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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Система управління замовленнями страв за підпискою»

Виконав:
студент IV курсу, групи КП-72
Теплицький Святослав Вікторович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н,
Рибачок Наталія Антонівна _____

Консультант з нормоконтролю:
Старший викладач кафедри ПЗКС, к.т.н,
Рибачок Наталія Антонівна _____

Рецензент:
Доцент кафедри ЕІ ФЕЛ, к.т.н. доцент,
Вунтесмері Юрій Володимирович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2020 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Теплицькому Святославу Вікторовичу

1. Тема проєкту «Система управління замовленнями страв за підпискою», керівник проєкту Рибачок Наталія Антонівна, старший викладач кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету від «25» травня 2020 р. №1181-с.

2. Термін подання студентом проєкту «18» липня 2021 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - обґрунтування вибору засобів реалізації;
   - структурно-алгоритмічна організація;
   - особливості реалізації програмного забезпечення;

5. Перелік обов'язкового графічного матеріалу:
   - алгоритм створення замовлення (креслення);
   - структура бази даних (креслення);
   - архітектура вебдодатку (плакат);
   - дерево проблем (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Рибачок Н.А., старший викладач		

7. Дата видачі завдання «30» жовтня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	15.11.2020	
2.	Розроблення та узгодження технічного завдання	28.11.2020	
3.	Розроблення структури вебдодатку	17.12.2020	
4.	Підготовка матеріалів першого розділу дипломного проєкту	29.12.2020	
5.	Розроблення дизайну сторінок та графічних елементів Розроблення структури вебдодатку	07.02.2021	
6.	Підготовка матеріалів другого розділу дипломного проєкту	21.02.2021	
7.	Програмна реалізація вебдодатку	13.03.2021	
8.	Тестування вебресурсу	19.03.2021	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2021	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	14.04.2021	
11.	Підготовка графічної частини дипломного проєкту	22.05.2021	
12.	Оформлення документації дипломного проєкту	28.05.2021	

Студент Святослав ТЕПЛИЦЬКИЙ

Керівник проєкту Наталія РИБАЧОК

АНОТАЦІЯ

Цей дипломний проєкт має на меті розроблення вебзастосунку для створення та управління замовленнями їжі “за підпискою”. Програмне рішення може бути використаний компаніями, які надають такі послуги.

Метою даного проєкту є створення вебдодатку, який надасть можливість керування підписками, враховуючи особливості бізнес-моделі та включаючи специфічний для цього бізнесу набір можливостей. Це дозволить малим компаніям, які надають такий сервіс, спростити обробку замовлень та не витрачати ресурси на налаштування універсальних інформаційних систем управління бізнесом. Клієнти таких компаній, у свою чергу, отримають можливість оформлювати підписку через вебсайт, не телефонуючи в компанію та зможуть бути ще більше впевненими що не станеться помилки з їхнім замовленням через людський фактор.

В даному проєкті оглянуто новітні технології та інструменти для створення програмних рішень, обґрунтовано використання монолітної архітектури та C# як мови програмування для серверної частини, TypeScript як мови програмування для клієнтської частини.

Під час виконання дипломного проєкту розроблено архітектуру вебзастосунку, алгоритм створення підписки на доставку їжі, можливість додавання, редагування та видалення планів харчування та розроблено дизайн користувацького інтерфейсу.

ABSTRACT

This diploma project aims to develop a web application for creating and managing food orders "by subscription".

The software solution is created as a web application that can be used by companies that provide food delivery services by subscription.

The purpose of this project is to create a web application that will provide subscription management functionality but taking into account the specifics of the business model and including business-specific functionality. This will allow small companies that provide such a service to simplify the processing of orders without spending resources on setting up universal business management information systems. Customers of such companies, in turn, will be able to subscribe through the website without calling the company and will be even more confident that there will be no error with their order due to human factors.

This project reviewed the latest technologies and tools for creating software solutions and justifies the use of monolithic architecture and C # as a programming language for the server side and TypeScript as a programming language for the client side.

This diploma project includes the architecture of the web application, the algorithm for creating a subscription to the delivery of healthy food, the procedure for adding, editing and deleting nutrition plans, and the design of the user interface.

ДП.045440-01-90. Система управління замовленнями страв за підпискою.
Відомість проєкту

Позначення	Найменування	Кіл-ть	Приміт ка
	Документація проєкту		
ДП.045440-02-91	Система управління	5	
	замовленнями страв		
	за підпискою.		
	Технічне завдання		
ДП.045440-03-81	Система управління	54	
	замовленнями страв		
	за підпискою.		
	Пояснювальна записка		
ДП.045440-04-51	Система управління	4	
	замовленнями страв		
	за підпискою.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Система управління	9	
	замовленнями страв		
	за підпискою.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Приміт ка
ДП.045440-06-99	Система управління	1	
	замовленнями страв		
	за підпискою.		
	Структура таблиць БД.		
	ERD діаграма		
ДП.045440-07-99	Система управління	1	
	замовленнями страв		
	за підпискою.		
	Алгоритм додавання нового		
	харчування. Діаграма		
	діяльності		
ДП.045440-08-98	Система управління	1	
	замовленнями страв		
	за підпискою.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

СИСТЕМА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ СТРАВ ЗА ПІДПИСКОЮ

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Наталія РИБАЧОК

Виконавець:

_____ Святослав ТЕПЛИЦЬКИЙ

2020

ЗМІСТ

1. Найменування та галузь застосування 3
2. Підстава для розроблення 3
3. Призначення розробки 3
4. Вимоги до програмного продукту 3
5. Вимоги до проєктної документації 4
6. Етапи проєктування 5
7. Порядок тестування розробки 5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Система управління замовленнями страв за підпискою**.**

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення вебзастосунку є завдання на дипломне проєктування, затверджене кафедрою ПЗКС (програмного забезпечення комп'ютерних систем) Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмне рішення призначене для використання компаніями які надають послуги з приготування їжі та її доставки до клієнтів та працюють за моделлю «їжа за підпискою».

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

1) Оновлення контенту вебсторінки в режимі реального часу;
2) можливість авторизації користувачів типу «Адміністратор» та «Клієнт»;
3) можливість вібору плану харчування (користувач типу «Клієнт»);
4) можлибість додання,оновлення та видалення страв (користувач типу «Адміністратор»);
5) можливість формування планів харчування;
6) можливість перегляду активних підписок.

Розробку виконати на платформі .NET з використанням мови програмування C# з використанням технології Angular та AJAX.

Додаткові вимоги:

1) реєстрація з підтвердженням електронної пошти;
2) перегляд страв які будуть доставлені у наступні 1-3 дні;
3) трекінг найближчої доставки страв (статус: приготування/пакування/доставка, наближений час отримання);
4) адаптивний користувацький інтерфейс;
5) використання статичного доменного імені;
6) розгортка в середовищі Microsoft Azure.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

Під час виконання дипломного проєкту має бути розробленою така документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Алгоритм додання нового плану харчування.Діаграма діяльності»;
   - «Структура бази даних. ERD-діаграма».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи 15.11.2020

Розроблення та узгодження технічного завдання 28.11.2020

Розроблення структури вебдодатку 17.12.2020

Розроблення дизайну сторінок та графічних елементів 07.02.2021

Програмна реалізація вебдодатку 13.03.2021

Тестування вебдодатку 19.03.2021

Підготовка матеріалів текстової частини проєкту 14.04.2021

Підготовка матеріалів графічної частини проєкту 22.05.2021

Оформлення технічної документації проєкту 28.05.2021

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2021 р.

СИСТЕМА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ СТРАВ ЗА ПІДПИСКОЮ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Наталія РИБАЧОК

Виконавець:

_____ Святослав ТЕПЛИЦЬКИЙ

2021

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 3
ВСТУП 4
1. ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ 5
1.1. Огляд проблеми, яка вирішується ПЗ 5
1.2. Опис вимог до розроблюваного ПЗ 6
1.3. Аналіз існуючих рішень 8
1.4. Результати проведеного аналізу 13
2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ 16
2.1. Вибір мови програмування для розроблення серверної частини 16
2.2. Вибір технології для розроблення клієнтської частини 22
2.3. Вибір СУБД 25
3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ 30
3.1. Загальний опис системи 30
3.2. Структурна організація програмного забезпечення 38
3.3. Структура БД 40
3.4. Створення нового плану харчування 42
4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 44
4.1. Реалізація розгортання та тестування вебдодатку 44
4.2. Опис користувацького інтерфейсу 45
4.3. Тестування вебзастосунку 48
4.4. Рекомендації щодо подальшого вдосконалення 50
ВИСНОВКИ 51
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 52
ДОДАТКИ 54

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення, програмний продукт.

Веббраузер – вебпереглядач, ПЗ що надає доступ до вебсайтів мережи інтеренет.

БД – база даних.

СУБД – система управління базами даних.

ООП - парадигма програмування, заснована на концепції "об'єктів", які можуть містити дані та код: дані у формі полів (часто відомих як атрибути або властивості) та код у формі процедур (часто відомих як методи) [1].

ОС – операційна система.

SQL Injection – це вразливість веб-безпеки, яка дозволяє зловмисникові втручатися в запити, які додаток робить до своєї бази даних [2].

XSS – це тип вразливості безпеки, який зазвичай зустрічається у вебдодатках. Атаки XSS дозволяють зловмисникам вставляти клієнтські скріпти в веб сторінки які переглядають інші користувачі [3].

CSRF – це тип зловмисного використання вебсайту, де несанкціоновані команди подаються від користувача, якому веб-програма довіряє.

ACID – стандартний набір властивостей, які гарантують надійну обробку транзакцій баз даних [4].

CDN – географічно розподілена групи серверів, які працюють разом, щоб забезпечити швидку доставку вмісту в Інтернеті [5].

ВСТУП

Проблема організації здорового харчування стає все більш актуальною останніми роками. Кількість клієнтів компаній зростає, як і кількість компаній, які надають цей сервіс. Все частіше настає момент, коли Excel вже не задовольняє потреби компаній та вони наважуються на використання інформаційних систем для автоматизації та зручнішої обробки замовлень.

Але на даний момент існуючі рішення не покривають всі вимоги компаній, які надають цей сервіс. Тому виникає необхідність користуватися двома чи трьома інформаційними системами та налаштовувати їх під свою бізнес-модель аби задовольнити потреби бізнесу.

Наявні в даній галузі проблеми будуть вирішені цим дипломним проєктом. Розроблений вебдодаток дозволить клієнтам легко оформлювати підписку на доставку здорової їжі та надасть зручний інтерфейс для керування підписками, доступними стравами, планами харчування і т.п.

Метою створення даного проєкту є створення вебдодатку, який автоматизує процес керування замовленнями за підпискою, та надасть компаніям, які надають цей сервіс, комплексне рішення відповідно до їх бізнес-моделі.

1. ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

1.1. Огляд проблеми, яка вирішується ПЗ

В сучасному світі у все більше і більше людей відмовилися від готування їжі власноруч, адже в багатьох з них не вистачає часу на приготування їжі чи людина не хоче витрачати час на цю рутинну роботу. У все більшої кількості сімей працюють обидва партнери, тому готувати їжу після роботи зазвичай не має ні сил, ні бажання.

Всім відомо, що здорове харчування, здоровий спосіб життя продовжують тривалість життя та значно зменшують ймовірність захворювань, а також впливають на самопочуття людини, додають енергії, покращують фізичний стан особи, витривалість та насолоду життям. Люди, що ведуть здоровий спосіб життя, рідше хворіють на рак, серцево-судинні хвороби та залишаються бадьорими навіть на старості. Також це впливає на психологічний стан особи, її продуктивність, задоволення життям.

Також все більше людей намагаються вести здоровий образ життя, невід'ємною частиною якого є здорове харчування. Але слідкування за своїм харчуванням потребує багато часу: потрібно підрахувати калорії, створити меню, яке буде сповнене багатьма вітамінами, мінералами та корисними речовинами і при цьому матиме високі смакові якості. Для цього потрібно вивчати багато інформації. Тому люди доручають цю роботу спеціалістам, які створюють здорове меню, але з урахуванням індивідуальних потреб кожного клієнта.

Також всім відома проблема сьогочасності – ожиріння. Великий відсоток людей мають надлишкову вагу. Це заважає їм почуватися комфортно в суспільстві, знаходити гідного партнера для укладання шлюбу. Такі люди є об'єктом булінгу, насмішок та героями жартів. Отож немає несподіванки у тому що багато хто з них бажає скинути вагу. Для цього їм потрібна правильна дієта. Їм можуть прописати дієту спеціалісти, але її дотримання займає багато часу. Тому вони вирішують доручити цю справу професіоналам, які готують

відповідні страви.

Багато людей сьогодні займаються спортом, чи навіть бажають набрати м’язову масу. Для цього їм потрібно спеціальне харчування з більшою кількість калорії й підвищенням вмістом білків. Вони потребують особливого меню, зробленого спеціально для набору маси. Також це важливо для професійних спортсменів та бодібілдерів, для яких підтримувати форму є критично важливим для професійного успіху. Вони тренуються багато часу та зазвичай такі тренування є фізично виснажливими, тож енергії для готування такої кількості їжі не залишається.

Компанії, які надають такий сервіс, існують вже на ринку, але вони стикаються з проблемою недостатнього пакету можливостей існуючих систем. Для того аби виділити ці необхідні, але відсутні можливості, треба розібратися в бізнес-моделі таких компаній.

Цільовими користувачами цієї розробки є компанії, що надають послуги з доставки здорової їжі за бізнес-моделлю підписки. Такий бізнес працює так: клієнт заходить на сайт компанії (або дзвонить по телефону) й оформлює підписку (для цього обирає план харчування, інтервал оплати). Плани харчування можуть відрізнятися за статтю, кількістю калорій, метою клієнта (схуднути чи набрати м’язову масу), типом їжі (більше м’ясної чи більше рибної): наприклад, план «Схуднення» 1200 кілокалорій для жінок та 1300 кілокалорій для чоловіків. Також клієнт обирає інтервал поновлення замовлення: наприклад, якщо клієнт обрав інтервал 30 днів, то під час оформлення замовлення клієнт заплатить за перші 30 днів користування сервісом, а після того, як пройде цей термін, клієнту потрібно буде оплатити наступні 30 дні і так далі, поки клієнт не скасує підписку. Після того як клієнт оформив підписку, ця інформація попадає в інформаційну систему компанії (або в Excel-документ).

1.2. Опис вимог до розроблюваного ПЗ

Бізнес процеси компаній, які займаються доставкою здорової їжі за моделлю підписки можна розкласти на дві складові.

По-перше, це бізнес доставки їжі. Ця складова схожа на бізнес доставки інтернет-замовлень з ресторану: замовлення створюється, готується, доставляється. До того ж існує багато ресторанів, а розробка повноцінного вебсайту з можливістю створювати замовлення потребує значних інвестицій. Тому досить часто ресторани використовують готові рішення, які можуть бути як у вигляді шаблонного сайту, що підлаштовується під ресторан, так і у вигляді вебкомпонента, який має можливість створення замовлення для додання на готовий інформаційний сайт ресторану. Також часто ресторани не мають можливості створити замовлення онлайн, а лише в телефонному режимі. Але як тільки кількість замовлень ресторану стає великою, потреба в можливості створення замовлень через вебсайт стає критичною. Тому вважатимемо це базовою та необхідною вимогою до системи управління.

Отже, інформаційна система повинна мати такі можливості:

- вебсайт з можливістю створення замовлення;
- перегляд активних замовлень (адміністратору);
- зміна статусу замовлення: створено, готується, доставляється (адміністратору).

По-друге, це бізнес доставки за підпискою. Ця складова схожа на бізнес підписки на паперові видання, де клієнт оформлює підписку та йому кожного дня чи тижня доставляється поштою свіжий екземпляр якогось видання чи новий випуск газети. Також невеликі видання можуть не мати власного вебсайту з можливістю оформлення підписки тому вважатимемо можливість оформлення підписки обов’язковою та необхідною вимогою.

Отже, інформаційна система повинна мати такі можливості:

- оформлення підписки;
- скасування підписки;
- перегляд активних підписників та продуктів, на які вони підписані (для доставки необхідних клієнтам замовлень).

Також, оскільки це сервіс доставки саме здорового харчування й клієнт зазвичай не знає, що йому буде доставлено, необхідними є такі можливості:

- створення та редагування планів харчування;
- додання виключень з їжі, наприклад: м'ясо (для вегетаріанців), різноманітні алергени, лактоза та інше;
- планування набору страв для кожного плану харчування.

Також додамо вимоги до базових можливостей систем управління:

- авторизація та реєстрація;
- користувацький вебінтерфейс.

Отож, проаналізувавши усі складові бізнесу компаній, які займаються доставкою здорового харчування, було сформовано вищевказані вимоги до інформаційної системи.

1.3. Аналіз існуючих рішень

В результаті аналізу було знайдено інформаційні системи, які покривають хоча б якусь частину необхідного пакету можливостей. Їх можна поділити на 2 категорії: системи управління онлайн замовленнями для ресторанів (GloriaFood, FlipDish, Poster, СБИС Presto та інші) та системи управління підписками для видань (преса, журнали, і т.п.) (SubscriptionFlow, Subscription DNA).

Також бізнеси, що працюють за моделлю підписки, інколи обирають універсальне рішення (наприклад Microsoft Dynamics CRM чи продукти 1С) та налаштовують його під себе. В даному аналізі не будемо враховувати універсальні рішення, так як для невеликих компаній, які займаються доставкою здорової їжі за підпискою, вони не підходять через високу ціну (налаштування та ліцензія Microsoft Dynamics CRM) та недоцільність використання.

1.3.1. *GloriaFood*

Даний сервіс широко використовується за кордоном. Надає можливість отримати вебсайт для ресторану та налаштувати обробку замовлень. Для управління доставкою інтегрується з іншими сервісами, які спеціалізуються на доставці онлайн замовлень. GloriaFood має привабливий інтерфейс. Але це

сервіс, який розрахований на ресторани, а не на сервіси доставки їжі за підпискою, йому бракує необхідних можливостей для управління підписками та планами харчування.

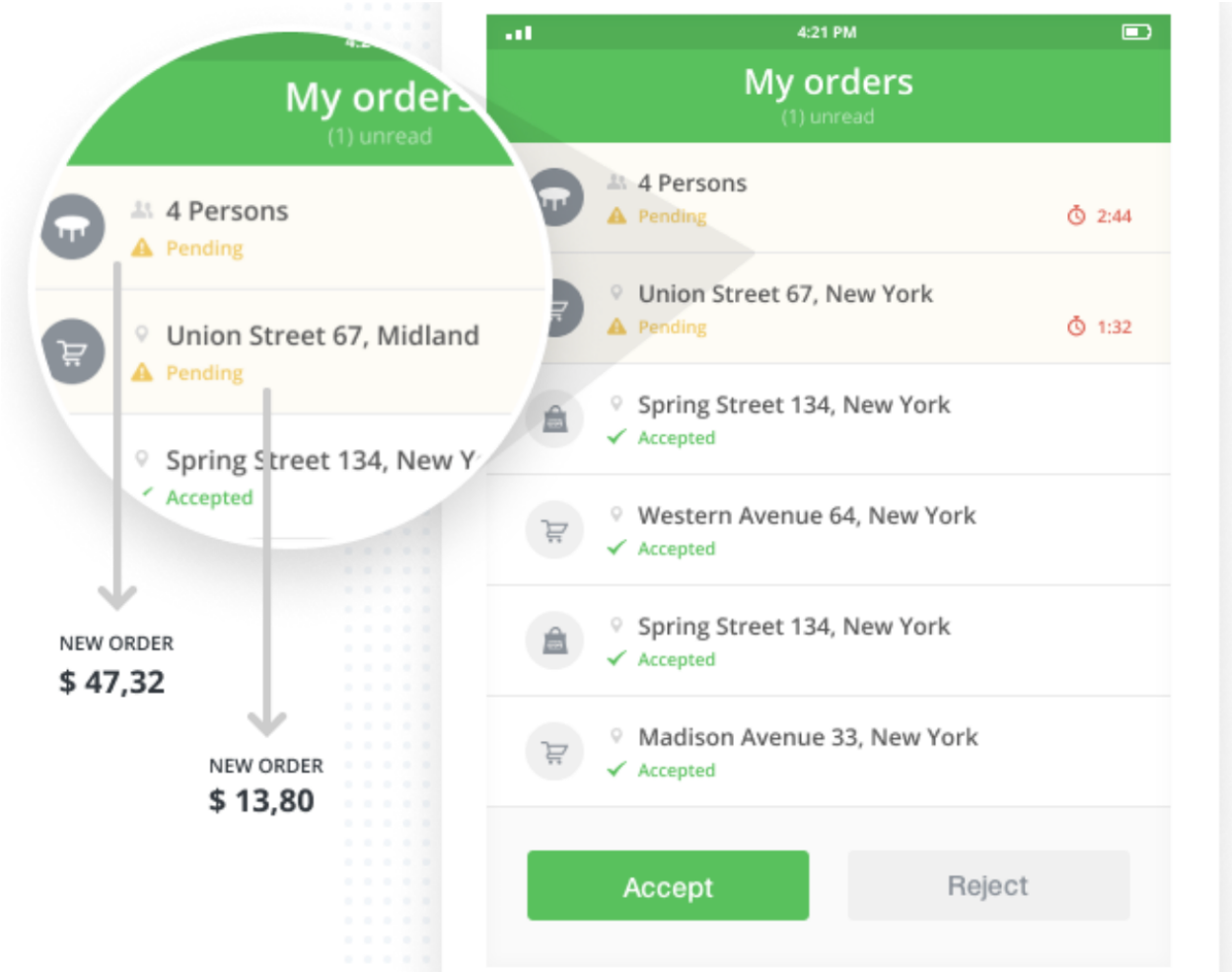


Рис. 1.1 Інтерфейс GloriaFood

1.3.2. *FlipDish*

Даний сервіс також досить поширений закордоном, є достатнього багатим на можливості, як для свого призначення. Він пропонує клієнтам вебсайт та мобільний додаток для отримання замовлень, а також надає додаткові можливості для ресторанів такі як бронювання столику та програмне забезпечення для автоматів самообслуговування в ресторанах та кафе. Але ця система створена передусім для ресторанів та не має потрібних функцій для компаній, що займаються доставкою здорового харчування за підпискою.

