

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Веб-система обліку для кухні їдальні на
підприємстві»**

Виконав студент групи ЗПЗк-20-1
ГОЛУБ Роман Сергійович

Керівник роботи:
КУБРАК Юрій Олександрович

Консультант:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГОЛУБ Роман Сергійович**

Керівник роботи: **КУБРАК Юрій Олександрович**

Тема роботи: «Веб-система обліку для кухні їдальні на підприємстві»,
затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №71с

Вихідні дані для роботи: Об'єктом дослідження є процес роботи комірника на кухні комунального підприємства. Предметом дослідження є розробка веб-додатку для кухні їдальні, який покращить роботу не лише комірника, а й усього колектива кухні.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кубрак Ю.О., Єфремов Ю.М.	16.02.2023р.	16.02.2023р.
2	Кубрак Ю.О., Єфремов Ю.М.	16.02.2023р.	16.02.2023р.
3	Кубрак Ю.О., Єфремов Ю.М.	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів розробки	Примітка
1	Аналіз вимог та розробка концепції	16 лютого 2023 – 28 лютого 2023	Виконано
2	Проектування інтерфейсу та бази даних	1 березня 2023 – 15 березня 2023	Виконано
3	Реалізація функціональності та алгоритмів	16 березня 2023 – 3 квітня 2023	Виконано
4	Тестування та відладка	4 квітня 2023 – 12 квітня 2023	Виконано
5	Документування та підготовка звіту	11 травня 2023 – 2 червня 2023	Виконано
6	Попередній захист роботи	12 травня 2023	Виконано

РЕФЕРАТ

Випускна кваліфікаційна робота присвячена розробці веб-системи, що призначена для ведення обліку на кухні їдальні на підприємстві. Пояснювальна записка до випускної роботи містить 60 сторінок, 30 ілюстрацій та 2 таблиці.

Метою моєї роботи є розробка веб-системи обліку для кухні їдальні на підприємстві, що дасть можливість спростити комірнику кухні паперову роботу та перевести її у електронний вигляд. Основним завданням розроблення веб-сервісу є спрощення роботи комірника кухні з документами, що пов'язані зі створенням та оформленням меню, розрахунками порцій та веденням обліку продуктів.

У цій роботі я визначив головні завдання, пов'язані з розробкою веб-проекту. Я провів аналіз аналогічних систем, розроблених раніше, визначив основний функціонал і вимоги до системи. Крім того, я займався проектуванням і реалізацією алгоритмів, які використовує система, і провів тестування програмного продукту.

КЛЮЧОВІ СЛОВА: ВЕБ-СИСТЕМА, ПРОЕКТУВАННЯ, ПІДПРИЄМСТВО, ОПТИМІЗАЦІЯ РОБОТИ, МЕНЮ

					ІПЗ.КР.Б – 121 – 23 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Веб-система обліку для кухні їдальні на підприємстві Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Р.С. Голуб					4	60
Керівник		Ю.О. Кубрак						
н. контр.		О.П. Тичина				Житомирська політехніка, група ЗІПЗк-20-1		
Зав. каф.		А.В.Морозов						

ABSTRACT

The final qualification work is devoted to the development of a web-based system designed to keep records in the canteen kitchen at an enterprise. The explanatory note to the final work contains 60 pages, 30 illustrations and 2 tables.

The purpose of my work is to develop a web-based accounting system for the canteen kitchen at an enterprise, which will make it possible to simplify the kitchen storekeeper's paperwork and translate it into electronic form. The main task of developing a web service is to simplify the work of the kitchen storekeeper with documents related to the creation and design of menus, portion calculations, and product accounting.

In this paper, I have identified the main tasks related to the development of a web project. I analyzed similar systems developed earlier, identified the main functionality and requirements for the system. In addition, I designed and implemented the algorithms used by the system and tested the software product.

