умовах. Метою є перевірка відповідності системи вимогам користувача, стабільності та надійності виконання.

Обраний тип тестування: функціональне ручне тестування (manual functional testing)

Цей тип тестування дозволяє вручну перевірити правильність виконання бізнес-логіки через взаємодію з API за допомогою інтерфейсу на React.

Перше що бачить користувач це вхід в систему (рис 4.1)

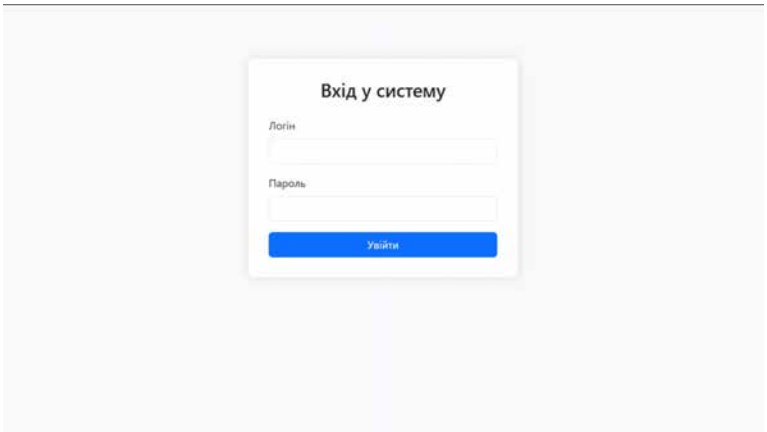

Рис. 4.1. - Вхід в систему

В залежності від ролі: client, company, provider відображається інтерфейс.

Почнемо з користувача із ролю client.

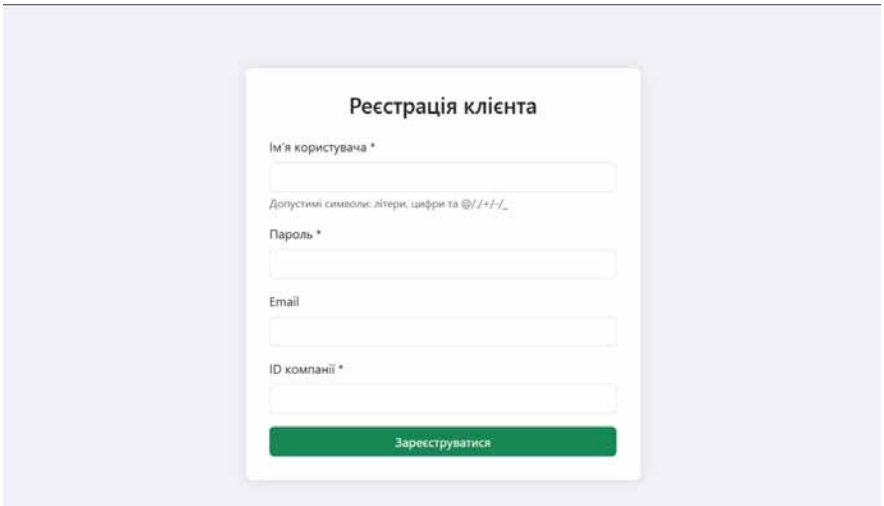

Рис. 4.2 - Реєстрація користувача із ролю client

Після авторизаіції користувач бачить перед собою події харчування(рис 4.3), які вже сторив адміністратор компанії в якій client працює (рис 4.5)

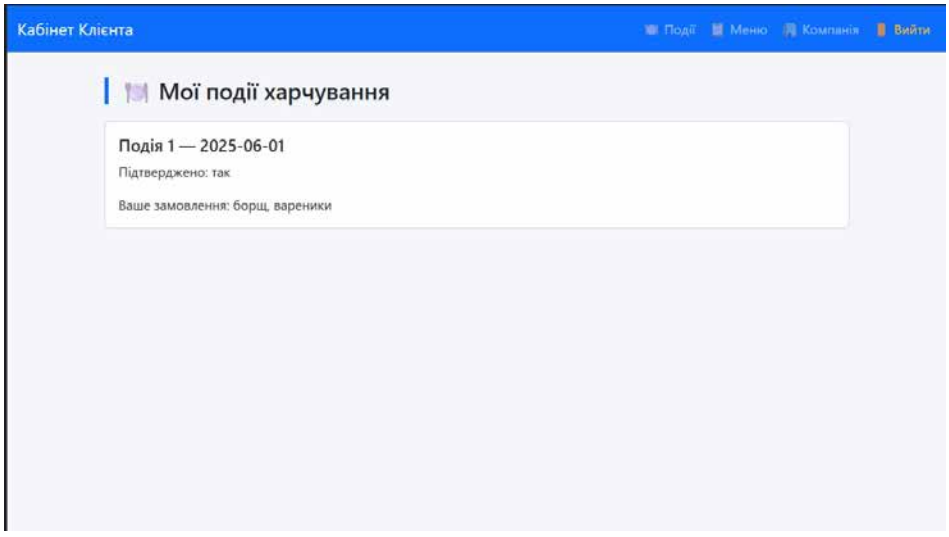

Рис. 4.3 - Події харчування користувача із ролю client

Також користувач може вибирати страви із меню доступного для його компанії (рис. 4.4).