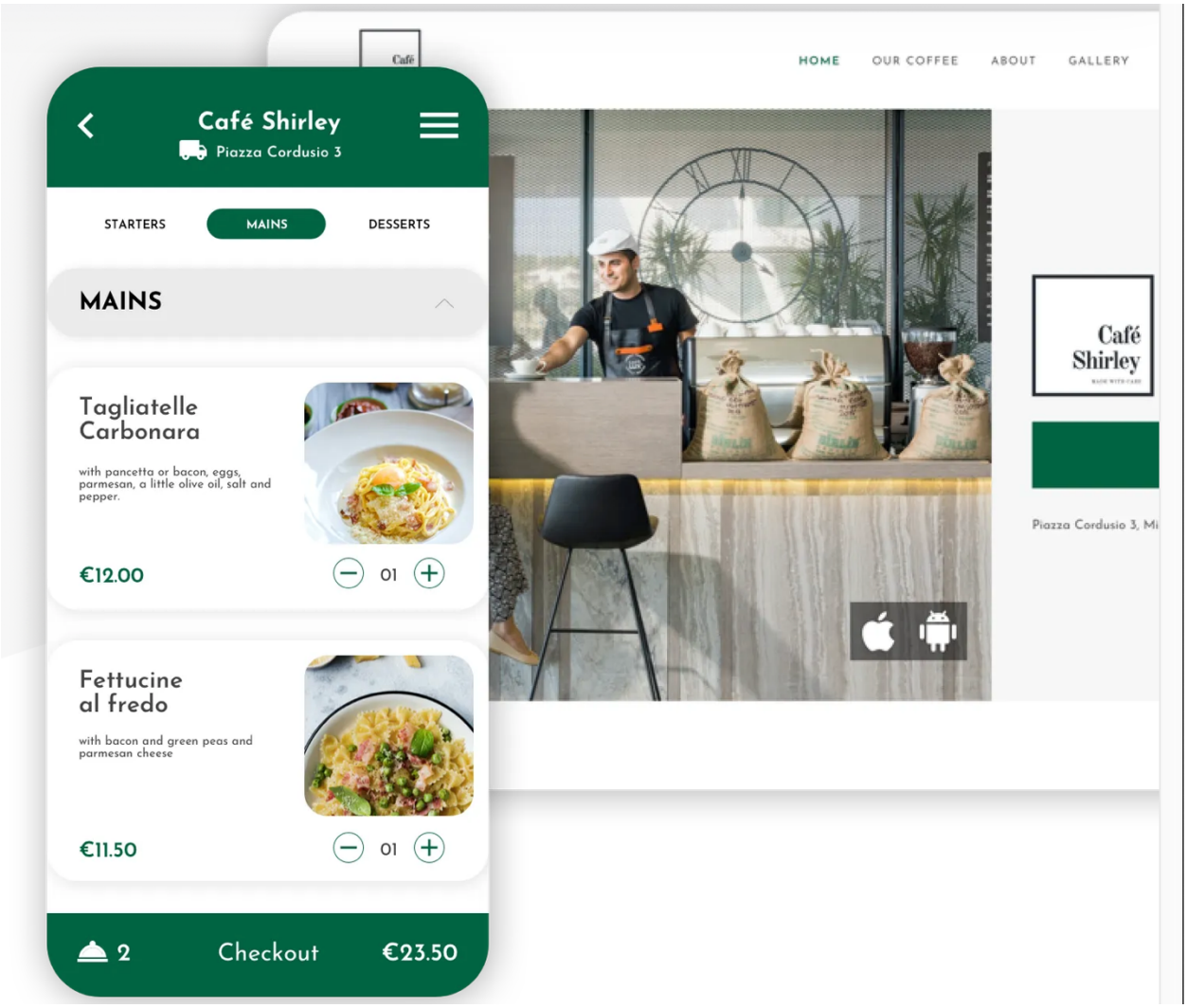


Рис. 1.2 Приклади вебсайту та мобільного додатку що надає FlipDish

1.3.3. *Poster*

Даний сервіс є поширеним у країнах СНГ та має російський інтерфейс. Він надає інтегративну аналітику, редагування меню та поставок, фінансову звітність, створення програм лояльності. Але можливості управління замовленнями додаються як розширення та потребують додаткової оплати за користування. З розширенням для управління замовленнями додаються можливості для управління замовленнями та столиками у ресторані. Але в ньому також бракує можливостей управління підписками та немає можливості керувати планами харчування.

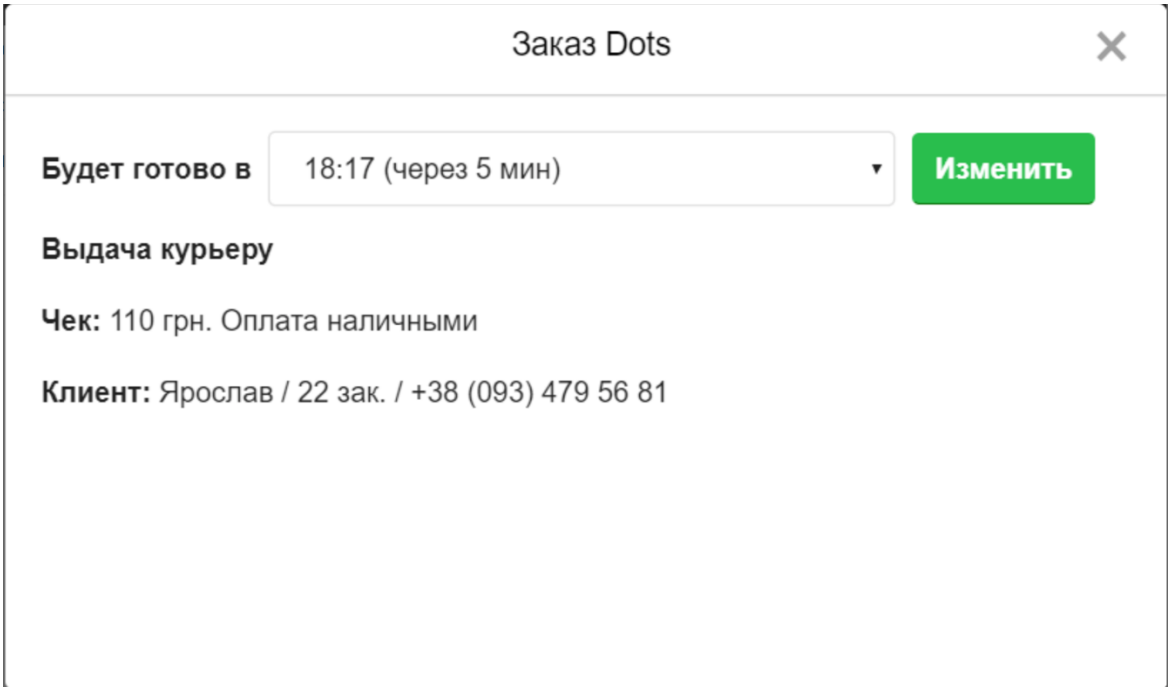


Рис. 1.3 Приклад інтерфейсу Poster

1.3.4. *СБИС Presto*

Цим сервісом користуються більше в країнах СНГ. Він не надає вебсайт з можливістю створення замовлень, але має багато функцій для ресторанів з підтримкою управління замовленнями на доставку. Сервіс надає зручний інтерфейс для офіціантів та інструменти керування рестораном для адміністратора. Але, як і інші подібні системи, сервіс не надає можливості управління підписками та керування планами харчування.

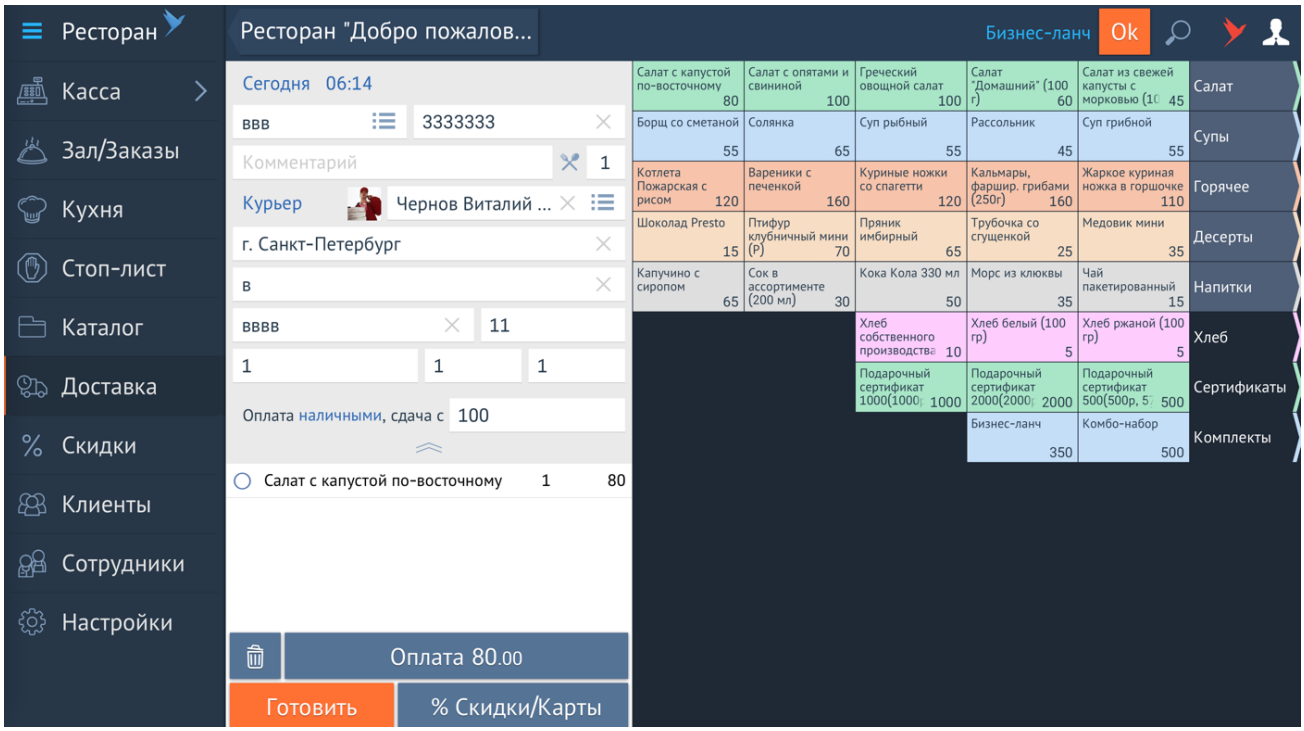


Рис. 1.4 Приклад інтерфейсу СБИС Presto

1.3.5. *SubscriptionFlow*

SubscriptionFlow це комплексна система управління підписками на друковані видання. Вона користується популярністю в США, Великобританії та інших країнах закордоном. SubscriptionFlow надає можливість управління підписками, в тому числі регулярні платежі, зміну статусу підписки, перегляд даних про клієнта та інше. Але система не має можливостей, необхідних для обробки замовлень доставки їжі, планування набору страв на майбутнє, управління планами харчування та інше.

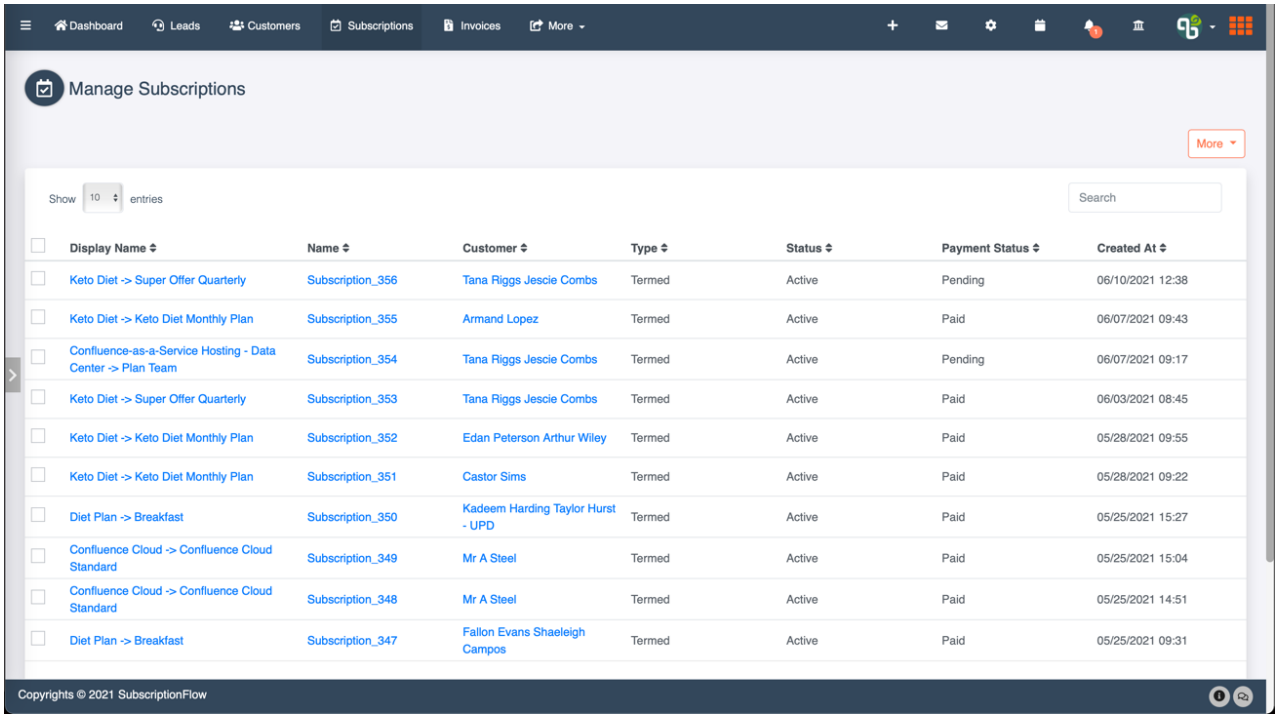


Рис. 1.5 Приклад інтерфейсу SubscriptionFlow

1.3.6. *Subscription DNA*

Subscription DNA це комплексне рішення для управління бізнесом, яке надає можливість оформити підписки на друковані видання. Воно поширене переважно закордоном, має схожий пакет можливостей, як і попередній варіант. Але в ньому немає можливості для управління замовленнями їжі, створення меню та списку доступних страв, планування набору страв на майбутнє та інших критично необхідних можливостей.

1.4. Результати проведеного аналізу

Порівняння можливостей аналогів

Таблиця 1.1

	Gloria Food	Flip Dish	Poster	СБИС Presto	Subscription Flow	Subscription DNA
Вебсайт з можливістю створення замовлення	+	+	-	-	-	-
Перегляд активних замовлень	+	+	+-	+	-	-

Продовження табл. 1.1

Управління підписками	-	-	-	-	+	+
Перегляд активних підписників та продуктів, на які вони підписані	-	-	-	-	+	+
Створення та редагування планів харчування	-	-	-	-	-	-
Планування набору страв для кожного плану харчування	-	-	-	-	-	-

Базуючись на результатах проведеного аналізу, зроблено висновок, що наявні застосунки повністю не вирішують потребу в організації та систематизації замовлень доставки здорової їжі за підпискою.

Всі перераховані системи управління можна розділити на дві категорії: для ресторанів та для підписок на друковані видання. Вони покривають лише частину необхідного пакету можливостей та не підлаштовані під формат «їжа за підпискою».

Жоден з перерахованих сервісів не надає можливостей, які є специфічними для доставки їжі за підпискою, наприклад: створення та редагування планів харчування, планування наборів страв на найближчий час наперед.

Таким чином, для вебсервісів, які дозволяють створити замовлення на доставку їжі “за підпискою” було виділено ряд обов’язкових функцій:

1. вебсайт з можливістю створення замовлення;

2. управління підписками;
3. управління планами харчування;
4. планування наборів страв для кожного плану харчування наперед.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Після уточнення функціональних вимог до застосунку необхідно обрати інструменти для реалізації, такі як бібліотеки, фреймворки, СУБД, мови програмування. Модель вебдодатку складається з фронтенду (клієнтської частини) та бекенду (серверної частини), отож требі підібрати інструменти для обох цих компонентів.

2.1. Вибір мови програмування для розроблення серверної частини

Базуючись на уточнених вимогах, було отримано список критеріїв для мови програмування бекенду:

- підтримка статичної типізації;
- підтримка асинхронного виконання;
- підтримка зручного веб-фреймворку для розробки серверу;
- обєктно-орієнтованість;
- наявність великої кількості бібліотек для вирішення різних завдань.

Розглянемо найбільш поширені мови програмування.

2.1.1. *Python*

Python – це інтерпретована, інтерактивна, динамічна та строго типізована, об’єктно-орієнтована мова програмування високого рівня. Найбільше поширення має серед галузей Data Science (наука про дані). Проста в освоєнні, часто використовується не лише розробниками.

Філософія дизайну Python полягає в тому, що код повинен читатися наче чиста англійська мова. Цей ефект частково досягається за рахунок заміни фігурних дужок та інших функціональних символів на підступи. Також ця мова має прості для прийняття конструкції, що схожі на звичайні фрази в англійській мові. Це має на меті допомогти програмісту сфокусуватися на логіці програми в не витрачати час на написання мовних конструкцій, що потрібні лише для коректного виконання програми. Все це разом з об’єктно-

орієнтованим підходом робить Python одним з найкращих мов програмування для нових розробників завдяки своїм можливостям та лаконічному вигляду.

Ця мова є динамічно-типізованою та з автоматичною очисткою пам'яті, що вже не потрібна програмі яка виконується. Вона дозволяє писати код не лише в об'єктно-орієнтованому стилі та й підтримує функціональну та структурні парадигми програмування.

Ця мова програмування вперше з'явилася на світ у 1980-х роках завдяки її розробнику з іменем Гідо ван Россум та перша версія була опублікована у 1991 році з ім'ям Python 0.9.0 [6]. Наступна версія, що мала номер 2.0 була опублікована у 2000 році. Цей реліз додав багато нових можливостей, таких списки. Також Python обзавівся системо автоматичного збору сміття, що було реалізовано за алгоритмом підрахунку посилань [6]. Третя версія з назвою Python 3.0 опублікована у 2008 році. Цей реліз став істотним переосмисленням мови та, очікувано, не є на 100% сумісним з кодом написаним на попередніх версіях Python отож значна частина коду минулих версій мови не працюватиме без змін з Python 3 [6]. Наразі підтримка Python 2 зупинена після релізу версії 2.7.18 в 2020 році [6].

2.1.2. *Java*

Java - це потужна, строго-типізована мова програмування загального призначення, що базується на ООП. Вона використовується не лише для розробки класичних додатків для ПК але й мобільних додатків та також для виконання завдань Data Science таких як обробка великого об'єму даних. За інформацією наданою компанією інтернет-гігантом Oracle, компанії, яка є власником Java, код написаний на мові Java виконується на близько 3 мільярдах девайсів, що розподілені по всьому світу. Цей факт є свідотством того що ця мова програмування є чи не найпопулярнішою мовою програмування серед усіх [7].