KEYWORDS: WEB SYSTEM, DESIGN, ENTERPRISE, WORK OPTIMIZATION, MENU

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБ-СИСТЕМИ ОБЛІКУ ДЛЯ КУХНІ ЇДІЛЬНІ	10
1.1 Постановка задачі	10
1.2 АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ	11
1.3 ВИБІР АРХІТЕКТУРИ ВЕБ-СЕРВІСУ	13
1.4 ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ.....	17
Висновки до розділу 1.....	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-СИСТЕМИ ДЛЯ КУХНІ ЇДАЛЬНІ НА ПІДПРИЄМСТВІ.....	25
2.1 Визначення варіантів використання та структури системи.....	25
2.2 Розробка бази даних системи.....	36
2.3 Опис взаємодії елементів системи під час роботи програмного комплексу	37
2.4 РЕАЛІЗАЦІЯ ВЕБ-СИСТЕМИ ОБЛІКУ ДЛЯ КУХНІ ЇДАЛЬНІ НА ПІДПРИЄМСТВІ.	38
Висновки до розділу 2.....	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ	42
3.1 Порядок встановлення та налаштування системи.	42
3.2 СТРУКТУРА ІНТЕРФЕЙСУ ВЕБ-СИСТЕМИ ОБЛІКУ ДЛЯ КУХНІ ЇДАЛЬНІ НА ПІДПРИЄМСТВІ	42
3.3 ТЕСТУВАННЯ РОБОТИ СИСТЕМИ	49
Висновки до розділу 3.....	54
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	58
ДОДАТКИ.....	61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

VS Code – Visual Studio Code.

ПК – Персональний комп'ютер.

TS – Typescript.

БД – База даних.

SOA – Service Oriented Architecture – сервіс орієнтована архітектура.

GUI – Graphic User Interface – графічний інтерфейс користувача.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						7

ВСТУП

Актуальність розроблення веб-системи обліку на кухні їдальні на підприємстві полягає у забезпеченні численних переваг та покращенні ефективності роботи комірника кухні. Ефективність роботи є нагальною проблемою в державних закладах, зокрема закладах освіти.

Практичне значення розробки веб-системи обліку на кухні їдальні полягає в:

1. Ефективному управлінні запасами: Розробка веб-додатку для кухні їдальні дозволяє автоматизувати процес розрахунку запасів, що значно пришвидшує шлях до моменту, коли потрібно поповнювати запаси продуктів на кухні
2. Оптимізації роботи їдальні: Завдяки додатку, який можна розширювати, додаючи нові модулі та оптимізуючи старі, можна добитись зменшення часу на роботу з паперами (або взагалі відмовитись від роботи з ними) та приділити увагу більше важливим своїм обов'язкам, таким як: сортування та регулярне перебирання овочів, забезпечення належного стану комор, дотримання норм витрат продуктів на кухні та видачі продуктів харчування працівникам харчоблоку за вагою [1].
3. Зменшенні ризиків, пов'язаних з людським фактором: Добре відтестована та налагоджена програма зменшить імовірність помилки, а якщо і помилка буде допущена, її буде легше знайти в електронному сховищі, ніж в паперовому.

Метою даної кваліфікаційної роботи є розробка веб-додатку для кухні їдальні, який надає ефективний та зручний інтерфейс для роботи з продуктами та звітами на кухні.

Головними завданнями роботи є:

- визначення функціональних вимог до додатку;
- ознайомлення з внутрішньою документацією підприємства;
- розробка алгоритмів створення меню, експорту;
- тестування та оптимізація роботи системи.

Об'єктом дослідження є процес роботи комірника на кухні комунального підприємства

Предметом дослідження є розробка веб-додатку для кухні їдальні, який покращить роботу не лише комірника, а й усього колектива кухні.

В процесі роботи використовувалися методи аналізу вимог користувачів. Інформаційна база дослідження включала знайомство з документацією підприємства, ознайомлення з робочими процесами комірника та отримання відгуків та пропозицій щодо пришвидшення його роботи.

Результати даної роботи можуть бути використані для розробки та впровадження на реальне підприємство веб-системи обліку на кухні.

Пояснювальна записка до кваліфікаційної роботи містить 60 сторінок, 30 ілюстрацій та 2 таблиці, список літературних джерел містить 30 найменувань.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБ-СИСТЕМИ ОБЛІКУ ДЛЯ КУХНІ ЇДАЛЬНІ

1.1 Постановка задачі

В рамках випускної роботи планується розробка програмного комплексу для веб-системи обліку на кухні їдальні на підприємстві. Основною метою роботи є спрощення робочих процесів та перехід від паперової документації до електронного формату. Для досягнення цієї мети необхідно вирішити наступні задачі:

1. Аналіз вимог та потреб користувачів: Вивчення поточного процесу обліку на кухні їдальні, спілкування з комірниками та іншими працівниками для виявлення їх потреб та вимог щодо функціональності та інтерфейсу програмного комплексу.