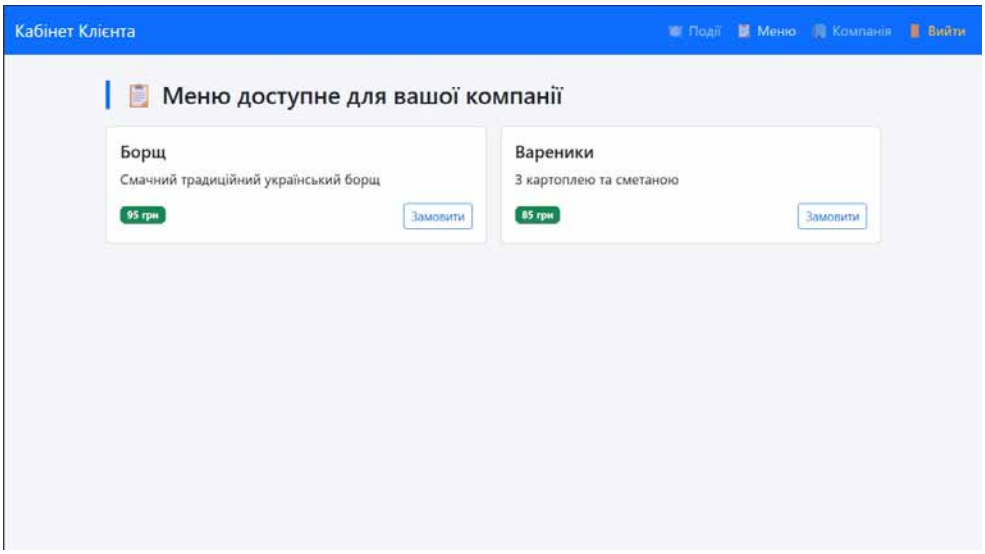

Рис. 4.4 - Доступне меню компанії

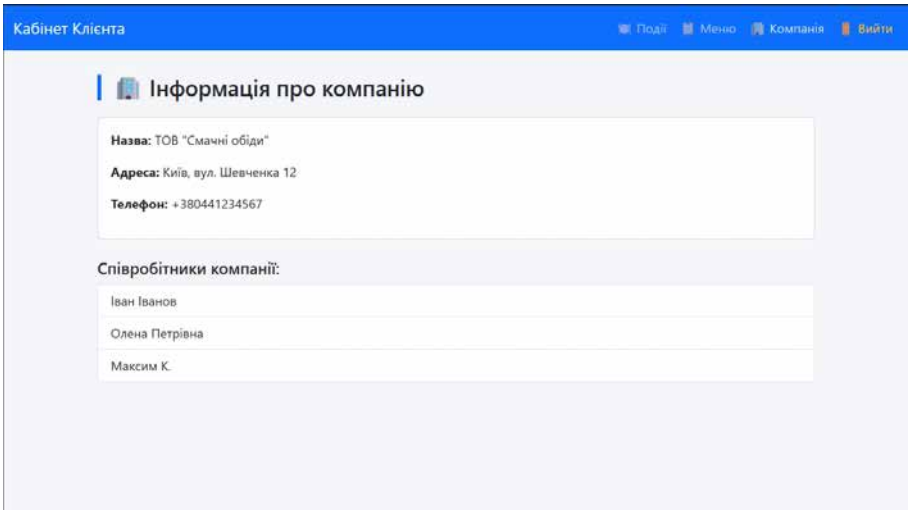

Рис. 4.5 - Загальні відомості про користувача

Протестуємо систему з ролі company

Компанії доступний функціонал створення та адміністрування подій харчування (рис.4.6), адміністрування співробіників (рис.4.7) та провайдерів (рис.4.8).