Java є високорівневою мовою програмування з переважно об'єктно-орієнтованим підходом та базується на класах та інтерфейсах щоб зробити код якнайменше залежним від кінцевої імплементації. Це досягається завдяки

інтерфейсам, що представляють собою контракт за яким зовнішній код має взаємодіяти з імплементацією інтерфейсу. Голосною ідеєю цієї мови програмування, її філософією, є те, що розробникам варто лише один раз написати свою програму та вони зможуть виконувати її на більшості доступних платформ та операційних систем. Тобто скомпілювавши програму написану на мові Java її можна запускати на всіх платформах, що підтримують Java та навіть немає необхідності перекомпільовувати програму під цю платформу [8].

Однією з важливих функцій Java є багатопоточність, це дає змогу створювати додатки, які мають можливість одночасно виконувати кілька операцій та завдань та використовувати усі можливості процесору, що встановлений в машину на якій виконується код. Ця функціональність дозволяє програмістам розроблювати інтерактивні додатки, які не будуть зупинятися та переставати відповідати під час високого навантаження [9].

Використання Just-In-Time компіляторів забезпечує мові Java високу продуктивність.

Код написаний на мові Java компілюються в байт-код, який здатен виконуватися на усіх JVM (віртуальних машинах Java), незважаючи на архітектуру пристрою. На даний момент, мова Java є однією з найпоширеніших мов програмування, та за інформацією GitHub частіше всього використовується для клієнт-серверних вебзастосунків та кількість розробників становить більше дев’яти мільйонів.

2.1.3. *C#*

C# є мовою програмування, що використовується для вирішення багатьох типів завдань та підтримує написання коду за багатьма парадигмами, що робить C# майже універсальною мовою розробки. Ця мова має сильну, статичну типізацію та код може бути написаний в імперативному, декларативну, та функціональному стилі програмування. Не зважаючи на це, головною парадигмою мови програмування C# є об’єктно-орієнтована, та зазвичай код створюється за принципами ООП.

Мова C# була представлена компанією IT-гігантом Microsoft у 2000 році, а згодом затверджена як міжнародний стандарт у 2002 році та 2003 роках. C# був створений Андерсом Хейлсбергом. На даний момент командою розробників мови програмування керує Мадс Торгерсен.

До переваг C# належить:

- замість великої кількості шуму (EJB, реалізації приватних статичних класів тощо) ви отримуєте елегантні та зручні власні конструкції, такі як Properties і Events;
- він підтримує власні ідіоми управління ресурсами (оператор using). Java 7 також збирається підтримати це, але C # має це набагато довший час;
- він має Lambdas та LINQ, тому підтримує невелику кількість функціонального програмування;
- це дозволяє як загальну коваріацію, так і противаріацію явно;
- він має динамічні змінні, якщо ви їх хочете;
- краща підтримка перерахування.

2.1.4. *JavaScript*

JavaScript (JS) - це легка, інтерпретована мова без строгої типізації. Хоча вона найбільш відома як мова для скриптів вебсторінок, у багатьох інших середовищах, її також використовують.

Наприклад за допомогою фреймворку Node.js на мові JavaScript можна створювати серверну частину додатків (бекенд). Також цю мову часто називають прототипною.

Мова програмування JavaScript є стандартизованою та регламентується специфікацією ECMAScript. Мова є високорівневою та багатопарадигменою [7].

JavaScript разом з HTML та CSS складають основу технологічного стеку інтернету. Понад 97% вебсайтів використовують його на стороні клієнта для поведінки вебсторінок зазвичай використовуючи сторонні бібліотеки та

фреймворки. Усі поширені на даний момент веббраузери мають JavaScript-двигун, що дозволяє виконувати код на стороні клієнта.

JavaScript є багатопарадигменою мовою, розробники можуть писати код використовуючи різні стилі програмування такі як функціональний, імперативний та навіть подійно-орієнтований. API JavaScript налає більшість необхідних сучасній мові програмування можливостей, включаючи роботу з текстом, датами та виконання регулярних виразів. Також є можливості для керування DOM (об'єктна модель документа) сторінки та набір стандартних структур даних [10].

Спочатку JavaScript-двигуни використовувались лише у веббраузерах, але наразі вони є головними модулями інших систем таких як сервери та інші програми.

Хоча JavaScript має схожі особливості з Java, наприклад назва, набір стандартних бібліотек та синтаксичні конструкції мови, вони сильно відрізняються за дизайном.

Node.js

Node.js – це платформа для виконання програмного коду написаного мовою JavaScript не у середовищі браузера. Проєкт Node.js це проєкт з відкритим вихідним кодом, який публічно доступний, тому як великі компанії так і індивідуальні розробники можуть пливати на цей проєкт пропонуючи функціонал для майбутніх версій. Платформа Node.js найчастіше використовується для написання серверу для клієнт-серверних веб додатків, це може бути як основний сервер так к спеціальний сервер для кращої роботи клієнтської частини, відображення (DFF- backend for frontend, бекенд для фронтенду).

Для виконання JavaScript коду використовується двигун V8, такий же двигун мають браузери побудовані на проекті Chromium (браузер з відкритим вихідним кодом, на його основі працює Google Chrome, оновлений Microsoft Edge та інші веббраузери).

Головна ідея Node.js полягає у можливості використання мови програмування JavaScript не тільки на клієнтській частині але й на серверній, це дозволяє програмістам, що працюють з мовою JavaScript розроблювати як клієнтську так і серверну частини [11]. Зазвичай програмістів, що займаються цим називають Full Stack (повний стек) розробниками. Ця особливість мови JavaScript дозволяє їй бути однією з найпоширеніших мов програмування у світі та часто найпершою мовою для розробників-новачків.

Назва проєкту Node.js є лише назвою а не є якимось особливим файлом з розширенням .js яке є стандартним для файлів з JavaScript-кодом.

Й хоча Node.js виконує мову програмування JavaScript, яка не підтримує багатопоточність, Node.js надає багато можливостей завдяки стандартному API втому числі для використання функцій операційної системи, який використовує libuv (багатоплатформена бібліотека, яка написана на мові C) для не блокуючого потік асинхронного вводу-виводу. Тому Node.js славиться своєю швидкістю виконання асинхронних операцій. Також це дозволяє будувати дуже швидкі серверні програми за допомогою горизонтального масштабування Node.js програм. Зазвичай це корисно при побудові додатків яким не потрібно виконувати важкі математичні обчислення (з високим використанням процесорного часу) та якщо є потреба швидко обробляти велику кількість запитів з операціями введення-виведення чи додаток працює з потоковим обміном даними з мінімальною затримкою (наприклад додатки для відео-конференцій та браузерні ігри).

Спочатку проєкт розробки Node.js керувався Node.js Foundation, але на сьогодні були об’єднані з JS Foundation з метою створення OpenJS Foundation, яка й керує проектом Node.js по цей сас.

Список компаній, що використовують програмне забезпечення Node.js складають такі гіганти як IBM, LinkedIn, Microsoft, Netflix, PayPal, Walmart, Yahoo! та Amazon Web Services [11].

2.1.5. *TypeScript*

TypeScript – це мультипарадигмена та компільована мова програмування головним завданням якої є додання статичної типізації до мови програмування JavaScript (яка є динамічно-типізованою).Типи дозволяють описати властивості об'єкта, покращуючи зрозумілість та легкість прийняття колу та та дозволяючи TypeScript перевірити, чи правильно працює код. TypeScript-код компілюється в JavaScript-код.

Цю мову розробляє компанія Microsoft з 2012 року [12].

2.1.6. *Висновки*

Для реалізації було обрано платформу C#, оскільки він є гнучким, підтримує сучасні технології, є статично типізованим, має високу швиткодію та підтримується компанією технологічним гігантом Microsoft. Мова C# є популярною та широко використовується для створення Web-додатків. Для реалізації back-end на мові програмування C# було обрано фреймворк .NET 5 тому, що він підтримується Microsoft та є кросплатформним.

2.2. Вибір технології для розроблення клієнтської частини

Останнім часом чи не єдиною мовою програмування для інтерактивних вебсайтів залишається JavaScript, його й буде використано у цьому проєкті.

Але в сучасному світі програмувати клієнтську частину на чистому JavaScript не доречно, займе багато часу та призведе до низької якості програмного рішення. Сьогодні існує близько 5 поширених та актуальних фронтенд фреймворків але для цього дипломного проєкту було розглануто лише 3 найпоширеніші веб фреймворки: React, Angular та Vue.

2.2.1. *Vue.js*

Vue.js – третій по популярності фронтенд фреймворк. Він імплементує парадигму Model-view-viewmodel (MVVM), що використовується для побудови користувальницьких інтерфейсів та односторінкових додатків.

Vue.js був написаний в 2014 році. Цей фреймворк неабияк полюбляють розробники (у ньому більше зірок GitHub, ніж у React).

Vue розробляється і підтримується основною командою з двадцяти розробників, і хоча він не підтримується безпосередньо одним з Інтернет-гігантів, він використовується у продуктах таких компаній, як Alibaba, Gitlab та Adobe. Vue має, мабуть, найкращу документацію серед усіх згаданих тут фреймворків а також є декілька форумів присвячених цьому фреймворку де можна отпримати необхідну інформацію чи отримати допомогу у вирішенні якоїсь проблеми. Vue також популярний серез PHP розробників і поставляється як частина Laravel.

Для створення компонентів відображення Vue ввикористовує шаблони з синтаксисом схожим на HTML. Для вставлення даних в шаблон фреймворк надає можливість зв'язувати атрибути та моделі даних. Він також дає змогу створювати окремі компоненти, в яких шаблон, JavaScript-код та CSS зберігається у спільному файлі. Vue має багатий інструментарій. Існує офіційний CLI для створення та розробки з Vue, а для Chrome і Firefox є розширення devtools, яке допомагає з відладкою.

2.2.2. *Angular*

Angular - це фронтенд фреймворк створений інтерент-гігантом Google. Він був створений у 2010 році як AngularJS (або Angular 1) і був хітом свого часу, перш за все тому, що це був перший фреймворк, який дозволив розробникам створювати SPA (односторінкові вебдодатки) [13].

Щоб вирішити проблеми з швидкодією та проблеми масштабування, Google переписав AngularJS з нуля та в 2016 році випустив Angular 2 (на сьогоднішній день просто Angular). Зараз AngularJS вкрай є застарілим, і не повинен використовуватися для нових проектів.

Що стосується Angular, то він використовується такими компаніями, як Google і Microsoft у своїх продуктах, тому, безумовно, добре перевірений та є вартим довіри. В Інтернеті також доступно багато ресурсів з інформацією про Angular, і на Stack Overflow є велика кількість питань, пов'язаних з Angular.

На відміну від React, який відповідає лише за відображення, Angular пропонує комплексне рішення для створення односторінкових клієнтських додатків. Компоненти можуть реалізовувати двосторонню прив'язку даних, що дозволяє їм одночасно прослуховувати події та оновлювати значення у батьківських та дочірніх компонентах. Для створення представлення компонента використовуються шаблони з кодом HTML та з специфічним синтаксисом Angular для використання функцій Angular та в’язання з моделлю для вставки даних в шаблон. TypeScript є основною мовою для розробки в Angular, що робить його беспечнішим отож він довбре підходить для корпоративних програм.

Інструментарій Angular має чудовий. Angular пропонує офіційний CLI для ініціалізації, розробки та обслуговування додатків на Angular. Існують також розширення Chrome і Firefox Dev Tools, доступні для відладки програм Angular. Angular має власні рішення для вирішення багатьох поширених завдань, таких як форми та маршрутизація, але все ще існує велика екосистема сторонніх бібліотек.

2.2.3. *React.js*

React – це найпопулярніший фронтенд фреймворк, доступний сьогодні. Він був випущений Facebook у 2013 році. React використовується у своїх продуктах такими компаніями, як Facebook, Netflix та Airbnb, і він має багато розробників - це означає, що в Інтернеті легко знайти допомогу та необхідну інформацію [13].

Основною метою React є складання інтерактивних інтерфейсів користувача з багаторазових компонентів. Він використовує JSX (розширення синтаксису до JavaScript) для шаблонування та використовує модель одностороннього потоку даних для вставки даних в представлення. Ці дані оновлюються в представленні щойно змінилися дані компонента. Також особливістю React є використання віртуального DOM, це зроблено для того

щоб модифікувати реальний DOM лише у тому місці де відбулася зміна та економити час на даремних операція з DOM, які є відносно повільними.

Інструментарій розробника – достатньо хороший. Команда React створила та підтримує CLI (Create React App) для швидкого та легкого створення нового проекту, а також розширення інструментів розробника для Chrome і Firefox.

React не намагається надати повну бібліотеку для створення вебдодатків. Він створений лише для розробки вебінтерфейсу додатку й бракує багатьох інструментів, які часом можуть бути необхідними для створення вебдодатку. Але це не є критичним недоліком адже існують зручні бібліотеки для покриття іншої необхідної функціональності й розробники можуть самостійно обирати які інструменти використовувати в своєму проєкті. Оскільки React вже можна вважати зрілою бібліотекою, існують загальні моделі його використання.

В доповнення до React була представлена архітектурна концепція Flux, яка базується на односпрямованому потоку даних (на відміну від двонапрямленого потоку в Angular) від моделі до представлення. Архітектура Flux є альтернативою вже досить поширеній архітектурній концепції MVC (модель-представлення-контролер) [14].

2.2.4. *Висновки*

Для реалізації користувацького інтерфейсу було обрано мову програмування (додаток до мови JavaScript) TypeScript адже він додає статичну типізацію й використовується у фреймворку Angular.

2.3. Вибір СУБД

Керування даними у додатку відіграє майже не одну з вирішальних ролей у наданні користувачу позитивного досвіду від використання додатку. Адже додаток повільно підгружає дані та реагує на користувацькі даї з затримкою то не навіть добре розроблений інтерфейс додатку та те наскільки правильно структурований та організований код не можуть затьмарити

негативного враження від системи. Окрім цього усі ці дані слід захищати, щоб зловмисники не могли їх дістати.

База даних - це місце, де ви зберігаються та впорядковуються всі дані, які збираються через вебдодаток, а тип програм що дозволяють керувати базою даних у зручний спосіб називається СУБД (система управління базами даних) [15].

2.3.1. *Вибір моделі СУБД*

На сьогоднішній день на ринку існує понад 300 СУБД. Вибір між такою кількістю інструментів справді надзвичайний але тут розглянемо лише найпоширеніші варіанти.

Усі БД можна класифікувати на реляційні (SQL) та не-реляційні (NoSQL). Реляційна база даних - це сукупність таблиць, які мають визначені зв'язки між собою. Для підтримки та запиту реляційної бази даних система управління базами даних використовує структуровану мову запитів (SQL).

Найкраще реляційні бази справляються з збереженням та обробкою саме структурованих даних, таких як дані профілю користувача, паспортні дані, технічні параметри телефону та інше. Найпоширенішими прикладами таких баз є є PostgreSQL, MySQL та MS SQL Server.

Бази даних NoSQL краще використовувати там де є необхідність зберігати та оброблювати саме неструктуровані дані (наприклад фотографії, файли, великі об'єми тексту) адже вони пропонують розробникам більшу гнучкість та більшу масштабованість [19].

Для реалізації вебзастосунку для управління замовленнями їжі за підпискою було обрано реляційну БД тому, що типи даних та сутностей заздалегідь відома, отож дані є структурованими.

2.3.2. *MySQL*

MySQL - одна з найпопулярніших СУБД, яка була представлена у 1995 році та підтримується компанією Oracle [24]. Вона має велику спільному користувачів та є програмним забезпеченням з відкритим вихідним кодом. На

сьогодні більшість бібліотек та фреймворків підтримує MySQL. Також вона є безкоштовно, але можна отримати додаткову функціональність за фіксовану ціну [16].

Розробники можуть встановлювати та використовувати MySQL, не витрачаючи багато часу на його налаштування. Є можливість виконати більшість операцій у командному рядку. Також для неї регулярно виходять оновлення та є якісна підтримка.

MySQL належав і фінансувався шведською компанією MySQL AB, яку згодом придбала Sun Microsystems (на даний момент Oracle Corporation) [16].

Для MySQL існує програмне забезпечення для керування базами даних безпосередньо але в більшості випадків MySQL використовується іншими програмами, які потребують реляційної бази даних. MySQL є компонентом стеку програмного забезпечення для вебдодатків LAMP (Linux, Apache, MySQL, PHP) та інших. Також MySQL використовується багатьма популярними вебсайтами, включаючи Facebook, Flickr, Twitter та YouTube.

Переваги:

- широка функціональність – MySQL підтримує більшість можливостей SQL;
- безпека – розробники попіклувались про безпеку в MySQL тому багато можливостей, які орієнтовані на безпеку доступні за замовчення;
- масштабованість – MySQL може легко працювати з великою кількістю даних та легко масштабується;
- швидкість – завдяки спрощенню деяких стандартів SQL, MySQL може швидше працювати.

Недоліки:

- обмеження – особливістю MySQL є те, що в неї закладені деякі обмеження, які час від часу можуть бути необхідними в деяких особливо вимогливих додатках;

- проблеми з надійністю – при використанні частини функціональності MySQL може програвати у надійності іншим базам даних.
- повільний випуск оновлення – є скарги користувачів на повільний процес розробки та випуск оновлень. Хоча треба зазначити, що існують інші СУБД розроблені на основі MySQL, наприклад MariaDB.

2.3.3. *PostgreSQL*

PostgreSQL, також відомий як Postgres, СУБД, яка створена передусім для обробки терабайтів даних тому чудово підходить для великих систем, аже він легко масштабується. На відміну від MySQL, PostgreSQL - це абсолютно безкоштовна база даних. Її код є відкритим й це означає, що вона документується та підтримується величезною спільнотою розробників [17].

PostgreSQL має транзакції з властивостями ACID (Atomicity, Consistentity, Isolation, Durability), автоматично оновлювані представлення (Views), тригери, зовнішні ключі та збережені процедури. Він може бути використаний для обробки широкого спектру робочих навантажень, від окремих серверів до спеціальних сховищ даних та вебсервісів із великою кількістю одночасних користувачів. PostgreSQL доступний на багатьох платформах таких як Linux, Windows, FreeBSD, OpenBSD.

Переваги PostgreSQL:

- безкоштовний, відкритий вихідний код;
- має велику спільноту розробників;
- є багато доповнень та розширень;
- підтримує об’єктно-орієнтований підхід, наслідування.

Недоліки PostgreSQL:

- швидкодія – при простих операціях читання PostgreSQL може значно сповільнити сервер і бути повільніше своїх конкурентів;
- не користується великим попитом.

2.3.4. *MS SQL*

Microsoft SQL Server - це реляційна СУБД представлена компанією Microsoft. Цей продукт представляє собою сервер баз даних. Його можна встановити локально (для тестування наприклад) чи на іншому сервері для того щоб доступатися до БД через мережу (локальну чи інтернет). Компанія випускає біля десятку різноманітних версій цього продукту для різних типів проектів, від маленьких програм до масштабних програмних систем.

До переваг Microsoft SQL Server належать все переваги реляційних систем управління БД та інтеграція з .NET та Azure.

2.3.5. *Висновки*

Базуючись на проведеному досліджені було обрано СУБД Microsoft SQL Server через її більшу зручність для додатків розроблених на платформі .NET, яким і є розроблюваний додаток.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ПЗ

3.1. Загальний опис системи

Вебзастосунок для створення та обробки замовлень доставки їжі за підпискою представляє собою вебсайт, який дозволяє своїм користувачам-клієнтам створити замовлення, а адміністраторам редагувати доступні плани харчування та оброблювати замовлення. Клієнт може вказати небажані інгредієнти та страви, які мають бути виключені з страв, що будуть доставлені клієнту. Це допоможе зробити замовлення доставки їжі за підпискою доступним для ще більшого кола потенційних клієнтів, адже цим зможуть скористатися люди з особливими потребами та вимогами до харчування, наприклад вегани та люди з алергією на який-небуть інгредієнт. Ця можливість дає перевагу у конкурентній боротьбі за клієнтів, тому що це буде чи не єдиний сервіс, що надає таку можливість.

3.1.1. *Аналіз проблематики та цілей вебдодатку*

Базуючись на характеристиці вебзастосунку, було отримано список проблем, вирішення яких є метою створення цього застосунку.

Отримані проблеми:

- персонал займається рутинною роботою, яку можна автоматизувати та як результат трапляються помилки через людський фактор та час персоналу використовується неефективно;
- неможливість збереження історії замовлень клієнта на довгий термін;
- неможливість швидко знаходити інформацію про минулі замовлення клієнта.