2. Проектування архітектури програмного комплексу: Визначення структури бази даних для зберігання інформації про замовлення, запаси, витрати та інші дані, які використовуються на кухні їдальні. Розробка модулів для обробки та аналізу цих даних з метою забезпечення ефективного обліку та контролю. Проектування веб-інтерфейсу, який буде зручним у використанні та забезпечуватиме швидкий доступ до необхідної інформації.

3. Реалізація функціональності програмного комплексу: Створення модулів для створення меню, розрахунків витрат, можливість збереження продуктів, даних про них, підрахунок витрат та експорт меню. Розробка механізмів для контролю запасів, включаючи оновлення кількості продуктів, виведення повідомлень про нестачу. Реалізація функцій обліку витрат, яка дозволить вести контроль над витратами продуктів та матеріалів.

4. Тестування та налагодження: Виконання ряду тестів для перевірки функціональності, стійкості та безпеки програмного комплексу. Виявлення та виправлення помилок, що можуть виникати під час роботи системи. Проведення тестування ефективності для перевірки швидкодії та відповідності вимогам продуктивності.

1.2 Аналіз аналогів програмного продукту

Важливість ефективної системи управління кухнею неможливо переоцінити в індустрії громадського харчування. Для того, щоб забезпечити безперебійну роботу закладу громадського харчування, важливо мати систему, яка відповідає унікальним потребам підприємства. Це пов'язано з тим, що традиційні аналоги на ринку не завжди мають необхідні функції, які вимагають державні підприємства.

Найбільш підходящим на роль веб-системи обліку на кухні може виступати Poster [2], інтерфейс програми зображено на рис. 1.1-1.3.

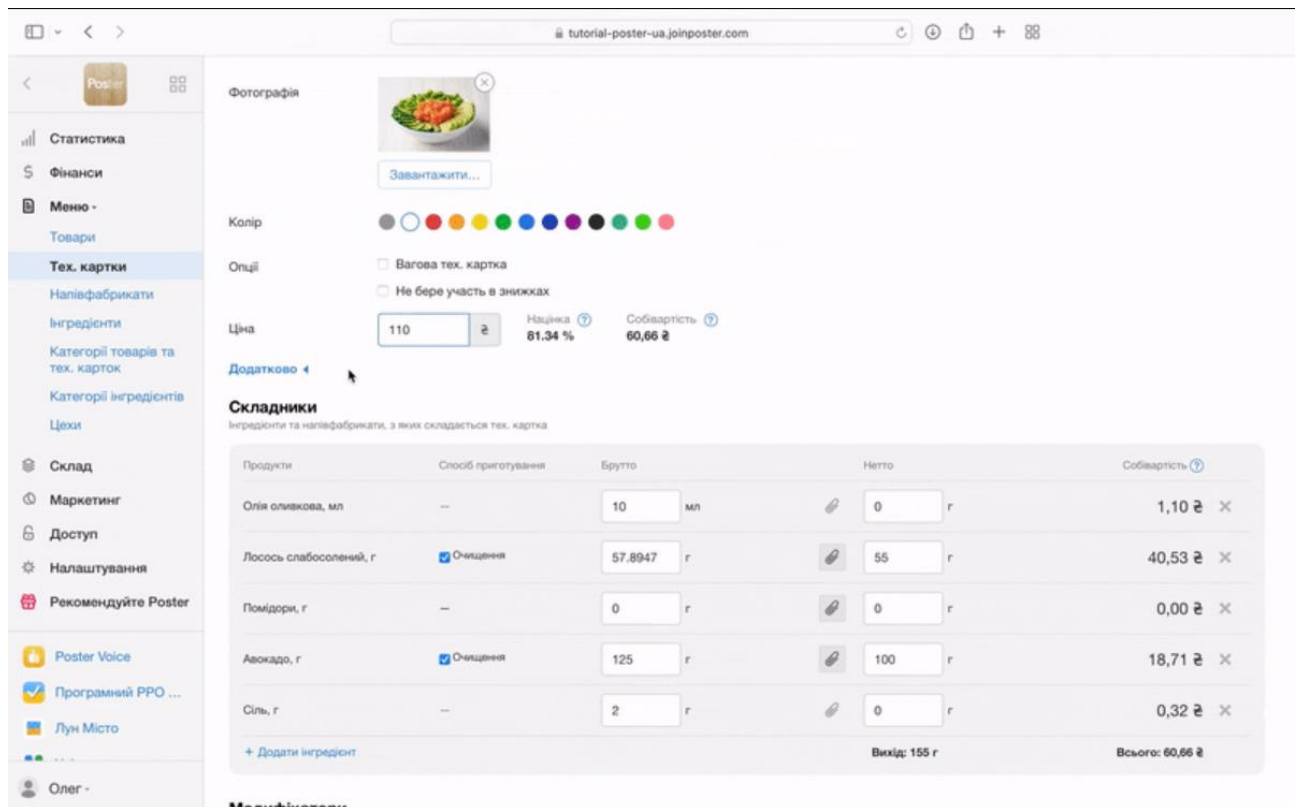
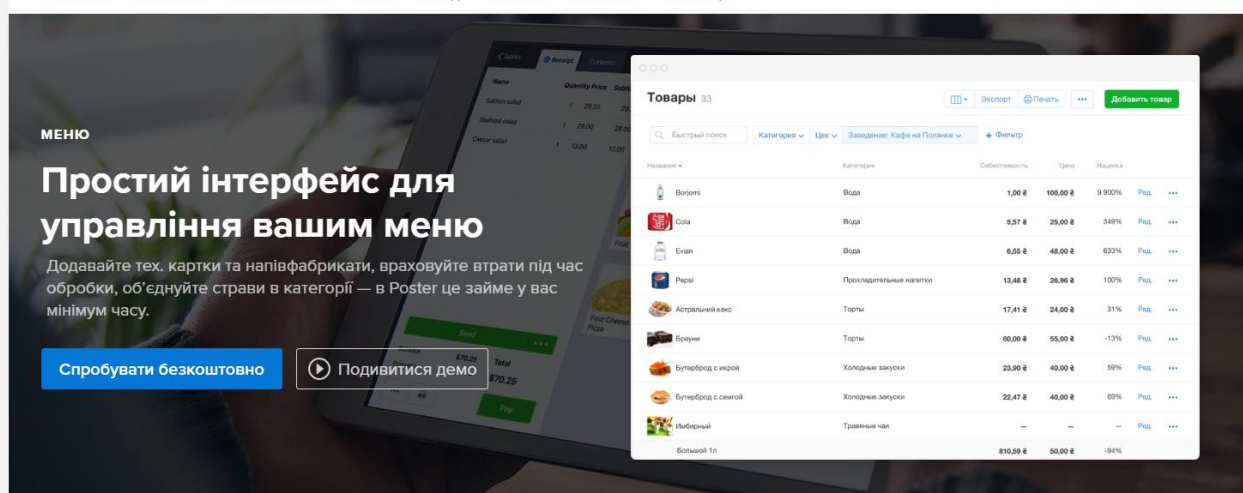