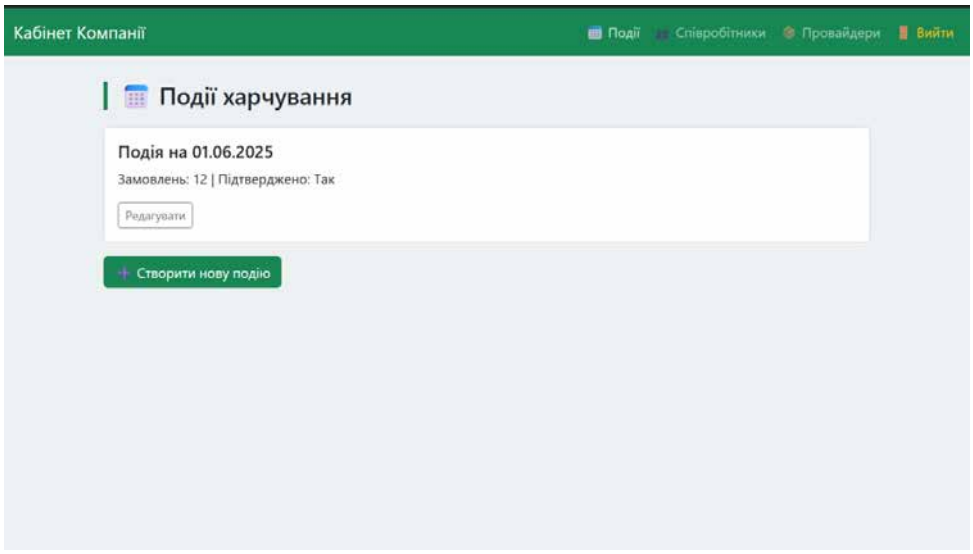

Рис. 4.6 - Наявні події харчування компанії

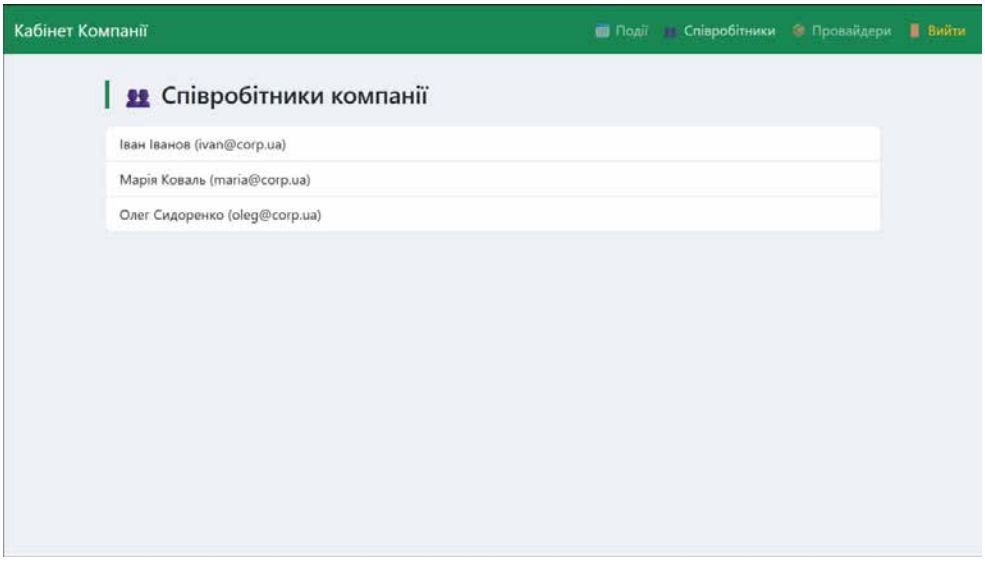

Рис. 4.7 - Інтерфейс адміністрування співробіників компанії

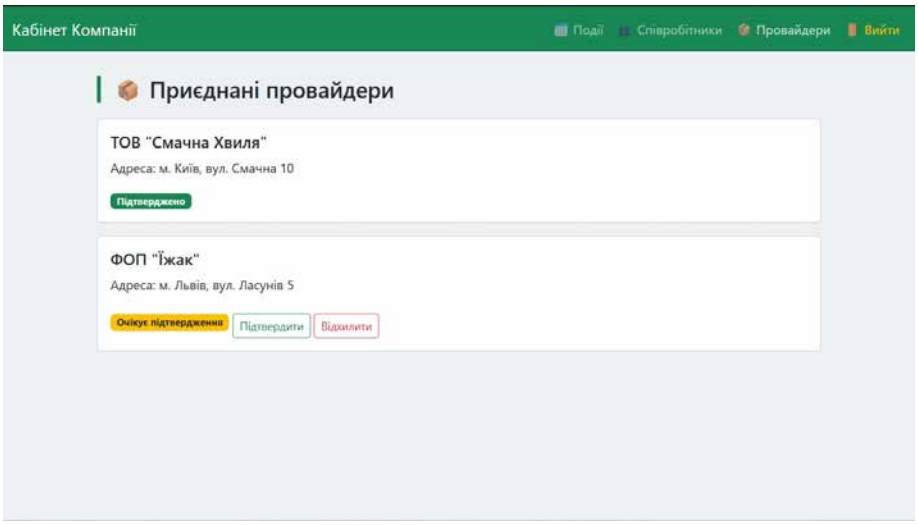

Рис. 4.8 - Інтерфейс адміністрування провайдерів

Закінчимо тестування **перевіркою системи з ролею provider.**

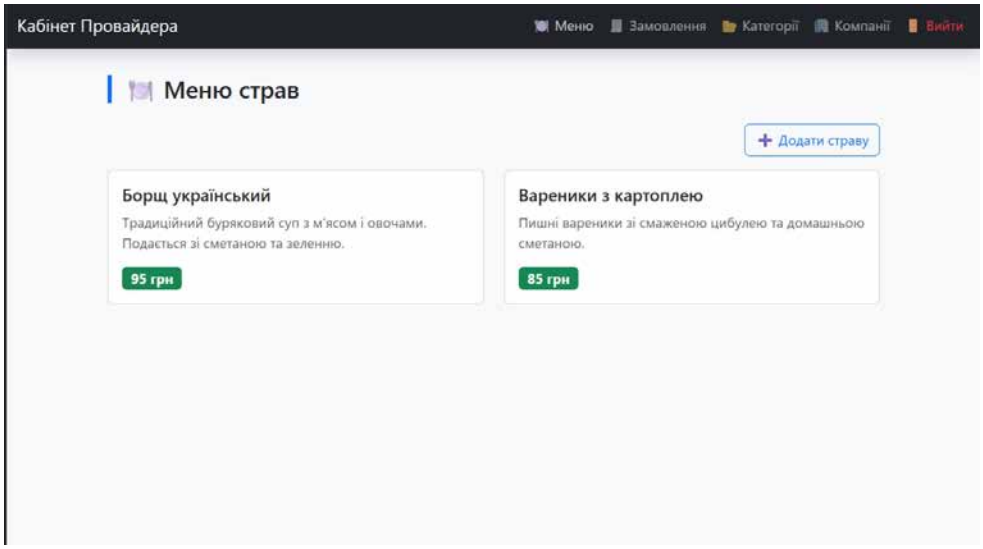

Рис. 4.9 - Вкладка із меню провайдера

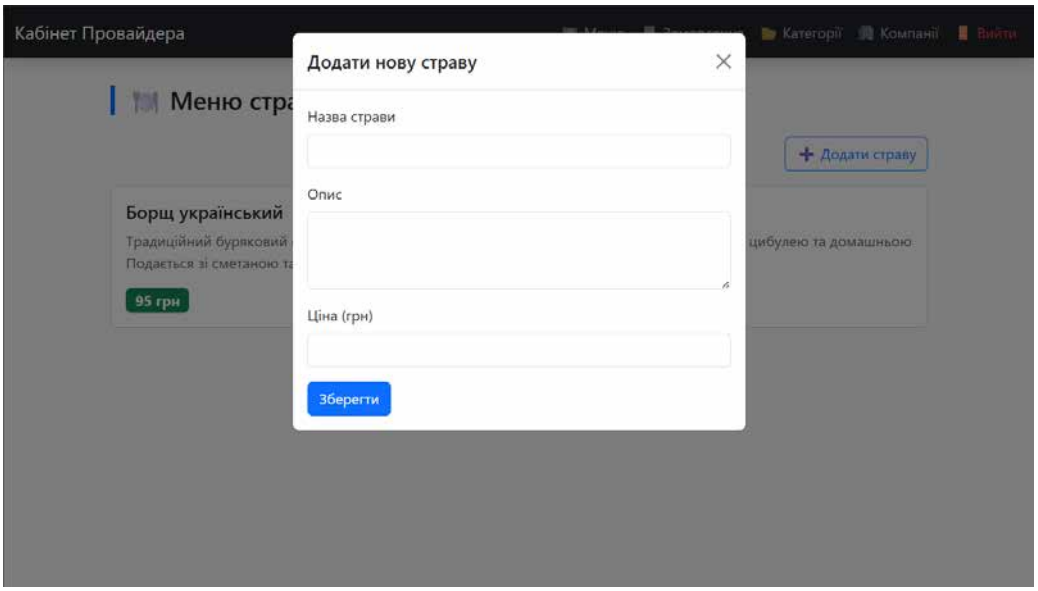

Рис. 4.10 - Добавлення найменування в меню

Після авторизації ми можемо адмініструвати меню(рис. 4.9), замовлення (рис. 4.11), категорії(рис. 4.12) та компанії(рис. 4.13).