На основі описаних проблем, було створено дерево проблем, яке знаходиться на рис. 3.1.

Рис. 3.1. Дерево проблем продукту

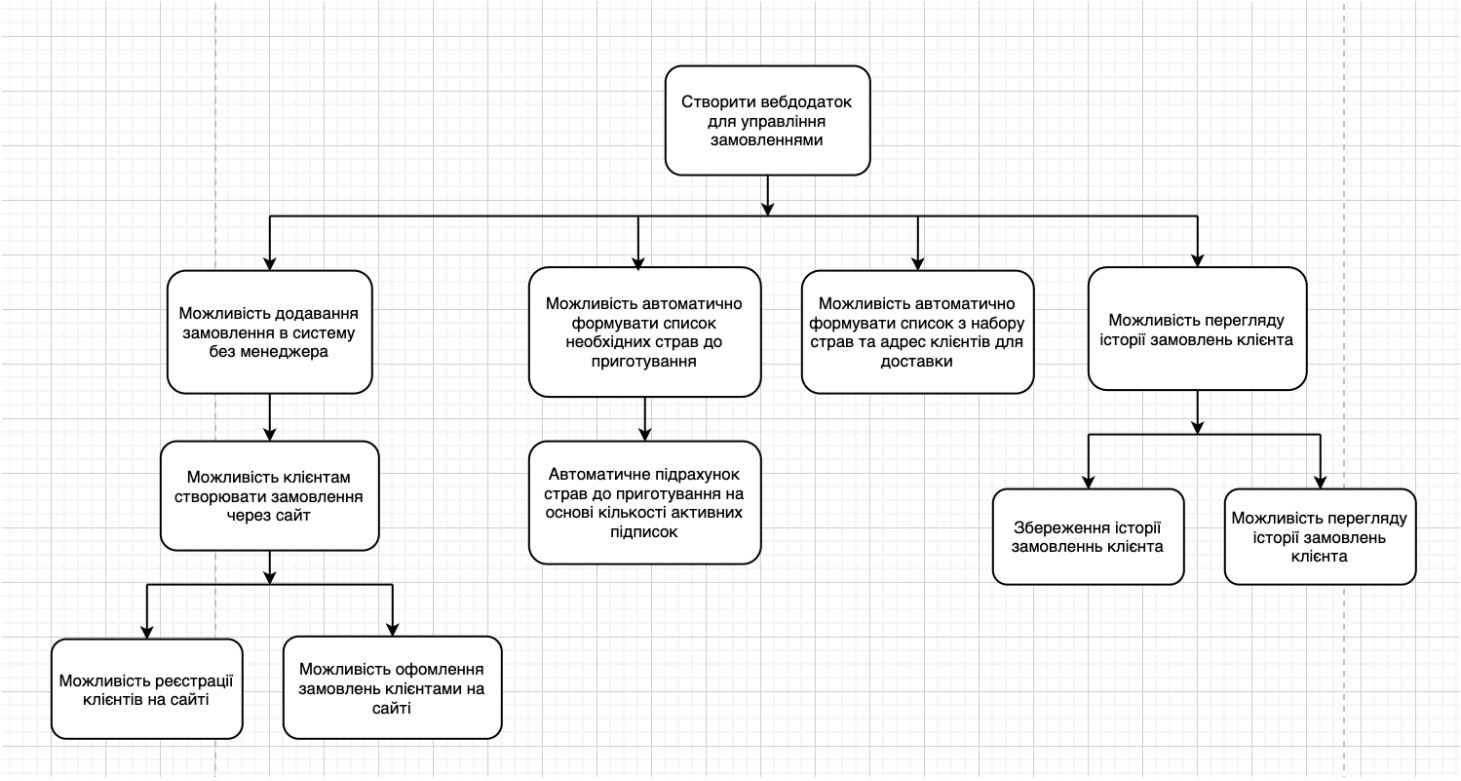


Рис. 3.2. Дерево цілей продукту

Дослідивши проблеми предметної області, було сформовано цілі, які повинні бути досягненими під час розробки вебзастосунку для створення та

обробки замовлень доставки їжі за підпискою призведе до повного вирішення наявних проблем. Загальною ціллю розроблення даного вебзастосунку є надання програмного рішення, завдяки якому компанії, які надають такий сервіс, зможуть легко оброблювати замовлення та зробити свій бізнес ефективніше, а також завжди мати доступ до історії замовлень клієнта.

На основі сформованих цілей отримано зображено дерево цілей проєкту (рис. 3.2).

Також було отримано дерево результатів проєкту (рис. 3.3).

Після проведення аналізу проблемної області вебдодатку для створення та обробки замовлень доставки їжі за підпискою, та визначивши цілі та результати, досягнення яких є метою створення застосунку, було сформовано перелік вимог, яким має відповідати розроблюване програмне рішення.

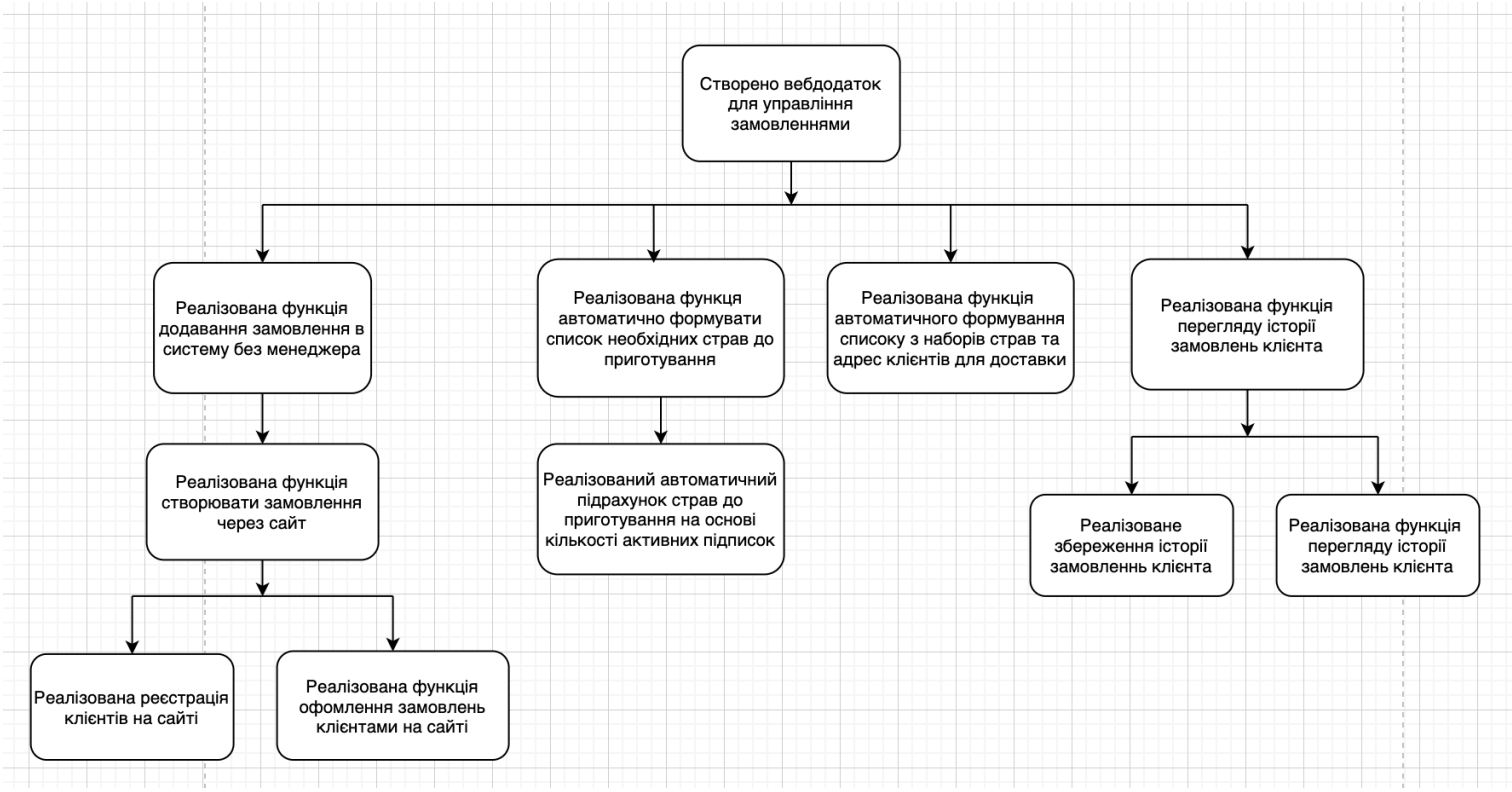


Рис. 3.3. Дерево результатів продукту

3.1.2. *Формулювання вимог до програмного забезпечення*

Отже, базуючись на характеристиці системи, було отримано список вимог.

Таблиця 3.1

Вимоги до продуктивності

Код вимоги	Опис вимоги
PER -1	Вебдодаток має завантажуватися не більше ніж за 3 секунди.
PER-2	Швидкість відповіді системи має складати не більше ніж 1 секунда.
PER-3	Система має витримувати навантаженя в 1000 запитів на сек.

Таблиця 3.2

Вимоги до безпеки

Код вимоги	Опис вимоги
SEC-1	Система повинна мати захист від відомих атак типу XSS, CSRF, SQL Injection
SEC-2	Паролі користувачів мають бути збереженими у вигляді після хешування, яке не дозволить декодувати код та отримати оригінальний пароль.

Таблиця 3.3

Вимоги до реалізації

Код вимоги	Опис вимоги
IMP-1	Бекенд вебдодатку має бути створений з монолітною архітектурою.
IMP-2	Додаток має підтримувати поширені вебпереглядачі якими користуються 95% інтернет-користувачів.

Щоб точніше описати вимоги, було зображено сценарії використання додатку.

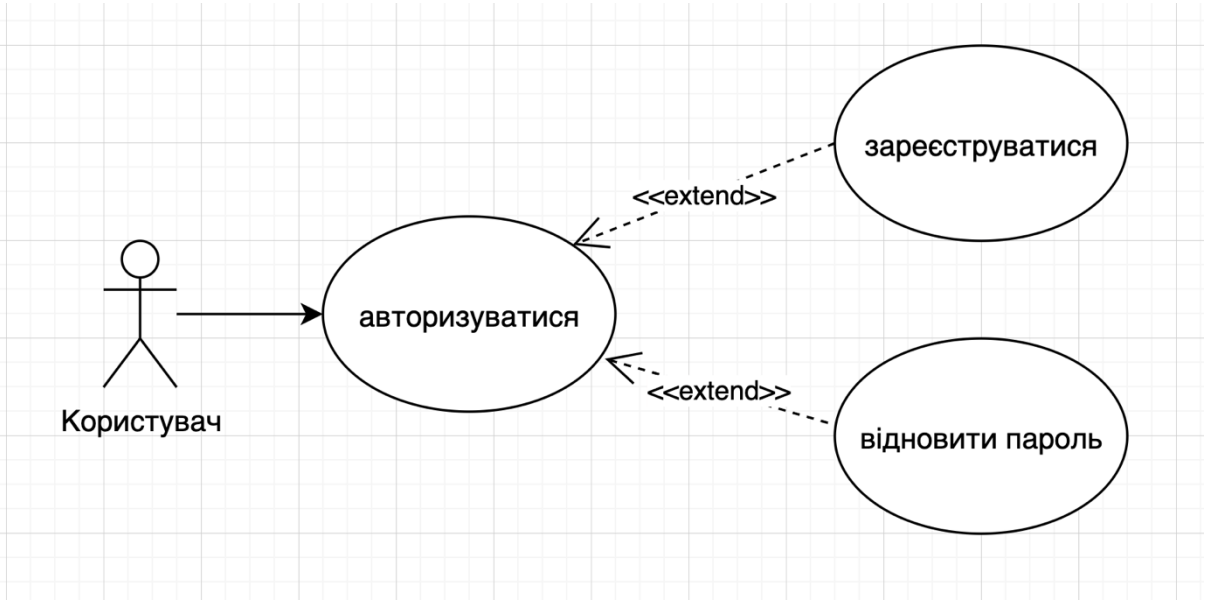


Рис. 3.4. Прецедент: Створення нового облікового запису або авторизація користувача в системі

Прецедент 1: «Створення нового облікового запису або авторизація користувача в системі» (рис. 3.4).

1. Користувач завантажує сторінку додатку.
2. Користувач відкриває форму авторизації або реєстрації.
3. Користувач заповняє форму.
4. Користувач підтверджує електронну адресу за допомогою перевіркового листа з посиланням на сторінку підтвердження, якщо користувач реєструється.
5. Якщо користувач авторизується та втратив чи втратив свій пароль для доступу в додаток, він має можливість надіслати запит на скидання паролю та отримати листа на електрону пошту з посиланням на сторінку зміни паролю.

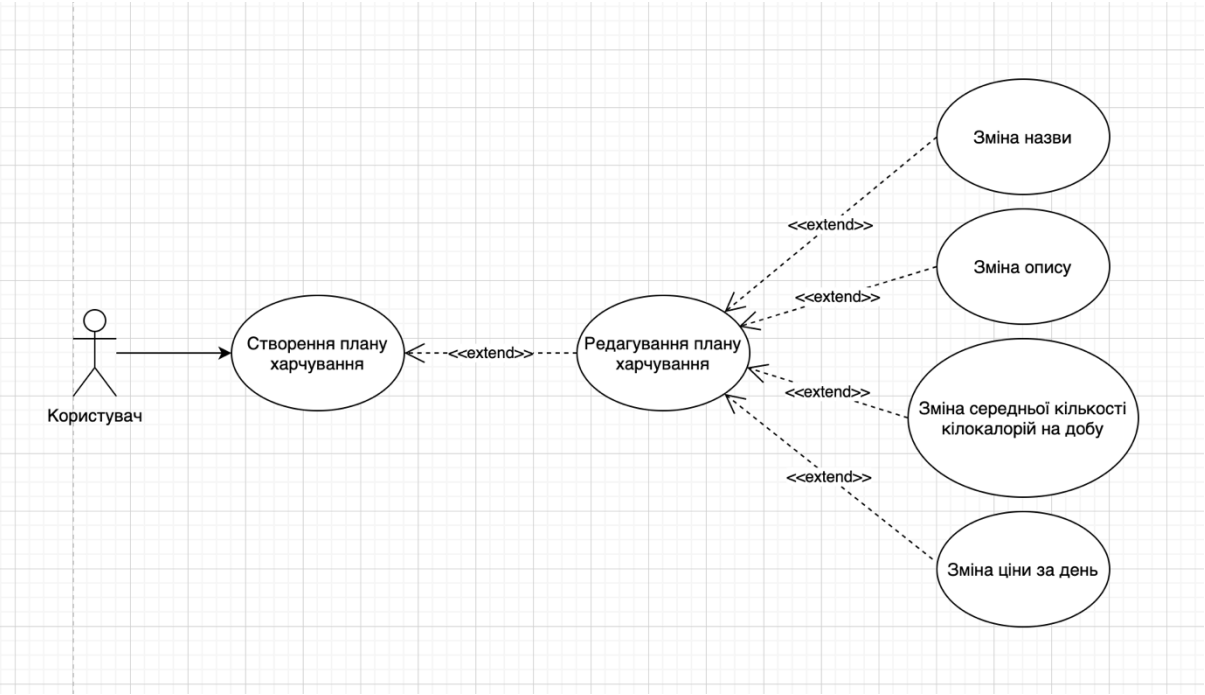


Рис. 3.5 Прецедент: Створення або редагування доступних планів харчування

Прецедент 2: «Створення або редагування доступних планів харчування» (рис. 3.5).

Передумова: користувач авторизований та є адміністратором.

1. Користувач знаходиться на сторінці адміністратора.
2. Користувач обирає підрозділ «Плани харчування».
3. Користувач обирає план харчування або тисне «створити».
4. Користувач заповнює або редагує необхідну інформацію.
5. Користувач зберігає план харчування натискаючи кнопку «зберегти».

Прецедент 3: «Оформлення підписки на доставку їжі» (рис. 3.6).

Передумова: користувач авторизований.

Користувач знаходиться на головній сторінці.

1. Користувач тисне «оформити підписку» й потрапляє на сторінку оформлення підписки.
2. Користувач додає всю необхідну інформацію для оформлення підписки.
3. Користувач бачить що йому буде доставлено наступні 7 днів.

4. Користувач тисне «оформити» й цим підтверджує оформлення підписки.

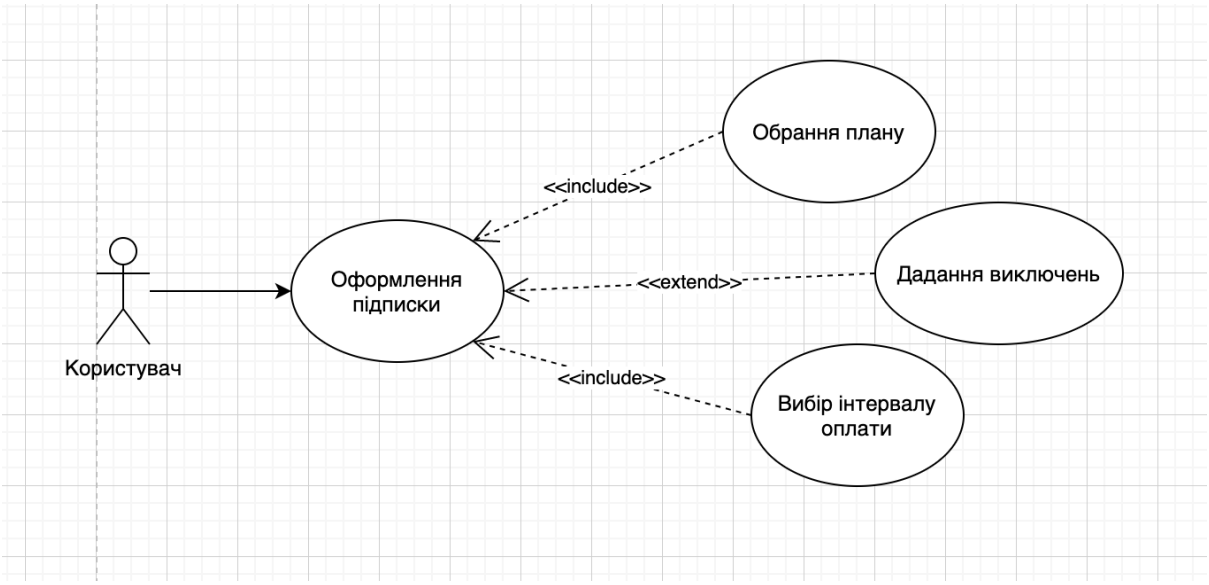


Рис. 3.6. Прецедент: Оформлення підписки на доставку їжі

Прецедент 4: «Перегляд списку що потрібно приготувати на наступний день»

Передумова: користувач авторизований та є адміністратором.

1. Користувач знаходиться на сторінці адміністратора.
2. Користувач натискає «Потрібно приготувати».
3. Користувач переглядає список страв що необхідно приготувати та усі виключення які необхідно врахувати.

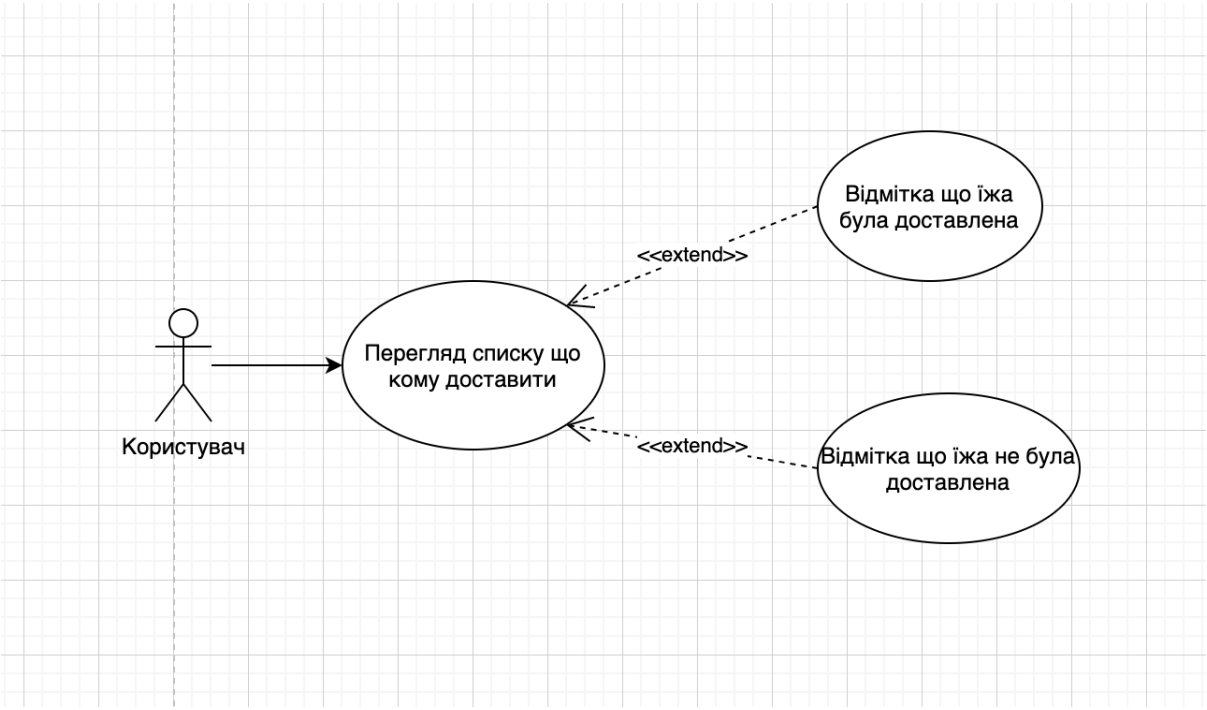


Рис. 3.7. Прецедент: Перегляд інформації для доставки

Прецедент 5: «Перегляд інформації для доставки» (рис. 3.7).

Передумова: користувач авторизований в системі та є адміністратором.

1. Користувач знаходиться на сторінці адміністратора.
2. Користувач тисне «доставка».
3. Користувач переглядає інформацію необхідну для доставки (якій набір страв та за якою адресою).
4. Користувач може відмітити що за якоюсь адресою не було доставлено замовлення щоб потім менеджер зв’язався з цим клієнтом.

Базуючись на сценаріях користування, отримали функціональні вимоги (табл. 3.4).

Таблиця 3.4

Функціональні вимоги до вебдодатку

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F1	Реєстрації користувачів з використанням електронної пошти та паролю	Високий	Середня
F2	Відновлення паролю за допомогою листа на електрону пошту	Середня	Середня
F3	Авторизація користувачів за допомогою електронної пошти та паролю	Високий	Середня
F4	Перегляд існуючих планів харчування неавторизованим користувачем	Високий	Низька
F5	Оформлення підписки	Високий	Середня
F6	Редагування планів харчування авторизованим адміністратором	Високий	Середня
F7	Додання страв що будуть доставлені протягом тижня	Високий	Середній
F8	Перегляд списку що потрібно приготувати на наступний день	Високий	Середній
F9	Перегляд списку для доставки (набір страв та адреса)	Високий	Середній
F11	Редагування меню на тиждень	Середній	Середній

3.2. Структурна організація програмного забезпечення

3.2.1. *Обрання типу архітектури ПЗ*

Поширеними типами архітектури є монолітна архітектура та мікросервісна архітектура.

Монолітна архітектура

Переваги монолітної архітектури:

- розпочати новий проєкт та розробити його простіше, використовуючи монолітну архітектуру;
- набагато простіше тестувати систему, яка є монолітом;
- розгортання це встановлення раніше підготовленої програми на сервер.

Як і будь-яке рішення, монолітна архітектура має свої недоліки. Ось перелік деяких недоліків:

- велику базу коду може бути значно важче зрозуміти;
- інтегроване середовище розробки може перевантажитися, а розмір може також уповільнити час запуску;
- кожен елемент тісно пов'язаний і залежать від інших, тому важко перейти на нову або вдосконалену технологію, мову чи структуру;
- оновлення може бути проблемою, оскільки це перенесення програми;
- проблеми з масштабованістю, оскільки кожен елемент має різні вимоги до ресурсів.

Мікросервісна архітектура

Одним з найпоширеніших аргументів для використання мікросервісної архітектури є те, що це може полегшити обслуговування програмного забезпечення, розкладаючи та роз’єднуючи його на менші блоки. Порівнюючи монолітну архітектуру та мікросервісну архітектуру, більшість монолітів недостатньо модулізовані, а їх особливості непросто перевірити (модульне тестування, поведінкове тестування). Розгортати зміни стає важко, оскільки через щільний зв'язок між модулями досить легко порушити деякі інші функції.

На відміну від монолітної архітектури, яка складається з одного елемента, мікросевіси складаються з сервісів що взаємодіють між собою. Кожен з них - це невеликий сервіс, який виконує функції з однієї сфери

бізнесу. Мікросервіси народились із SOA (Service-Oriented Architecture) як рішення проблем, пов'язаних з монолітними додатками. Поділ великих проєктів на малі набуває все більшого значення, особливо серед розробників, і вплив мікросервісів на електронну комерцію є значним. Однією з головних причин, чому незалежні модулі зустріли позитивний прийом, є здатність керувати API та виконувати конкретні завдання. Архітектура мікросервісну ділить послуги на невеликі автономні елементи, кожен зі своїми незалежними функціями.

Але використання мікросервісної архітектури доречне лише для великих проєктів, лише в такому випадку переваги перевищують недоліки. Тому для розроблюваного застосунку було обрано монолітну архітектуру оскільки додаток є відносно малим проєктом та не передбачає значне розширення функціональності в майбутньому.

3.3. Структура БД

3.3.1. *Користувач (User)*

- Ідентифікатор (ID) – унікальний ідентифікатор.
- Ім’я (Name) – Ім’я користувача.
- Улектронна пошта (Email) – електронная пошта користувача.
- Пароль (Password) – захешований пароль користувача.
- Дата створення (Created at) – дата реєстрації користувача.
- Тип користувача (Type) – вказує на роль користувача (клієнт чи адміністратор).
- Дата видалення (Deleted at) – відповідає статусу профіля користувача, якщо видалений – наявна дата в цьому полі.

3.3.2. *Клієнт (Client)*

- Ідентифікатор (ID) – унікальний ідентифікатор запису у таблиці User.
- ПІП (name) – прізвище, ім’я та по-батькові.

- Номер мобільного телефону (Phone) – номер мобільного телефону (обов’язкове поле для клієнтів).
- План харчування (food_plan) – активний план харчування (якщо є).
- Інтервал оплати (payment_interval) – інтервал стягування оплати (наприклад кожні 7 днів чи місяць).
- Адреса доставки (address) – адреса за якою будуть доставлятися страви.
- Список виключень (exclusion_list) - Список інгредієнтів чи страв для виключення.

3.3.3. *План харчування (Food plan)*

- Ідентифікатор (ID) – унікальний ідентифікатор запису у таблиці.
- Назва (Name).
- Опис (Description) – детальний опис плану харчування, може включати інформацяю які страви можуть входити.
- Кілокалорії на день (kcal) – середня кількість кілокалорій на день.
- Дата створення (Created at) – дата створення плану харчування.
- Дата останнього оновлення (Updated at) – дата останнього оновлення.
- Дата видалення (Deleted at) – відповідає статусу плана харчування, якщо видалений – наявна дата в цьому полі.

3.3.4. *Страва (Dish)*

- Ідентифікатор (ID) – унікальний ідентифікатор страви.
- Назва (Name).
- Опис (Description) – опис страви, може включати інформацяю, які інгредієнти входять.
- Дата створення (Created at) – дата надіслання повідомлення.
- Дата видалення (Deleted at) – дата коли страва була видалена.

3.3.5. *Елемент меню (MenuItem)*

- Ідентифікатор (ID) – унікальний ідентифікатор елемента меню.
- ПланХарчування (FoodPlanId) – ідентифікатор плану харчування.
- Страва (DishId) – ідентифікатор страви.
- Дата (Date) – дата коли ця страва буде доставлятися для цього плану харчування.

3.3.6. *ERD-діаграма БД*

Діаграма зав’язків зображена на рис. 3.8.

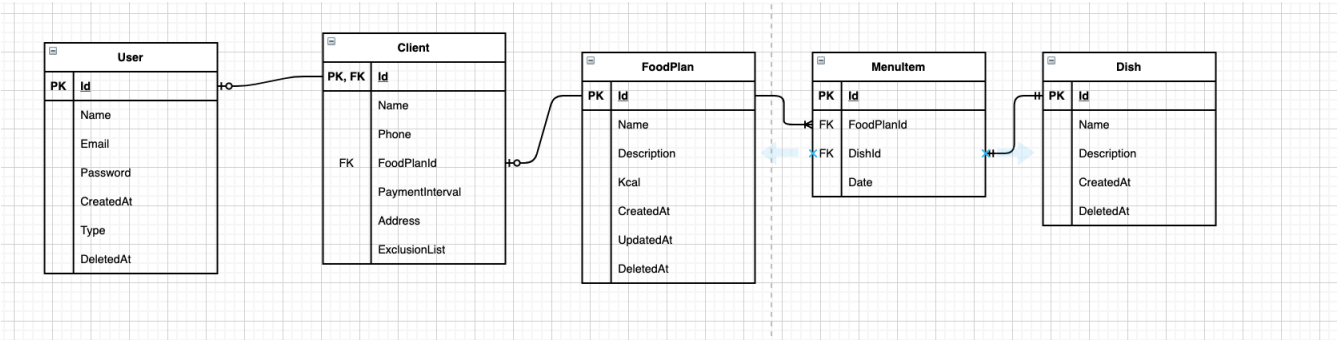


Рис. 3.8. ERD структура бази даних вебдодатку

3.3.7. *Висновки*

В результаті проєктування структури таблиць БД отримано набір сутностей та окреслено зв’язки між ними.

3.4. Створення нового плану харчування

Однією з функцій розроблювального застосунку для управління замовленнями їжі за підпискою є можливість створення та редагування планів харчування. Для цього користувач повинен мати тип адміністратор та бути авторизованим в системі.

Рис. 3.9. Алгоритм створення нового плану харчування. Діаграма діяльності

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Реалізація розгортання та тестування вебдодатку

Налаштування розгортання програмного коду стає все актуальнішою проблемою адже складність IT-систем зростає з кожним роком й кількість модулів та сервісів все збільшується. Різні проєкти потребують розгортки у різні середовища. Це може бути як віртуальний сервер так і магазин застосунків, або Microsoft Azure. Для налаштування процесу деплою коду використовуються різні CI/CD інструменти. Зазвичай цим займаються DevOps-спеціалісти.

Враховуючи обрану архітектуру додатку, було вирішено що в майбутньому розгорнути застосунок можна за допомогою засобів та сервісів Microsoft Azure. Також це рішення обумовлене важливістю підтримувати низькі експлуатаційні витрати. Сервіси Azure та база даних SQL Azure дбають про доступність, масштаб, безпеку та управління інфраструктурою ваших додатків, дозволяючи витрачати більше часу на зростання бізнесу та розширення прав і можливостей співробітників.

Для вебдодатків ASP.NET Azure пропонує єдину наскрізну повністю керовану платформу, яка спочатку підтримує Windows, пропонуючи неперевершену продуктивність розробників завдяки глибокій інтеграції Visual Studio та GitHub і спираючись на 25 років інновацій SQL разом із базою даних SQL Azure. Завдяки безкоштовним ресурсам хмарної міграції та таким інструментам, як Azure Migrate, спеціально розроблений для вебпрограм .NET та базам даних SQL, які їх живлять, перехід до хмарних технологій стає простішим.

Засоби моніторингу і аналізу дозволяють оптимізувати ресурси, щоб можна було отримати більше можливостей хмари з меншими витратами. Проста віддалена і інтерактивна відладка сайтів дозволяє діагностувати проблеми і швидко їх усувати.

Принцип безперервної інтеграції (CI) наполягає на автоматизованій інтеграції змін коду проєкту від кількох учасників [18]. Це дозволяє

програмістам часто об’єднувати зміни внесені в код проєкту в центральне сховище, де потім виконуються збірки та тести. Автоматизовані інструменти використовуються для підтвердження правильності нового коду перед інтеграцією.

Безперервна доставка – це підхід, коли команди випускають якісні продукти часто та передбачувано із сховища вихідного коду до робочого середовища автоматизованим способом.

4.2. Опис користувацького інтерфейсу

Користувацький інтерфейс представляє собою SPA (односторінковий додаток), що має такі сторінки:

1. головна сторінка додатку;
2. сторінка зареєстрованого користувача-клієнта;
3. сторінка зареєстрованого користувача-адміністратора;
4. сторінка оформлення підписки;
5. сторінка створення/редагування планів харчування;
6. сторінка редагування профілю користувача-клієнта;
7. сторінка перегляду кількості страв що необхідно приготувати на наступний день;
8. сторінка перегляду наборів страв та адрес куда треба їх доставити;
9. сторінка редагування меню на тиждень.

Компоненти інтерфейсу створені з використанням Angualr Meterial.

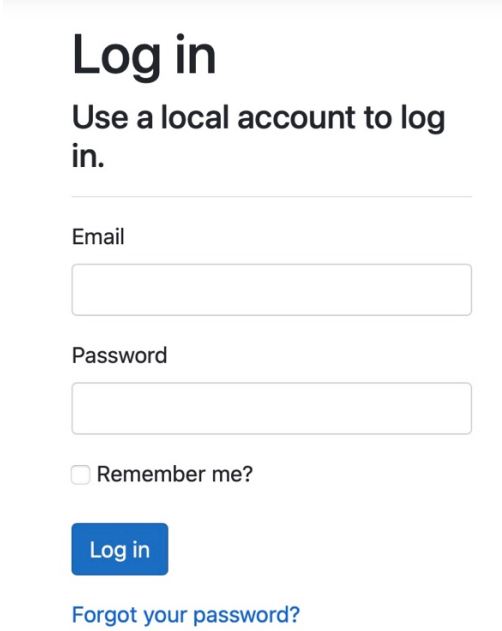


Рис. 4.1. Форма для авторизації користувача

На головній сторінці додатку розміщена інформація про доступні плани харчування. Для доступу до цієї сторінки не потрібна авторизація. Також з головної сторінки можна дістатися форми входу та реєстрації.

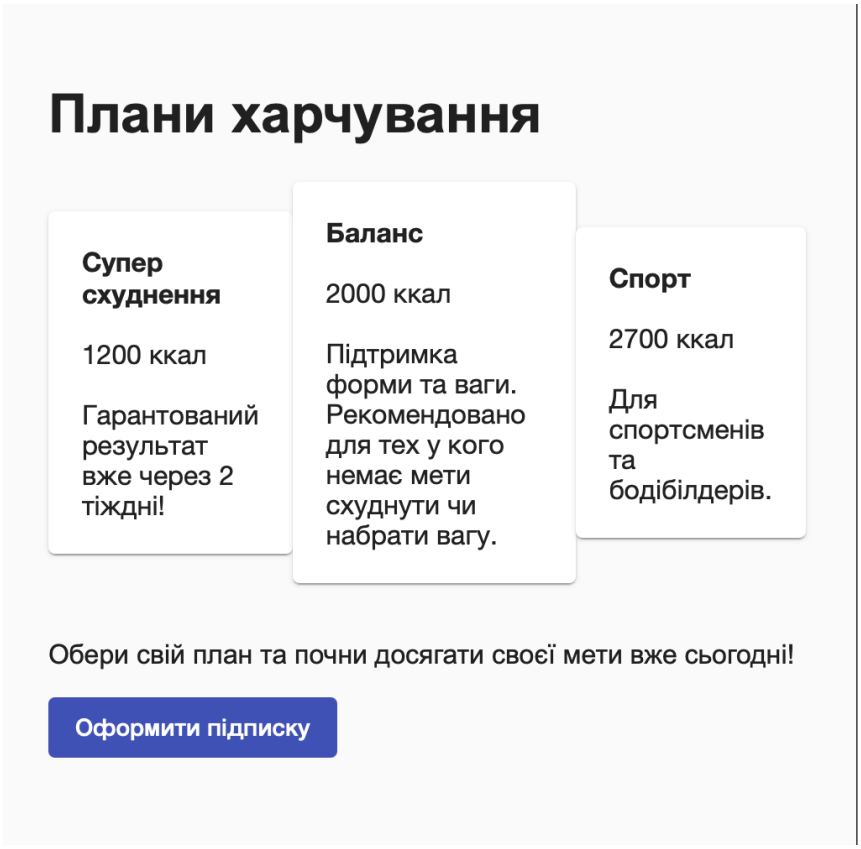


Рис. 4.2. Список доступних планів харчування на головній сторінці

Якщо зареєструватися через форму реєстрації, то для користувача буде створено аккаунт клієнта. З нього можна вибрати план харчування та оформити підписку заповнивши форму й вказавши в ній всі небхідні дані. Дана сторінка зображена на рис. 4.3.

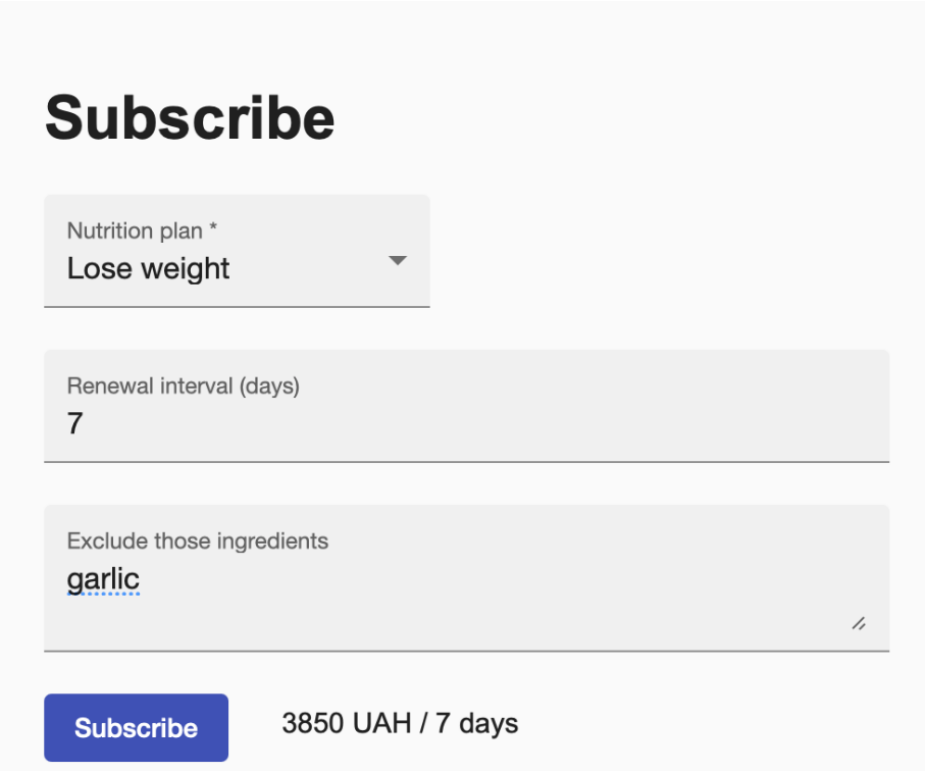


Рис. 4.3. Форма оформлення підписки

Якщо увійти в аккаунт адміністратора то відкриється сторінка адміністратора. На ній розташовані посилання на функції необхідні для обробки замовлень, такі як перегляд списку страв які необхідно приготувати та списку страв та вдрес клієнтів яким необхідно доставити їжу. Також на цій сторінці є посилання на сторінку редагування планів харчування.

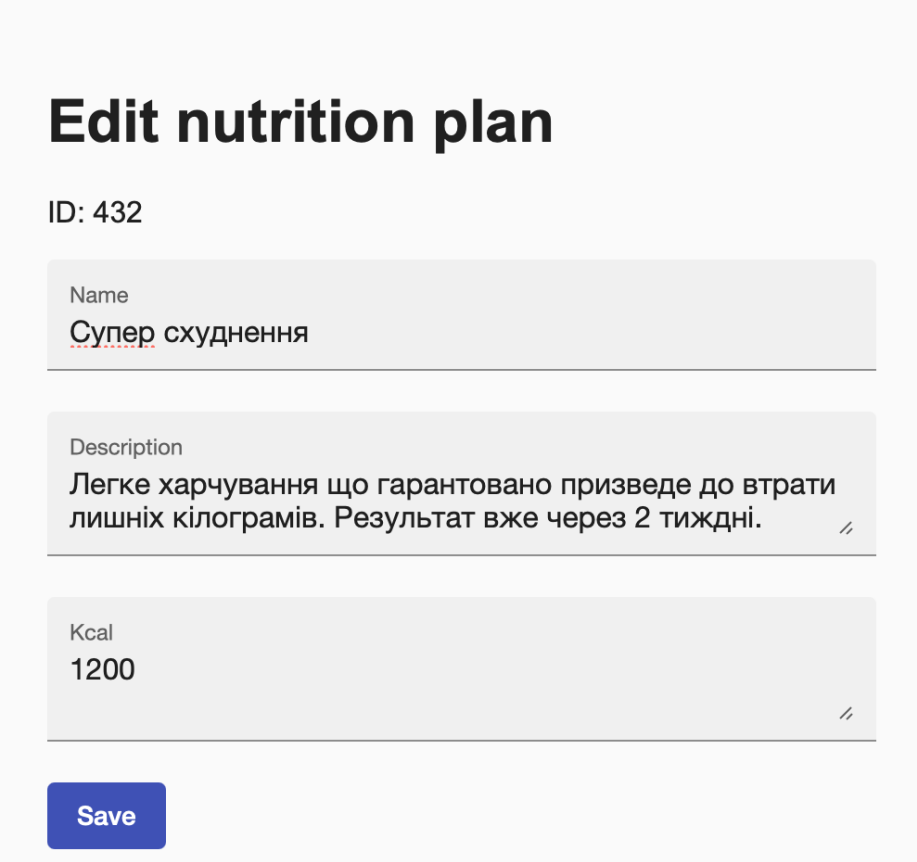


Рис. 4.4. Форма редагування планів харчування

4.3. Тестування вебзастосунку

Розроблене програмне забезпечення біло протестовано на тестових випадках з табл. 4.1.