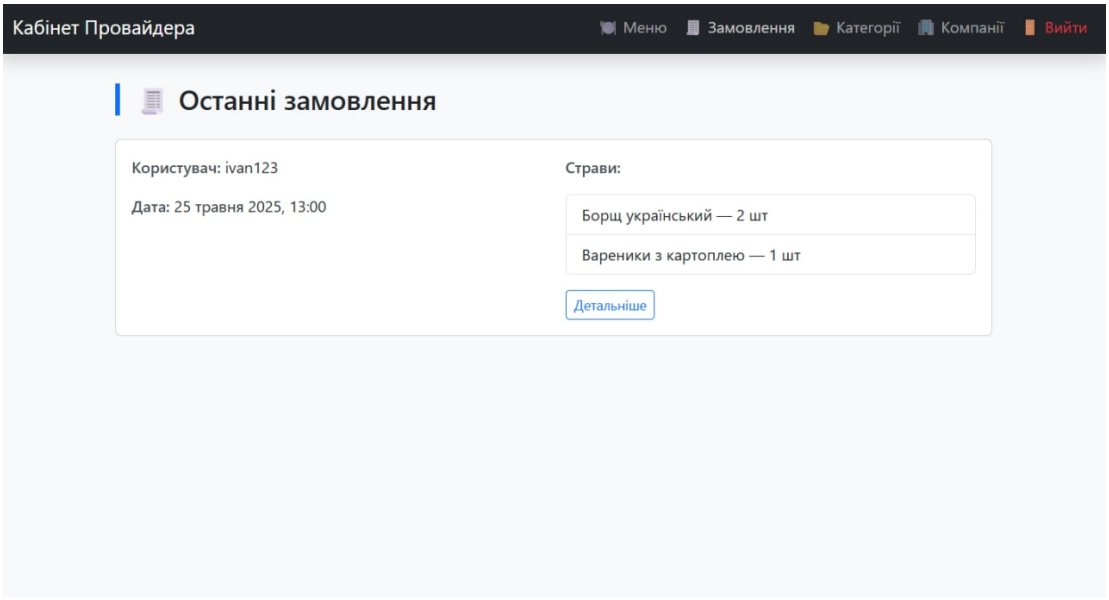

Рис. 4.11 - Інтерфейс адміністрування замовлень

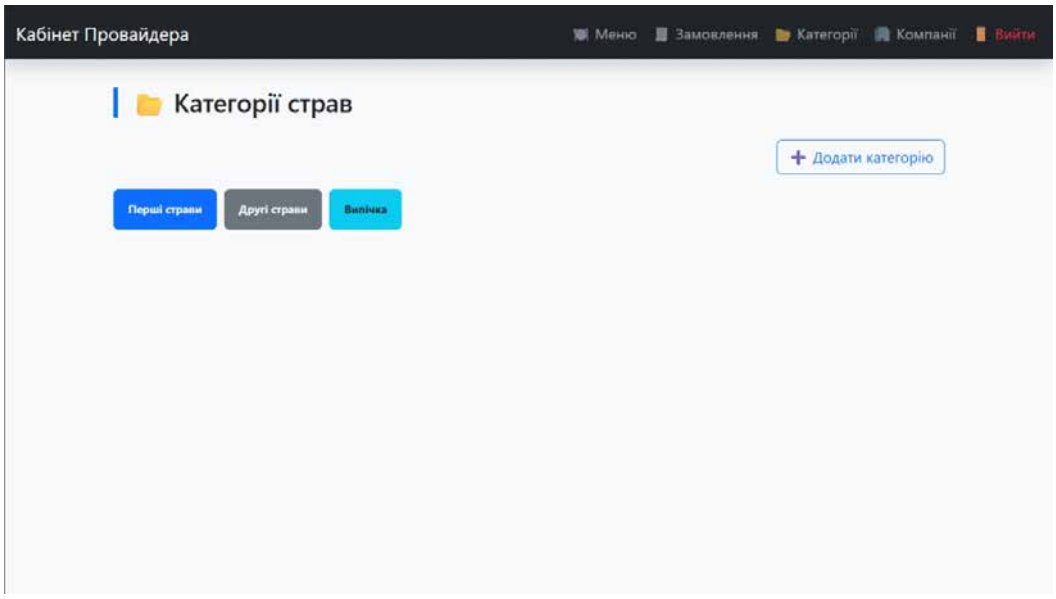

Рис. 4.12 - Інтерфейс категорій страв провайдера

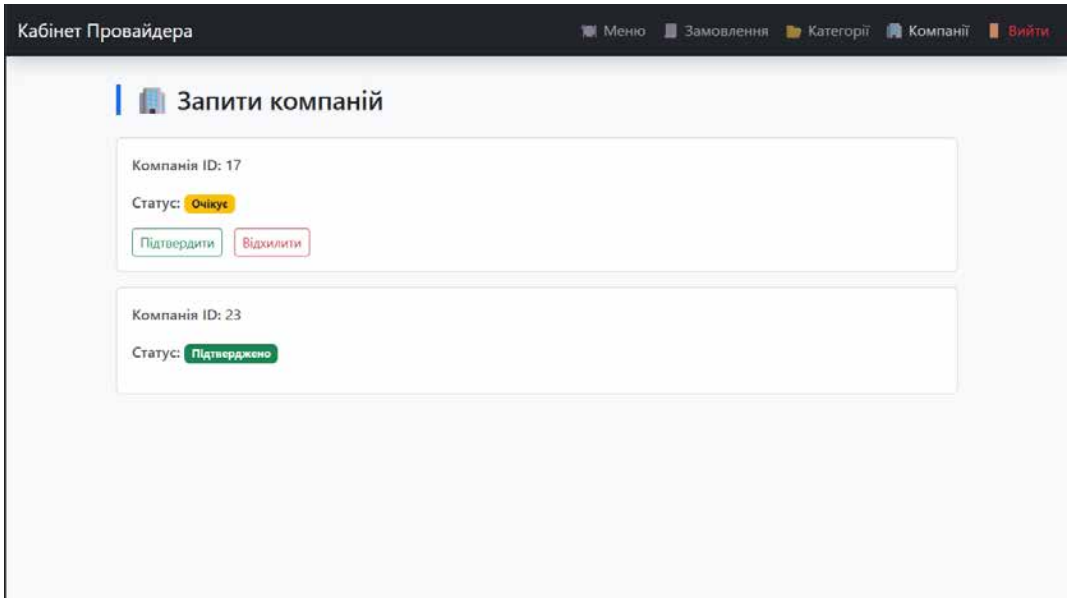

Рис. 4.13 - Адміністрування запитів компаній

Результат: всі функціональні сценарії відпрацьовують коректно. Система успішно обробляє запити, перевіряє права доступу, генерує відповіді у відповідному форматі. API має документацію у форматі Swagger, що дозволяє легко відтворити всі запити вручну.

4.2 Вимоги до апаратного та програмного забезпечення

Перед впровадженням інформаційної системи необхідно визначити мінімальні та рекомендовані вимоги до апаратного і програмного забезпечення для її стабільного функціонування. Враховуючи архітектуру вебсервісу (клієнт-серверна модель з використанням REST API), система може бути встановлена як на локальний сервер, так і в хмарне середовище (наприклад, Heroku, DigitalOcean, AWS, або інший VPS).

Апаратні вимоги до сервера (рекомендовані):

- Процесор: не менше 2-х ядер (2.0+ GHz)
- ОЗП: від 4 ГБ
- Дисковий простір: від 10 ГБ
- З’єднання з мережею: стабільне (для API/React)

Програмні вимоги до сервера:

- Операційна система: Linux (Ubuntu 20.04+), Windows Server або Docker-сумісна ОС
- Python 3.10+
- PostgreSQL 13+
- Node.js 18+ (для збирання React-клієнта)
- pip/pipenv або poetry (для залежностей)
- Docker (опціонально — для спрощеного розгортання)

Вимоги до клієнтської сторони:

- Будь-який сучасний веббраузер (Chrome, Firefox, Edge)
- Підтримка JavaScript (React SPA)
- Доступ до інтернету

У разі розгортання на великій організації або в умовах великого навантаження рекомендується використовувати додаткові сервіси кешування (Redis), балансування навантаження (Nginx) та сертифікації HTTPS (Let’s Encrypt або аналоги).

4.3 Склад інсталяційного пакету