Таблиця 4.1

Тестові випадки димового тестування

№	Мета тест-кейсу	Дія	Очікуваний результат
1	Перевірити чи відкривається сторінка додатку	Відкрити сторінку додатку в веббраузері	Браузер відображує головну сторінку
2	Перевірити можливість створити нового користувача	• Ввести дані у форму, • Натиснути на кнопку «зареєструватися».	Профіль користувача створено. Тепер доступна сторінка оформлення замовлення.

Продовження табл. 4.1

3	Перевірити можливість увійти в профіль	• Натиснути на кнопку «увійти». • Ввести дані у форму. • Натиснути на кнопку «увійти».	Користувач входить у свій профіль. Тепер доступна сторінка оформлення замовлення (для клієнтів) чи сторінка адміністратора для відповідних користувачів.
4	Перевірити можливість відновити пароль	• Натиснути на кнопку «увійти». • Ввести дані у форму. • Натиснути на кнопку «забув пароль». • Відкрити лист, що прийшов на вказану електрону адресу. • Перейти за посиланням в ньому. • Змінити пароль.	Після цього очікуйте лист на електронну пошту з посиланням на сторінку зміни пароля.
5	Оформити замовлення	• Авторизуватися як користувач-клієнт. • Натиснути «оформити підписку». Заповнити форму та натиснути «Оформити».	На екрані з’явиться текст «підписка оформлена успішно». Тепер треба зайти під користувачем-адміністратором та на сторінці адміністратора натиснути «Список для приготування» й відкриється список з стравами для обраного клієнтом плану. Також якщо під адміністратором зайти в розділ «Доставка» то з’явиться інформація які страви треба доставити клієнту.

Продовження табл. 4.1

6	Відредагувати доступний план харчування	• Авторизуватися як адміністратор. • Натиснути «Плани харчування». • Обрати план. • Внести зміни . • Зберегти.	На головній сторінці відображаються оновлені плани харчування.

4.4. Рекомендації щодо подальшого вдосконалення

Реалізований додаток не покриває 100% бажаного пакету можливостей, тому створено список рекомендованих функцій для подальшого розвитку та покращення застосунку.

Рекомендовані покращення:

- можливість реєстрації під час оформлення підписки – це значно спростить взаємодію клієнта з додатком, насамперед для нових клієнтів;
- додання можливості управління доставкою, призначення замовленню кур’єра, трекінг геолокації кур’єрів;
- додання модулю аналітики;
- додання можливості інтеграції з іншими системами, наприклад системами управління доставкою.

ВИСНОВКИ

Метою даного проєкту було створення вебдодатку, який надасть компаніям, що займаються доставкою здорової їжі за підпискою зручне комплексне рішення, яке задовольнить їх потреби у специфічному для цього виду бізнесу можливостях. Результатом даного проєкту став вебдодаток для управління замовленнями їжі за підпискою.

Після аналізу проблем, які наявні в предметні області та огляду існуючих аналогів, було встановлено, що розроблення вебдодатку для управління замовленнями їжі за підпискою є актуальним на даний момент. Проаналізовано наявні на ринку інструменти для розроблення ПЗ. В результаті чого обрано мову програмування Typescript та вебфреймворк Angular для розроблення клієнтської частини, мову програмування C# на основі платформи .NET для реалізації бекенду, як СУБД обрано Microsoft SQL Server.

Додаток був повністю розроблений та протестований згідно тест-ситуаціям описаним у розділі тестування.

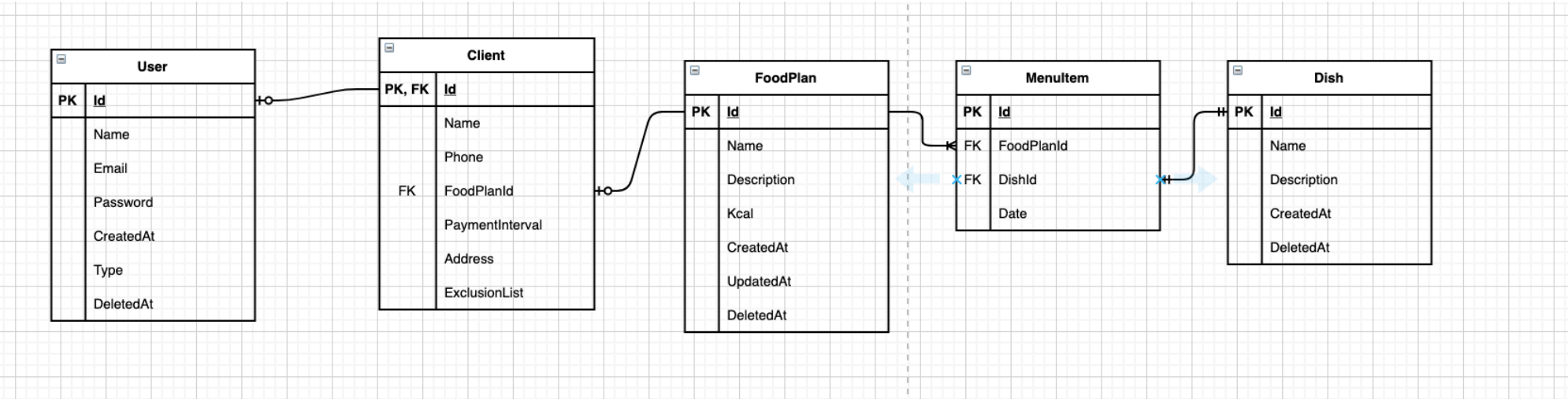
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Object-oriented programming [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Object-oriented_programming
2. SQL injection [Електронний ресурс]. — Режим доступу: https://portswigger.net/web-security/sql-injection
3. Cross-site scripting [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Cross-site_scripting
4. ACID [Електронний ресурс]. — Режим доступу: https://database.guide/what-is-acid-in-databases/
5. CDN [Електронний ресурс]. — Режим доступу: https://www.cloudflare.com/en-gb/learning/cdn/what-is-a-cdn/
6. Python [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language)
7. Comparison of 10 Programming Languages [Електронний ресурс]. — Режим доступу: https://reubenrochesingh.medium.com/comparison-of-10-programming-languages-f43b0ac337a4
8. Java [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Java_(programming_language)
9. Java Features [Електронний ресурс]. — Режим доступу: https://www.tutorialspoint.com/What-are-the-major-features-of-Java-programming
10. JavaScript [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/JavaScript
11. Node.js [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Node.js
12. TypeScript – JavaScript that scales [Електронний ресурс]. — Режим доступу: https://typescriptlang.org
13. The 5 Most Popular Front-end Frameworks Compared [Електронний ресурс]. — Режим доступу: https://www.sitepoint.com/most-popular-frontend-frameworks-compared/

14. React [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/React_(JavaScript_library)
15. Система управления базами данных [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/Система_управления_базами_данны
16. MySQL [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/MySQL
17. PostgreSQL [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/PostgreSQL
18. Continuous Integration Essentials [Електронний ресурс]. — Режим доступу: https://codeship.com/continuous-integration-essentials
19. NoSQL [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/NoSQL
20. MySQL [Електронний ресурс]. — Режим доступу: https://mysql.com
21. PostgreSQL [Електронний ресурс]. — Режим доступу: https://www.postgresql.org

ДОДАТКИ

Додаток 1
Копії графічних матеріалів



ДП.045440-06-99
Система управління замовленнями страв за підпискою.
Структура таблиць БД. ERD діаграма

ДП.045440-07-99
Система управління замовленнями страв за підпискою.
Алгоритм додання нового плану харчування.
Діаграма діяльності

Додаток 2
Лістинг програми

Startup.cs

```
using FoodSubscription.Application;
using FoodSubscription.Application.Common.Interfaces;
using FoodSubscription.Infrastructure;
using FoodSubscription.Infrastructure.Persistence;
using FoodSubscription.WebUI.Filters;
using FoodSubscription.WebUI.Services;
using FluentValidation.AspNetCore;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.SpaServices.AngularCli;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using NSwag;
using NSwag.Generation.Processors.Security;
using System.Linq;

namespace FoodSubscription.WebUI
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }
```

```
        // This method gets called by the runtime. Use this method to add
services to the container.

        public void ConfigureServices(IServiceCollection services)

        {

            services.AddApplication();

            services.AddInfrastructure(Configuration);

            services.AddDatabaseDeveloperPageExceptionFilter();

            services.AddSingleton<ICurrentUserService, CurrentUserService>();

            services.AddHttpContextAccessor();

            services.AddHealthChecks()

                .AddDbContextCheck<ApplicationDbContext>();

            services.AddControllersWithViews(options =>

                options.Filters.Add<ApiExceptionFilterAttribute>())

                    .AddFluentValidation();

            services.AddRazorPages();

            // Customise default API behaviour

            services.Configure<ApiBehaviorOptions>(options =>

            {

                options.SuppressModelStateInvalidFilter = true;

            });

            // In production, the Angular files will be served from this
directory

            services.AddSpaStaticFiles(configuration =>

            {

                configuration.RootPath = "ClientApp/dist";

            });

            services.AddOpenApiDocument(configure =>

            {

                configure.Title = "FoodSubscription API";
```

```
                configure.AddSecurity("JWT", Enumerable.Empty<string>(), new
OpenApiSecurityScheme

                {

                    Type = OpenApiSecuritySchemeType.ApiKey,

                    Name = "Authorization",

                    In = OpenApiSecurityApiKeyLocation.Header,

                    Description = "Type into the textbox: Bearer {your JWT
token}."

                });

                configure.OperationProcessors.Add(new
AspNetCoreOperationSecurityScopeProcessor("JWT"));

            });

        }

        // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.

        public void Configure(IApplicationBuilder app, IWebHostEnvironment
env)

        {

            if (env.IsDevelopment()) {

                app.UseDeveloperExceptionPage();

                app.UseMigrationsEndPoint();

            } else

            {

                app.UseExceptionHandler("/Error");

                // The default HSTS value is 30 days. You may want to change
this for production scenarios, see https://aka.ms/aspnetcore-hsts.

                app.UseHsts();

            }

            app.UseHealthChecks("/health");

            app.UseHttpsRedirection();

            app.UseStaticFiles();
```

```
            if (!env.IsDevelopment())
            {
                app.UseSpaStaticFiles();
            }
            app.UseSwaggerUi3(settings =>
            {
                settings.Path = "/api";
                settings.DocumentPath = "/api/specification.json";
            });
            app.UseRouting();
            app.UseAuthentication();
            app.UseIdentityServer();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller}/{action=Index}/{id?}");
                endpoints.MapRazorPages();
            });
            app.UseSpa(spa =>
            {
                spa.Options.SourcePath = "ClientApp";
                if (env.IsDevelopment())
                {
                    spa.UseAngularCliServer(npmScript: "start");
//spa.UseProxyToSpaDevelopmentServer("http://localhost:4200");
                }
```

```
            });
        }
    }
}
```

Program.cs

```
using FoodSubscription.Infrastructure.Identity;
using FoodSubscription.Infrastructure.Persistence;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Threading.Tasks;

namespace FoodSubscription.WebUI
{
    public class Program
    {
        public async static Task Main(string[] args)
        {
            var host = CreateHostBuilder(args).Build();
            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
```

```
                    var context =
services.GetRequiredService<ApplicationDbContext>();

                    if (context.Database.IsSqlServer())

                    {

                        context.Database.Migrate();

                    }

                    var userManager =
services.GetRequiredService<UserManager<ApplicationUser>>();

                    var roleManager =
services.GetRequiredService<RoleManager<IdentityRole>>();

                    await
ApplicationDbContextSeed.SeedDefaultUserAsync(userManager, roleManager);

                    await
ApplicationDbContextSeed.SeedSampleDataAsync(context);

                }

                catch (Exception ex)

                {

                    var logger =
scope.ServiceProvider.GetRequiredService<ILogger<Program>>();

                    logger.LogError(ex, "An error occurred while migrating or
seeding the database.");

                    throw;

                }

            }

            await host.RunAsync();

        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>

            Host.CreateDefaultBuilder(args)

                .ConfigureWebHostDefaults(webBuilder => {

                    webBuilder.UseStartup<Startup>();

                });
```

```
}
```