Для забезпечення простого розгортання системи розроблений повноцінний інсталяційний пакет, який включає в себе всі необхідні компоненти для запуску як у ручному режимі, так і з використанням контейнеризації (Docker).

Серверна частина (бекенд):

Повний вихідний код Django-проєкту (manage.py, конфігурація settings.py, файли моделей, views, serializers);

Файл requirements.txt з переліком усіх бібліотек та залежностей;

Файли міграцій (migrations/) для створення структури бази даних;

Власні middleware, permissions, API-документація на основі Swagger (drf-yasg).

Клієнтська частина (фронтенд):

Вихідні файли React-застосунку (src/, public/);

Список залежностей у файлі package.json;

Готова збірка у директорії build/ для розгортання на сервері або CDN;

Налаштування взаємодії з API через Axios.

База даних:

SQL-дамп (init.sql) або початковий набір тестових даних у форматі JSON (seed_data.json);

Конфігураційний файл .env з шаблоном змінних середовища (логіни, паролі, URI до БД тощо).

Сценарії запуску:

Bash-скрипти або інструкція ручного встановлення;

Dockerfile та docker-compose.yml — для автоматизованого підняття серверу, бази даних та клієнта в контейнерах.

Супровідна документація:

README.md з короткою інструкцією;

INSTALL.md — детальна покрокова інструкція з локального запуску або розгортання на сервері;

Swagger UI для інтерактивної документації REST API

У цьому розділі було описано процес тестування системи, сформовано апаратні та програмні вимоги відповідно до архітектури, а також наведено чіткий склад інсталяційного пакету. Це створює всі необхідні передумови для ефективного впровадження та подальшої експлуатації системи.

ВИСНОВКИ

У межах цієї кваліфікаційної роботи було виконано повний цикл розробки вебсервісу організації корпоративного харчування, що передбачав аналіз предметної області, проєктування логічної та фізичної структури даних, реалізацію архітектури програмної системи, програмування модулів, тестування та підготовку до впровадження.

На першому етапі було здійснено системний аналіз предметної області. Визначено основні користувацькі ролі: клієнт, компанія та постачальник, описано функціональні вимоги до кожної з них. На основі цих вимог сформовано перелік сутностей, побудовано ER-діаграму та доведено відповідність моделі нормалізованим формам (зокрема, третій нормальній формі).

На етапі проєктування програмного забезпечення створено діаграми класів, кооперацій, компонентів і розміщення, що дозволили структурувати систему, визначити межі відповідальності модулів та способи їхньої взаємодії. Для забезпечення зручності й масштабованості була обрана клієнт-серверна архітектура: серверна частина реалізована на Python + Django REST Framework, а клієнтська — на React.js.

У рамках розробки програмної системи реалізовано:

- реєстрацію користувачів за ролями (клієнт, компанія, постачальник);
- механізм приєднання клієнтів до компаній через запити з підтвердженням;
- створення компаніями подій (обідів) та автоматичне формування замовлень;
- формування замовлень клієнтами з обраного меню постачальників;
- зміну статусів замовлень постачальниками на всіх етапах доставки;

- авторизацію з використанням JWT (JSON Web Token);
- REST API з повною документацією у форматі Swagger.

На етапі впровадження було проведено функціональне тестування системи з використанням Swagger UI, що дозволило перевірити відповідність програмної логіки заявленим сценаріям. Розроблено інсталяційний пакет з підтримкою Docker для швидкого розгортання системи на сервері.

Таким чином, поставлена мета роботи — створення сучасного вебсервісу для організації корпоративного харчування з урахуванням ролей користувачів, бізнес-логіки замовлень і сучасних технологій — досягнута повністю. Система є функціонально завершеною, масштабованою, придатною до використання в умовах реального підприємства або закладу.

Результати роботи можуть бути використані як основа для подальшого розвитку: інтеграції з платіжними сервісами, реалізації мобільного додатку, впровадження розширеної аналітики або модулів персоналізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойко А. І., Кравець В. С. Системи управління базами даних: навчальний посібник. — Львів: Видавництво ЛНУ, 2021.
2. Жураківський В. І. *Об'єктно-орієнтоване програмування та проектування: UML-підхід*. — Київ: ВНТУ, 2020.
3. Чалий О. В. *Розробка інформаційних систем на основі клієнт-серверної архітектури*. — Харків: НТУ "ХПІ", 2021.
4. Django Software Foundation. [Офіційна документація Django]. – https://docs.djangoproject.com/
5. Encode OSS Ltd. [Django REST Framework documentation]. – https://www.django-rest-framework.org/
6. Facebook Inc. [**React Documentation**]. – https://reactjs.org/
7. PostgreSQL Global Development Group. [PostgreSQL Documentation]. – https://www.postgresql.org/docs/
8. RFC 7519. *JSON Web Token (JWT)*. – Internet Engineering Task Force. – https://datatracker.ietf.org/doc/html/rfc7519
9. Sommerville I. *Software Engineering* (10th Edition). – Pearson Education, 2015.

ДОДАТОК А

SQL ЗАПИТИ

Сторінок – 6

2025

```
-- Таблиця користувачів (auth_user)
CREATE TABLE auth_user (
    id SERIAL PRIMARY KEY,
    username VARCHAR(150) UNIQUE NOT NULL,
    email VARCHAR(254) NOT NULL,
    password VARCHAR(128) NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('client', 'company', 'provider')),
    company_id INTEGER REFERENCES companies_company(id),
    avatar VARCHAR(255),
    about_me TEXT
);
-- Таблиця компаній (companies_company)
CREATE TABLE companies_company (
    id SERIAL PRIMARY KEY,
    user_id INTEGER UNIQUE REFERENCES auth_user(id),
    description TEXT,
    address VARCHAR(255),
    logo VARCHAR(255),
    phone_number VARCHAR(20),
    limit_one_lunch FLOAT
);
-- Таблиця постачальників (providers_provider)
CREATE TABLE providers_provider (
```

```
    id SERIAL PRIMARY KEY,
    user_id INTEGER UNIQUE REFERENCES auth_user(id),
    description TEXT,
    address VARCHAR(255),
    logo VARCHAR(255),
    phone_number VARCHAR(20)
);
-- Заявка клієнта на приєднання до компанії (companies_joinrequest)
CREATE TABLE companies_joinrequest (
    id SERIAL PRIMARY KEY,
    client_id INTEGER REFERENCES auth_user(id),
    company_id INTEGER REFERENCES companies_company(id),
    is_approved BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (client_id, company_id)
);
-- Зв'язок між компанією та постачальником (companies_companyprovider)
CREATE TABLE companies_companyprovider (
    id SERIAL PRIMARY KEY,
    company_id INTEGER REFERENCES companies_company(id),
    provider_id INTEGER REFERENCES providers_provider(id),
    is_confirmed BOOLEAN,
    UNIQUE (company_id, provider_id)
);
```

```
-- Події харчування (companies_lunchevent)
CREATE TABLE companies_lunchevent (
    id SERIAL PRIMARY KEY,
    company_id INTEGER REFERENCES companies_company(id),
    date TIMESTAMP NOT NULL,
    is_confirmed BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- Замовлення (orders_order)
CREATE TABLE orders_order (
    id SERIAL PRIMARY KEY,
    client_id INTEGER REFERENCES auth_user(id),
    event_id INTEGER REFERENCES companies_lunchevent(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- Пункти замовлень (orders_orderitem)
CREATE TABLE orders_orderitem (
    id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES orders_order(id),
    dish_id INTEGER REFERENCES providers_dish(id),
    quantity INTEGER DEFAULT 1,
    status VARCHAR(20) DEFAULT 'pending'
);
-- Категорії страв (providers_dishcategory)
```

```
CREATE TABLE providers_dishcategory (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    provider_id INTEGER REFERENCES providers_provider(id),
    parent_id INTEGER REFERENCES providers_dishcategory(id)
);
-- Страви (providers_dish)
CREATE TABLE providers_dish (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    description TEXT,
    price NUMERIC(8, 2),
    image VARCHAR(255),
    category_id INTEGER REFERENCES providers_dishcategory(id)
);
```