ApplicationDbContext.cs

```
using FoodSubscription.Application.Common.Interfaces;

using FoodSubscription.Domain.Common;

using FoodSubscription.Domain.Entities;

using FoodSubscription.Infrastructure.Identity;

using IdentityServer4.EntityFramework.Options;

using Microsoft.AspNetCore.ApiAuthorization.IdentityServer;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Options;

using System.Linq;

using System.Reflection;

using System.Threading;

using System.Threading.Tasks;

namespace FoodSubscription.Infrastructure.Persistence

{

    public class ApplicationDbContext :
ApiAuthorizationDbContext<ApplicationUser>, IApplicationDbContext

    {

        private readonly ICurrentUserService _currentUserService;

        private readonly IDateTime _dateTime;

        private readonly IDomainEventService _domainEventService;

        public ApplicationDbContext(

            DbContextOptions options,

            IOptions<OperationalStoreOptions> operationalStoreOptions,

            ICurrentUserService currentUserService,

            IDomainEventService domainEventService,

            IDateTime dateTime) : base(options, operationalStoreOptions)

        {
```

```
            _currentUserService = currentUserService;

            _domainEventService = domainEventService;

            _dateTime = dateTime;

        }

        public DbSet<User> Users { get; set; }

        public DbSet<Client> Clients { get; set; }

        public DbSet<FoodPlan> FoodPlans { get; set; }

        public DbSet<Dish> Dishes { get; set; }

        public DbSet<MenuItem> MenuItems { get; set; }

        public override async Task<int> SaveChangesAsync(CancellationToken
cancellationToken = new CancellationToken())

        {

            foreach
(Microsoft.EntityFrameworkCore.ChangeTracking.EntityEntry<AuditableEntity>
entry in ChangeTracker.Entries<AuditableEntity>())

            {

                switch (entry.State)

                {

                    case EntityState.Added:

                        entry.Entity.CreatedBy = _currentUserService.UserId;

                        entry.Entity.Created = _dateTime.Now;

                        break;



                    case EntityState.Modified:

                        entry.Entity.LastModifiedBy =
_currentUserService.UserId;

                        entry.Entity.LastModified = _dateTime.Now;

                        break;

                }

            }

            var result = await base.SaveChangesAsync(cancellationToken);
```

```
            await DispatchEvents();

            return result;

        }

        protected override void OnModelCreating(ModelBuilder builder)

        {

builder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());

            base.OnModelCreating(builder);

        }

        private async Task DispatchEvents()

        {

            while (true)

            {

                var domainEventEntity =
ChangeTracker.Entries<IHasDomainEvent>()

                    .Select(x => x.Entity.DomainEvents)

                    .SelectMany(x => x)

                    .Where(domainEvent => !domainEvent.IsPublished)

                    .FirstOrDefault();

                if (domainEventEntity == null) break;

                domainEventEntity.IsPublished = true;

                await _domainEventService.Publish(domainEventEntity);

            }

        }

    }

}
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Система управління замовленнями їжі за підпискою

Виконав: Теплицький Святослав Вікторович

Керівник: ст. викладач кафедри ПЗКС, к.т.н. Рибачок Наталія Антонівна

Київ – 2021

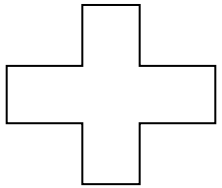
Актуальність

- Згідно з опитуванням, **59%** респондентів в тій чи іншій мірі стежать за харчуванням (2017 рік, опитування в одній з країн бувшого СРСР).
- Лише у Києві існує близько **20 компаній** що надають сервіс доставки здорової їжі за підпискою й кількість клієнтів цих компаній щорічно зростає. Також такі компанії працюють в багатьох великих містах України.
- Зі зростанням кількості клієнтів постає питання впровадження інформаційних систем для спрощення ведення бізнесу.

Актуальність

Бізнес модель

Доставка інтернет-замовлень з ресторану

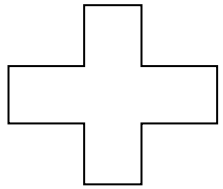


Доставка продуктів за підпискою (наприклад друкованих видань)

Аналоги

Інформаційні системи

Доставка інтернет-замовлень з ресторану



Доставка продуктів за підпискою (наприклад друкованих видань)

GloriaFood

Poster

flipdish

SUBSCRIPTION FLOW

SUBSCRIPTION DNA

сбис PRESTO

Аналоги

Система управління
замовленнями їжі за підпискою

Постановка задачі

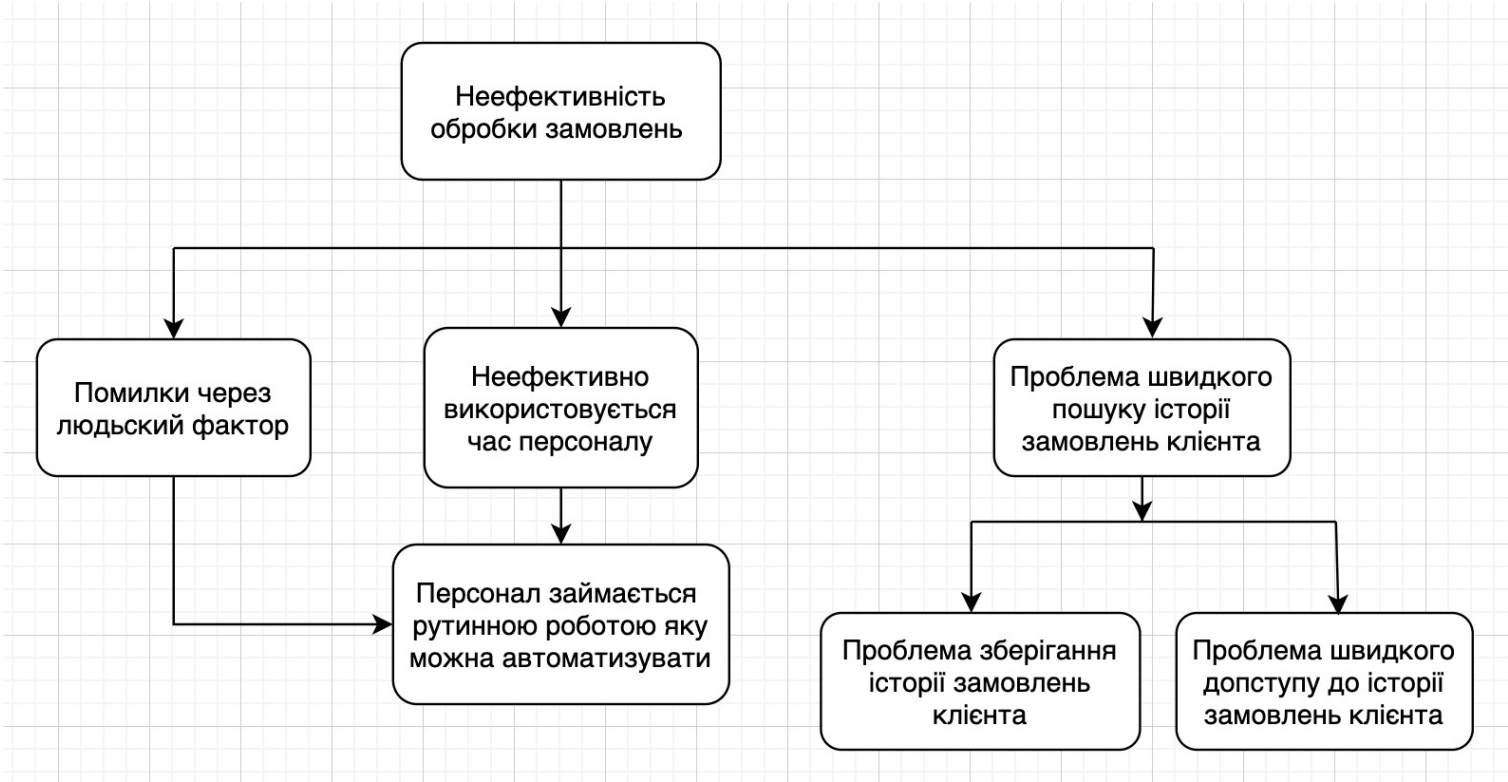
Мета проекту: створити інформаційну систему для бізнесу який надає сервіс доставки здорового харчування за підпискою

Постановка задачі

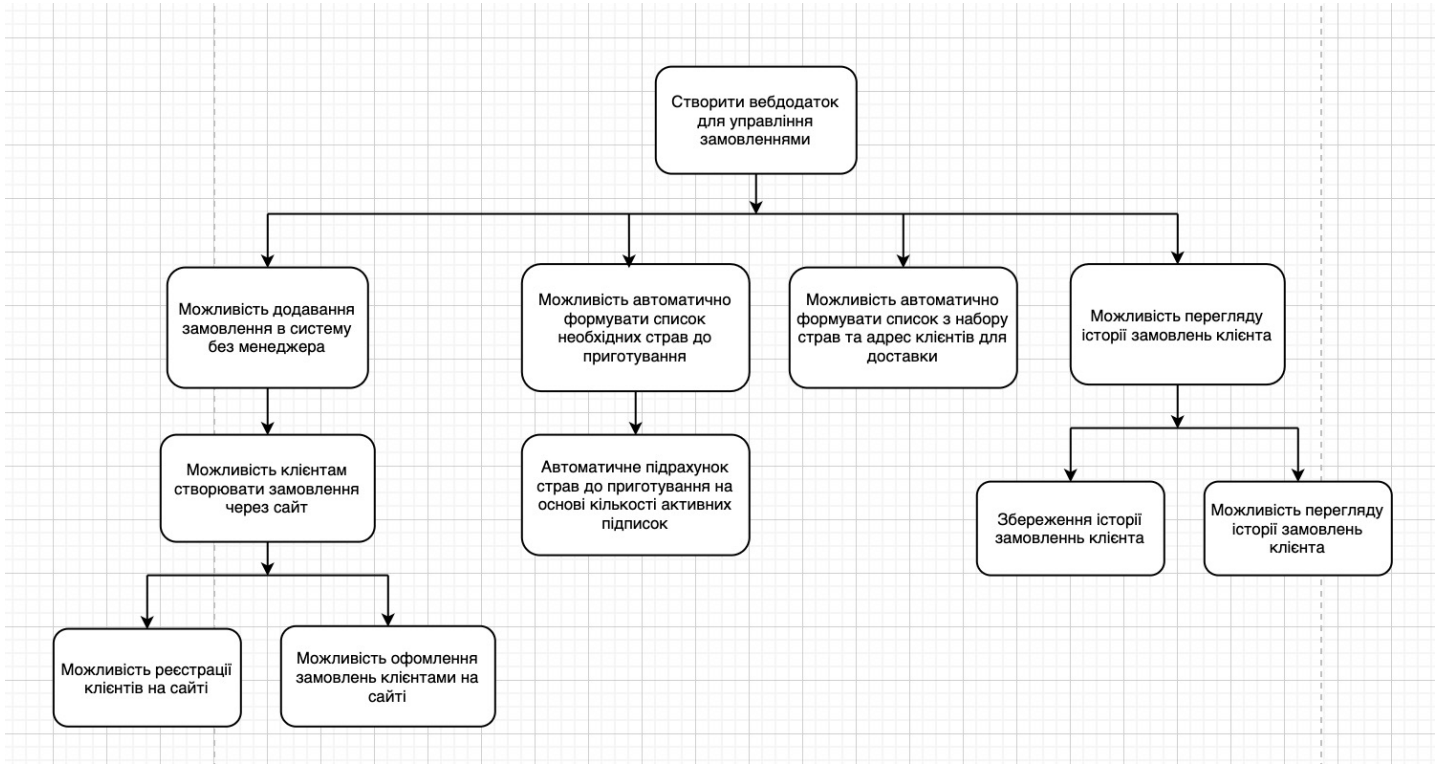
ЗАВДАННЯ:

1. Дослідити проблеми, з якими зтикаються компанії що надають сервіс доставки їжі за підпискою при організації бізнес процесів.
2. Проаналізувати існуючі програмні рішення.
3. Визначити вимоги до розроблюваного ПЗ.
4. Розробити веб-додаток у відповідності до сформованих вимог.
5. Провести тестування отриманого ПЗ та проаналізувати результати.
6. Знайти шляхи подальшого розвитку проєкту.

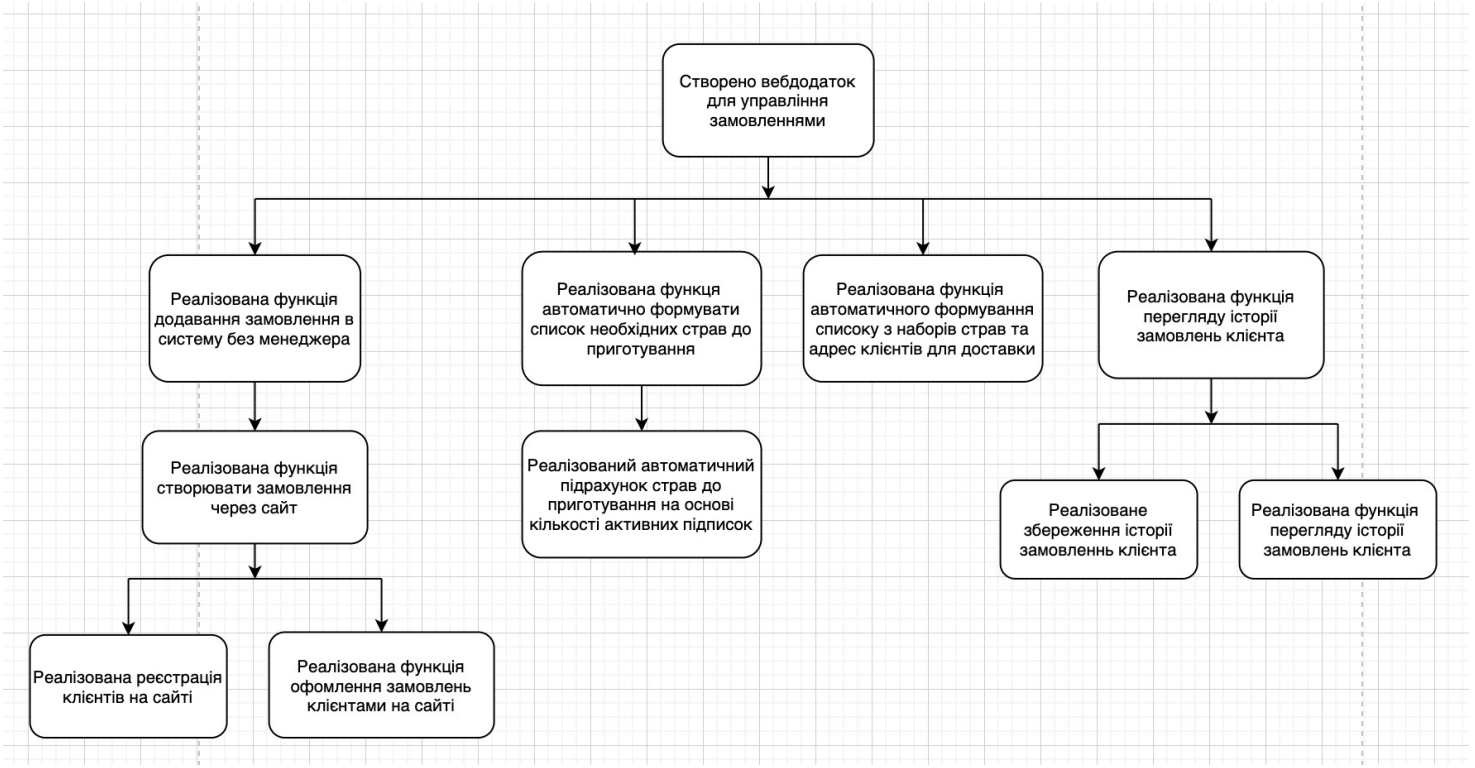
Дерево проблем



Дерево цілей

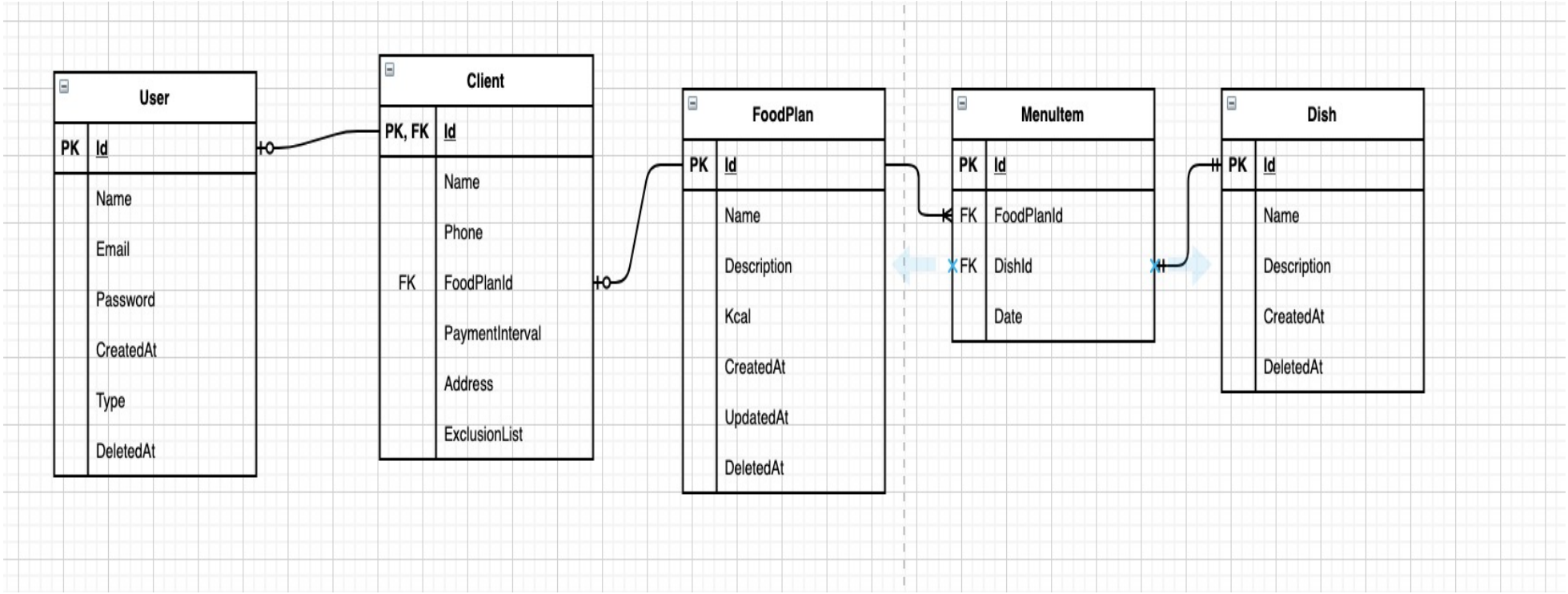


Дерево результатів

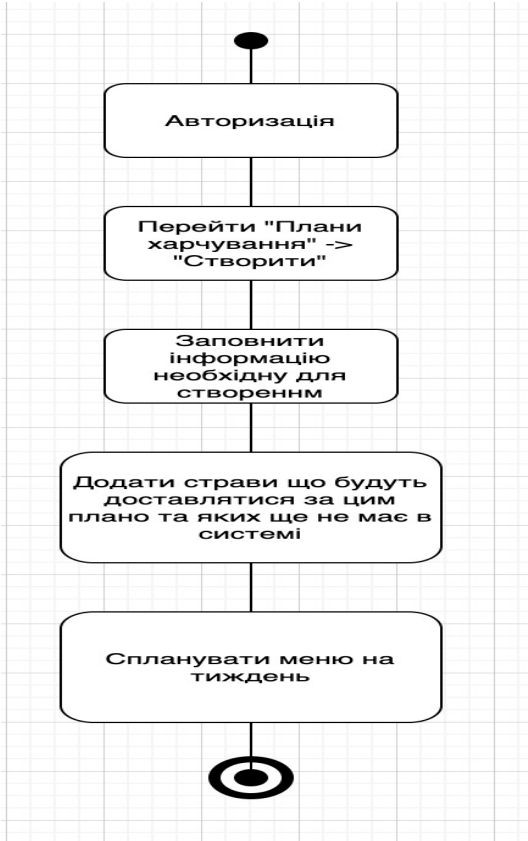


Вибір технологій розроблення

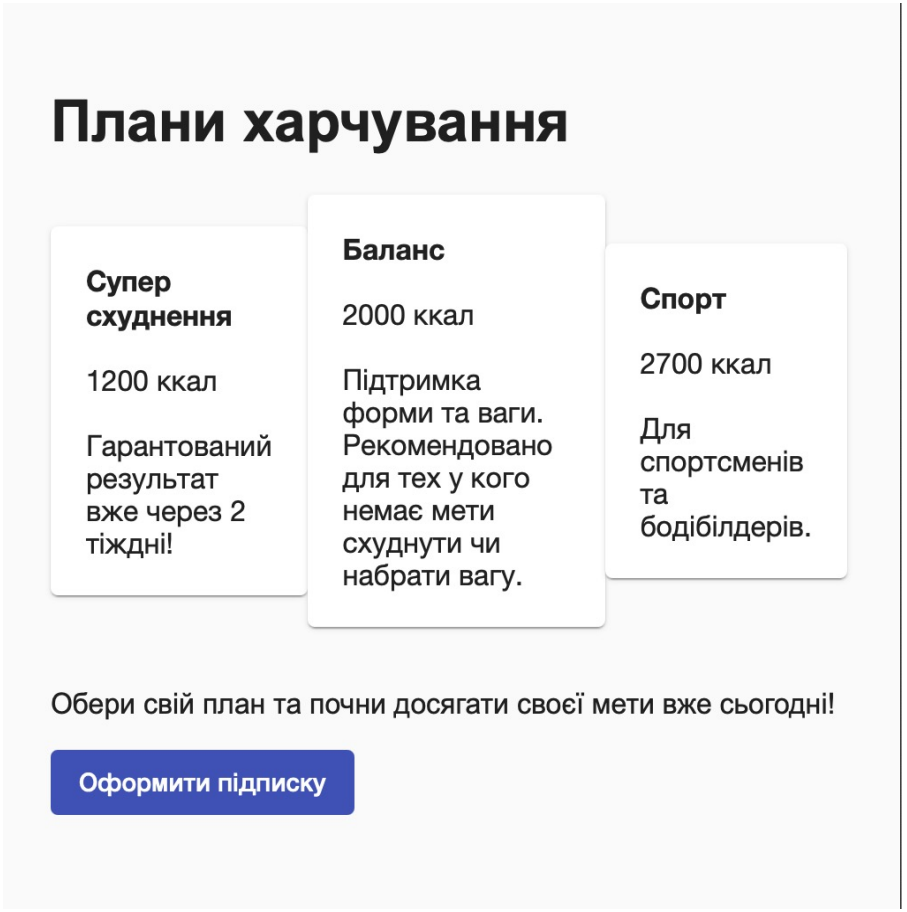
СТРУКТУРА БД. ERD ДІАГРАМА



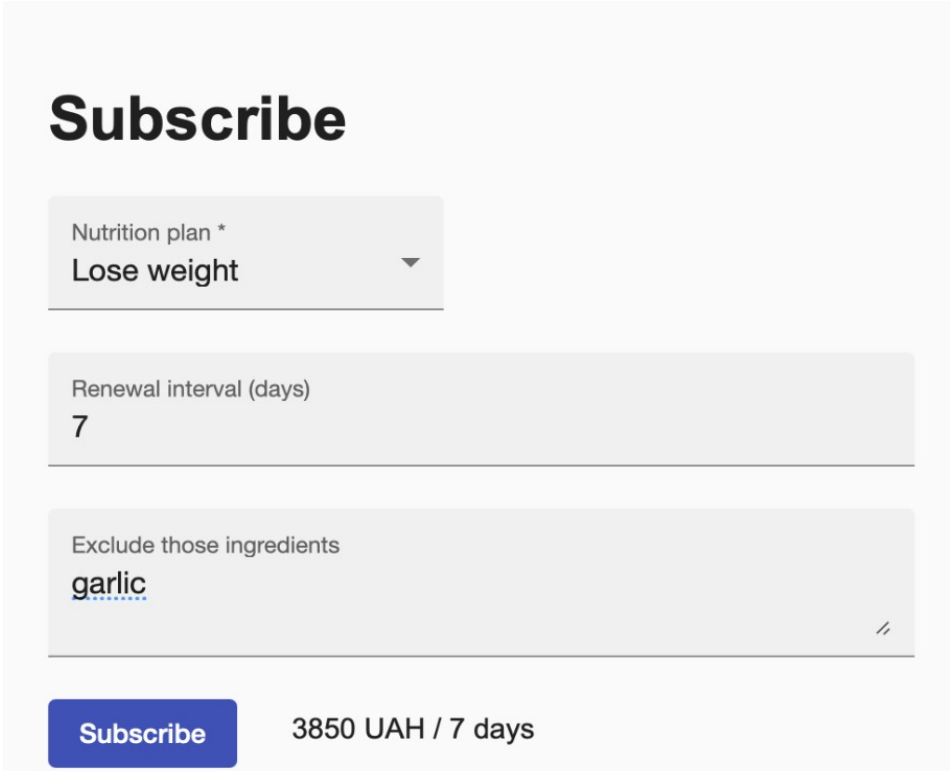
Додавання нового плану харчування. Діаграма діяльності



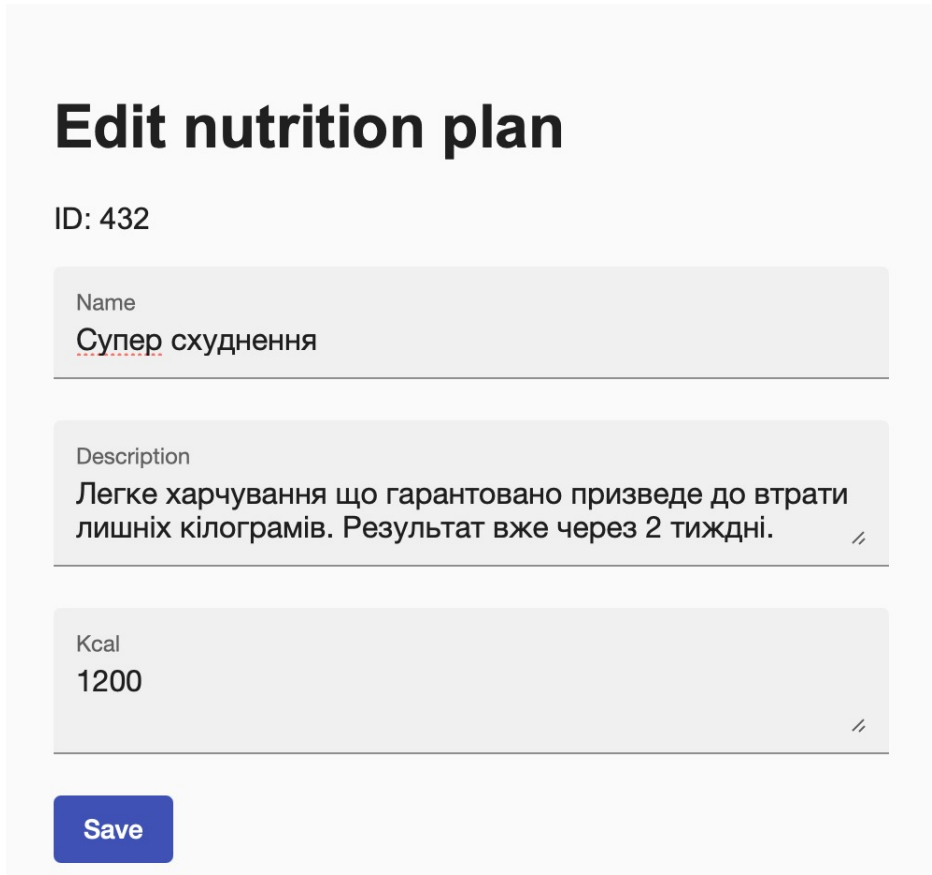
Інформаційна сторінка для клієнта



Форма оформлення підписки



Форма редагування плану харчування



ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ ПРОЄКТУ

- Додання можливості управління доставкою, призначення замовленню кур’єра, трекінг геолокації кур’єрів;
- Додання модулю аналітики;
- Додання можливості інтеграції з іншими системами, наприклад системами управління доставкою

Висновки

В процесі роботи над даним проєктом було зроблено:

1. Було досліджено проблеми, з якими стикаються компанії що надають сервіс замовлення доставки їжі за підпискою.
2. Знайдено та проаналізовано існуючі програмні рішення.
3. Сформовано і визначено вимоги до розроблюваного ПЗ.
4. Розроблено веб-додаток, який має базовий функціонал необхідний для організації бізнес-процесів компаній що доставляють їжу за підпискою.
5. Проведено тестування отриманого ПЗ та проаналізовано результати, на основі яких можна стверджувати, що розроблення виконано повністю та відповідає вимогам.
6. Знайдено шляхи подальшого розвитку проєкту.

Висновки

В рамках проєкту було **реалізовано**:

1. Вебінтерфейс користувача.
2. Клієнт-серверну архітектуру за допомогою технологій таких як .NET, ASP.NET, Angular та інших.
3. Можливість авторизації та реєстрації користувачів
4. Можливість створювати плани харчування та планувати які страви за тим чи іншим планом харчування будуть доставлені клієнтам в той чи інший день