ДОДАТОК Б

ДІАГРАМИ АКТИВНОСТІ ТА ПОСЛІДОВНОСТІ

Сторінок – 2

2025

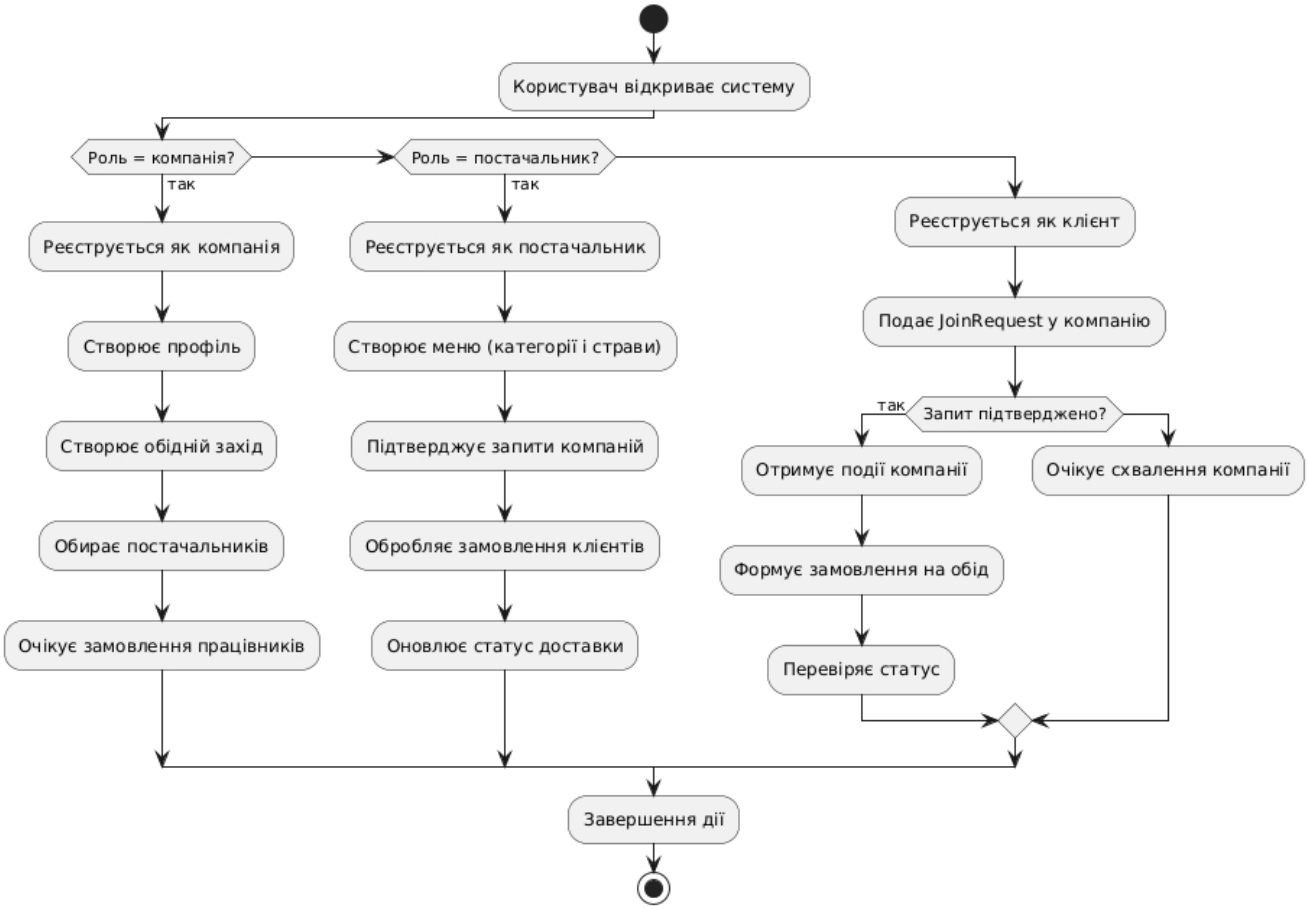

Рис. 1 - Діаграма активності

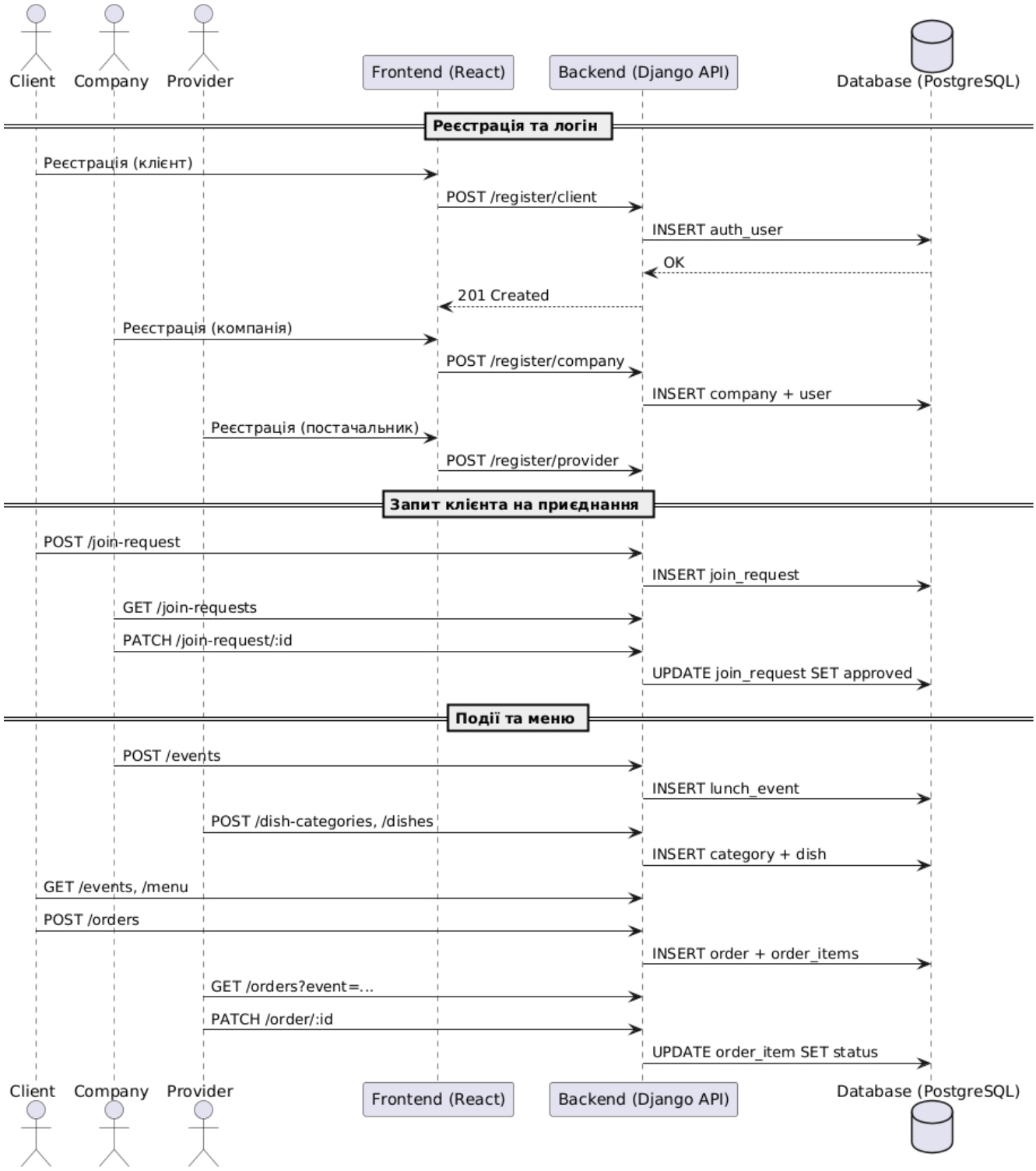

Рис. 2 -Діаграма послідовності

ДОДАТОК В

ПРОГРАМНИЙ КОД MODELS

Сторінок – 2

2025

```
from rest_framework import serializers
from .models import Company, JoinRequest, LunchEvent, CompanyProvider
from providers.models import Provider
from orders.serializers import OrderSerializer

class LunchEventSerializer(serializers.ModelSerializer):
    orders = OrderSerializer(many=True, read_only=True)

    class Meta:
        model = LunchEvent
        fields = ('id', 'company', 'date', 'is_confirmed', 'created_at', 'orders')
        read_only_fields = ('company', 'is_confirmed', 'created_at', 'orders')

class CompanyToProviderSerializer(serializers.ModelSerializer):
    class Meta:
        model = CompanyProvider
        fields = ('id', 'provider', 'is_confirmed')
        read_only_fields = ('is_confirmed',)

class JoinRequestApproveSerializer(serializers.ModelSerializer):
    class Meta:
        model = JoinRequest
        fields = ('id', 'client', 'company', 'is_approved')
        read_only_fields = ('client', 'company')

class MonthlyStatsSerializer(serializers.Serializer):
    month = serializers.CharField()
    total_orders = serializers.IntegerField()
    total_amount = serializers.DecimalField(max_digits=10, decimal_places=2)

class ProviderSerializer(serializers.ModelSerializer):
    class Meta:
        model = Provider
        fields = '__all__'
```

```
from django.db import models
from users.models import User

class Provider(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)

    description = models.TextField(blank=True)