5. Можливість переглядати, додавати, редагувати та видаляти страви які можуть бути доставлені клієнтам
6. Можливість переглядати список замовлень на сьогодні, змінювати статус замовлення

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

СИСТЕМА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ ЇЖІ ЗА ПІДПИСКОЮ

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Наталія РИБАЧОК

Виконавець:

_____ Святослав ТЕПЛИЦЬКИЙ

2020

ЗМІСТ

1. Об’єкт випробувань 3
2. Мета тестування 3
3. Методи тестування 3
4. Засоби та порядок тестування 4

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебдодаток для управління замовленнями доставки їжі за підпискою створений на платформі .NET та написаний на мовах програмування C# (серверна частина) та TypeScript (клієнтська частина) з використанням технології Angular та AJAX.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціональна працездатність елементів сторінок вебдодатку;
- наявність доступу до бази активних підписок;
- забезпечення належного рівня безпеки даних;
- зручність роботи з вебсайтом;
- відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) Функціональне тестування, зокрема на рівні інтеграційного тестування (тестування взаємодії різних модулів програми) та тестування інтерфейсу (тестування користувацької взаємодії).
2) Тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-ресурсу перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) тестування коду юніт тестами;
4) тестування вебдодатку в різних веббраузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2021 р.

СИСТЕМА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ ЇЖІ ЗА ПІДПИСКОЮ

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Наталія РИБАЧОК

Виконавець:

_____ Святослав ТЕПЛИЦЬКИЙ

2021

ЗМІСТ

1. Опис структури вебдодатку 3
2. Процедура авторизації користувача 4
3. Головна сторінка користувача-клієнта 4
4. Сторінка адміністратора 5
5. Сторінка створення плану харчування 6
6. Сторінка перегляду активних підписок 7
7. Сторінка перегляду активних підписок 8

1. Опис структури вебдодатку

Система управління замовленнями їжі за підпискою побудована у вигляді вебдодатку за моделлю SPA, тому усі сторінки вебдодатку є динамічними та генеруються в реальному часі на стороні клієнта за допомогою коду на мові програмування TypeScript та фреймворку Angular.

Динамічні сторінки вебдодатку представлені наступними вебсторінками:

- Головна сторінка додатку;
- Сторінка зареєстрованого користувача-клієнта;
- Сторінка зареєстрованого користувача-адміністратора;
- Сторінка оформлення підписки;
- Сторінка створення/редагування планів харчування;
- Сторінка редагування профілю користувача-клієнта;
- Сторінка перегляду кількості страв що необхідно приготувати на наступний день;
- Сторінка перегляду наборів страв та адрес куди треба їх доставити;
- Сторінка редагування меню на тиждень.

Кожна вебсторінка складається із таких блоків:

- шапка сайту;
- вміст сторінки;

В шапці компонентах знаходяться корисні посилання на інші сторінки вебдодатку.

2. Процедура авторизації користувача

Для авторизації користувачів треба натиснути «Login» в шапці сторінки. Після цього для користувача буде відкритою сторонка авторизації.

Сторінка авторизації містить такі поля: Email Address – електронна пошта користувача та Password – пароль користувача.

Для забезпечення безпеки даних відвідувача, пароль, який він вводить буде заховано за зірочками. Дане вікно містить посилання на форму відновлення паролю.

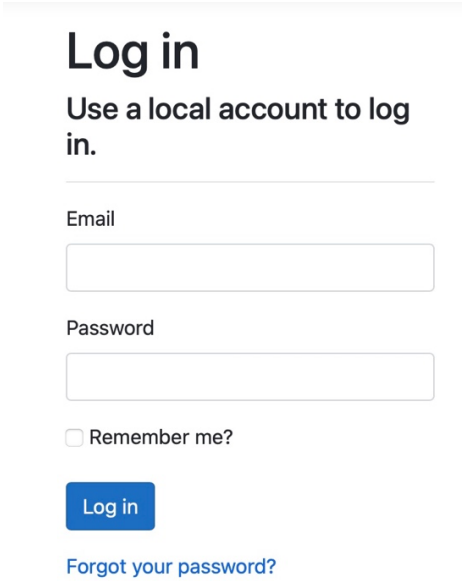


Рис. 1. Сторінка авторизації користувача

Після того як користувач увійде до свого аккаунту то в залежності від ролі користувача він буде перенаправлений на головну сторанку (якщо користувач має роль – Клієнт) чи сторінку адміністратора (якщо користувач має роль – Адміністратор).

3. Головна сторінка користувача-клієнта

На головній сторінці розміщений список усіх доступних планів харчування. Також на головній сторінці є посилання на форму оформлення підписки на обраний план харчування.

Для доступу до цієї сторінки не обов’язково потрібно бути авторизованим в системі. Також відкрити сторінку авторизації можна натиснувши на посилання «Login» у шапці сторінки.

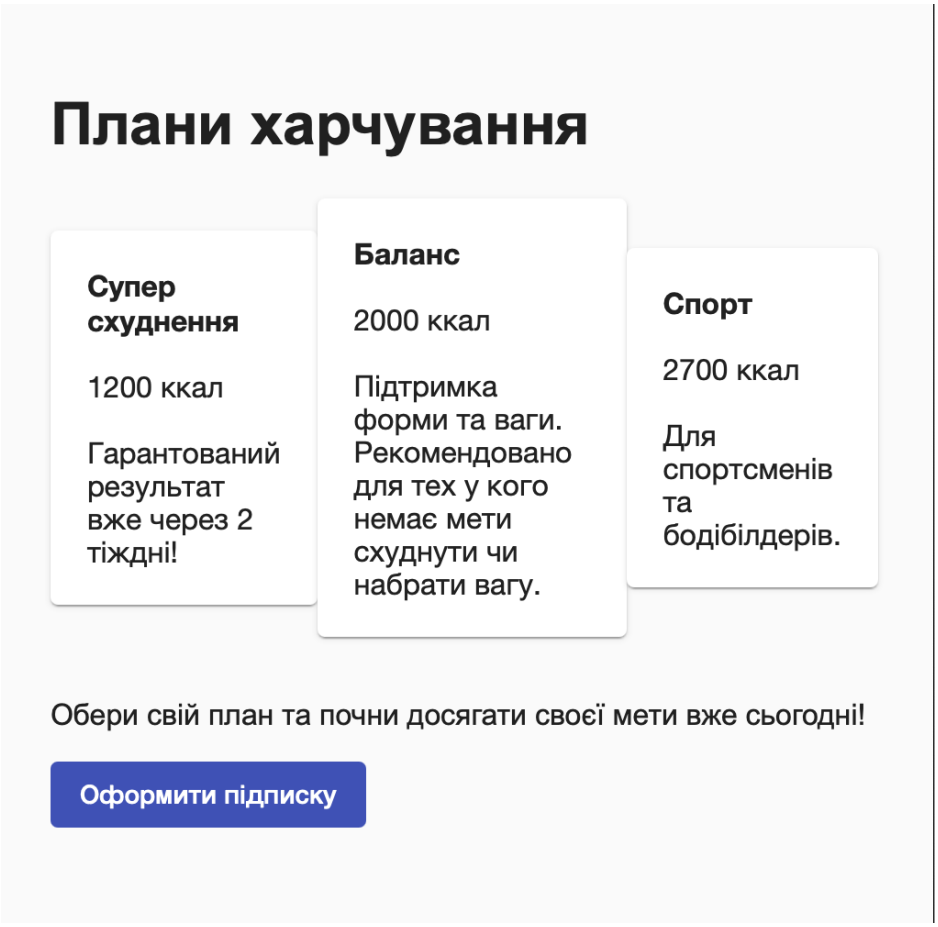


Рис. 2. Інформація про плани харчування на головній сторінці

4. Сторінка адміністратора

На сторінці адміністратора є посилання на основні функції системи управління, такі як редагування планів харчування, обновлення списку доступних страв, планування наборів страв для доставки у обрані дні та перегляд наборів страв та адрес клієнтів на які треба здійснити доставку.

На головній сторінці розміщений список усіх доступних планів харчування. Також на головній сторінці є посилання на форму оформлення підписки на обраний план харчування.

Для доступу до цієї сторінки потрібно бути авторизованим користувачем і також бути користувачем з роллю «адміністратор».

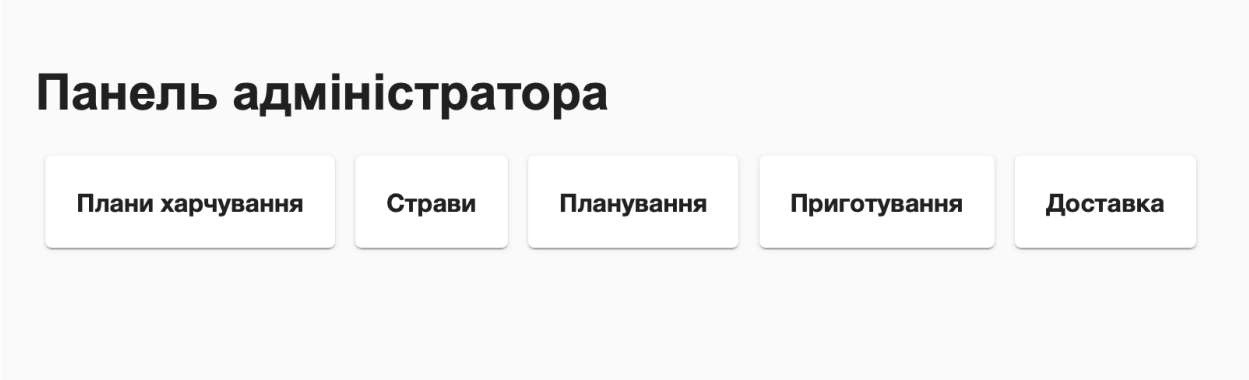


Рис. 3. Сторінка адміністратора

5. Сторінка оформлення підписки

На сторінці оформлення підписки знаходиться форма яку треба заповнити для того щоб успішно підписатися на обраний план доставки їжі.

Для доступу до цієї сторінки потрібно бути авторизованим користувачем з роллю «клієнт».

Форма оформлення підписки складається за таких полів: план харчування, інтервал оплати (кількість днів), список виключень.

Також на цій сторінці після заповнення форми даними буде підраховано та відображено суму першого платежу, яка буде складати кількість днів помножену на вартість денного раціону та враховуватиме знижки що додаються якщо клієнт обрав великий інтервал оплати (наприклад 30 днів та більше). Сума знижки не буде відображена але буде врахована у відображеній вартості.

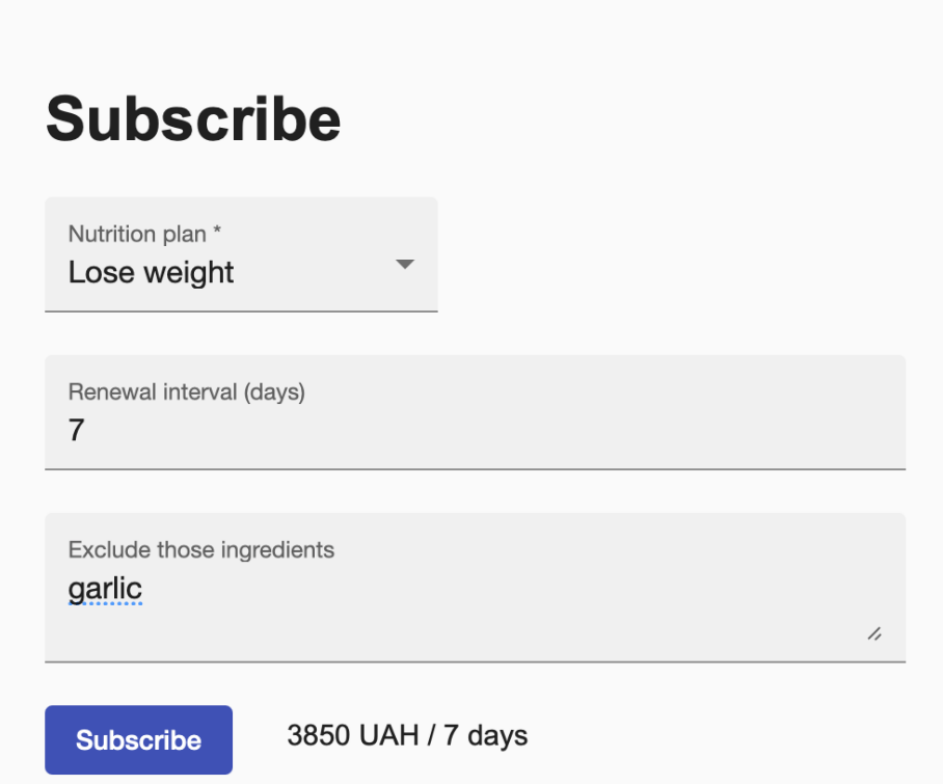


Рис. 4. Форма оформлення підписки

6. Сторінка редагування плану харчування

На сторінці редагування плану харчування розміщена форма. Для внесення змін потрібно відредагувати інформацію в формі та натиснути на кнопку «Зберегти».

Після збереження дані автоматично будуть оновлені.

Для доступу до цієї сторінки потрібно бути авторизованим користувачем з типом «адміністратор».

Форма редагування плану харчування складається з таких полів:

- Ідентифікатор (ID) – незмінний, автоматично призначається системою.
- Назва (Name).
- Опис (Description) – детальний опис плану харчування, може включати інформацяю які страви можуть входити.
- Кілокалорії на день (kcal) – середня кількість кілокалорій на день.

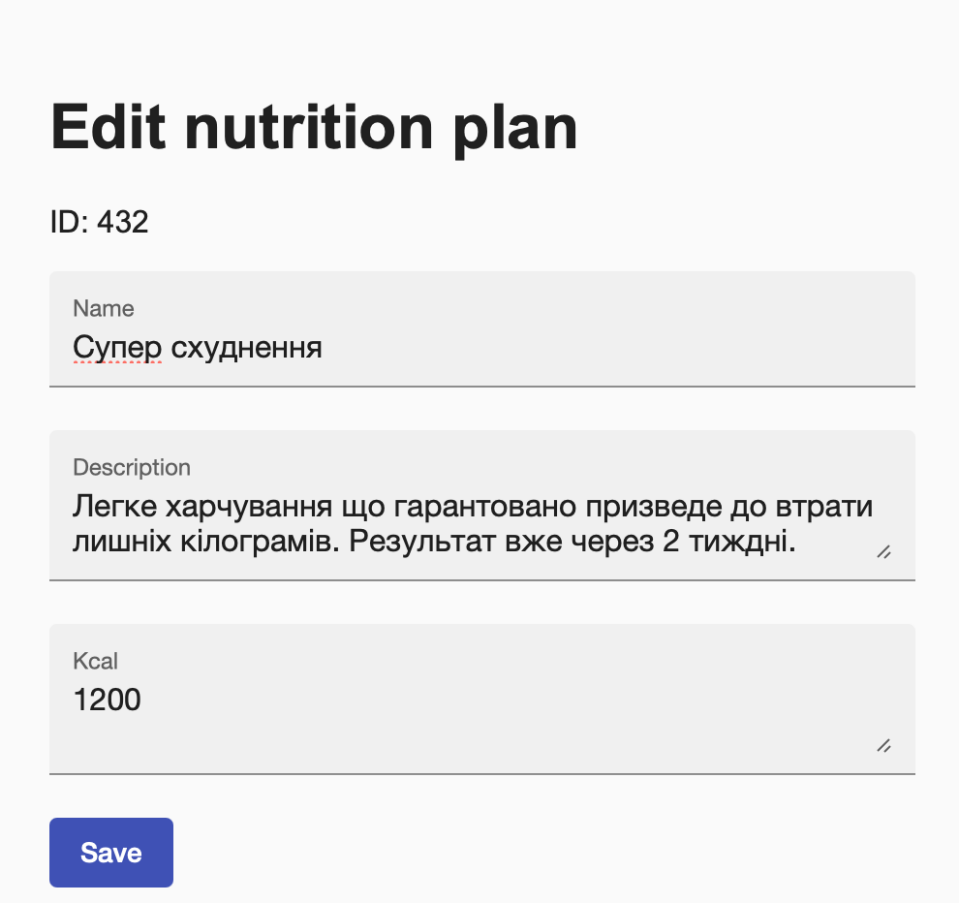


Рис. 5. Форма редагування плану харчування

7. Сторінка доступних страв

На сторінці доступних страв розміщена широка таблиця. Для того зоб внести зміни в список доступних страв треба відредагувати дані що знаходяться у цій таблиці та натиснути кнопку «Save».

Для доступу до цієї сторінки потрібно бути авторизованим користувачем з типом «адміністратор».

Кожна страва має такі поля:

a. Ідентифікатор (ID) – генерується системою автоматично, не можна редагувати.
b. Назва (Name).
c. Опис (Description) – опис страви, може включати інформацяю які інгредієнти входять.

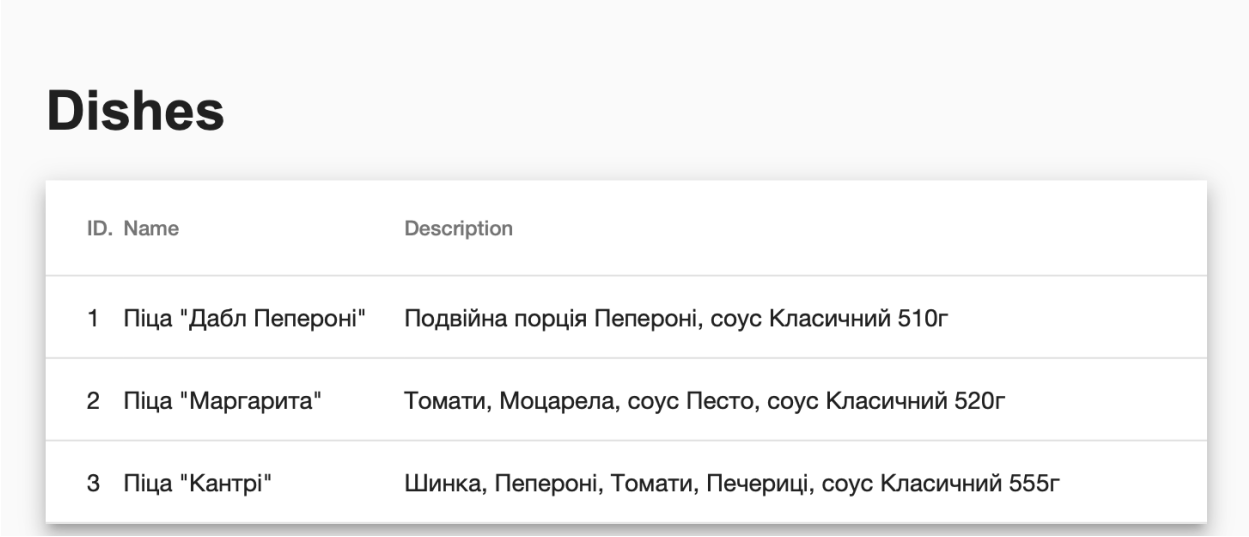


Рис. 6. Сторінка доступних страв