    address = models.CharField(max_length=255, blank=True)
    logo = models.ImageField(upload_to='provider_logos/', null=True, blank=True)
    phone_number = models.CharField(max_length=20, blank=True)

    def __str__(self):
        return self.user.username

class DishCategory(models.Model):
    name = models.CharField(max_length=100)
    provider = models.ForeignKey(Provider, on_delete=models.CASCADE,
related_name='dish_categories')
    parent = models.ForeignKey('self', null=True, blank=True,
related_name='subcategories', on_delete=models.CASCADE)

    def __str__(self):
        return self.name

class Dish(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    price = models.DecimalField(max_digits=8, decimal_places=2)
    image = models.ImageField(null=True, blank=True)
    category = models.ForeignKey(DishCategory, null=False,
on_delete=models.CASCADE, related_name='dishes')

    def __str__(self):
        return f"{self.name} ({self.category.provider.name})"

    @property
```

```
    def provider(self):
        return self.category.provider


from django.contrib.auth.models import AbstractUser
from django.db import models

from django.contrib.auth.models import AbstractUser
from django.db import models

class User(AbstractUser):
    ROLE_CHOICES = (
        ('client', 'Client'),
        ('company', 'Company'),
        ('provider', 'Provider'),
    )
    role = models.CharField(max_length=20, choices=ROLE_CHOICES)
    company = models.ForeignKey('companies.Company', null=True, blank=True,
on_delete=models.SET_NULL, related_name='employees')
    avatar = models.ImageField(null=True, blank=True)
    about_me = models.TextField(blank=False, null=True)
    def __str__(self):
        return f"{self.username} ({self.role})"
```