Рис. 1.1 – Інтерфейс програми Poster



Товари

Усе, що ви не готуєте в закладі та продаєте без обробки. Наприклад, шоколад чи лимонад у пляшках.



Інгредієнти

Продукти, з яких складаються чи готуються ваші страви. Наприклад, борошно, олія чи огірки.



Тех. картки

Страви, які ви готуєте за рецептом з інгредієнтів чи напівфабрикатів та подаєте в закладі. Наприклад, салат чи суп.



Напівфабрикати

Заготовки для ваших тех. карток — вони також готуються, але не продаються як страви. Наприклад, бульйон для супу чи тісто для кривасанів.

Рис 1.2 – Інтерфейс веб-сторінки Poster

ABC							 Распечатать		21 октября — 21 ноября			
 Быстрый поиск		Все официанты		Все заведения		Все категории		Все товары				
#	Товар	Продажи	Продажи, %	Выручка	Выручка, %	Прибыль	Прибыль, %	Продажи	Выручка	Прибыль		
1	Шу Пуэр 20 лет	372,00 шт	2,20%	164 889,00 €	11,55%	105 091,82 €	11,49%	A	A	A		
2	Калык (Al Fakher, Класическая чаша, Вода)	415,00 шт	2,45%	105 825,00 €	7,41%	99 475,50 €	10,88%	A	A	A		
3	Стейк из семги	424,00 шт	2,51%	76 320,00 €	5,35%	63 973,17 €	7,00%	A	B	C		
4	Салат с семгой	397,00 шт	2,35%	64 711,00 €	4,53%	58 683,22 €	6,42%	A	B	C		
5	Стейк Рибай	393,00 шт	2,32%	53 055,00 €	3,72%	49 464,72 €	5,41%	A	B	C		
6	Пломбир в стакане	409,00 шт	2,42%	50 307,00 €	3,52%	39 722,30 €	4,34%	A	B	C		
7	Речная форель	387,00 шт	2,29%	48 762,00 €	3,42%	31 852,94 €	3,48%	A	B	C		
8	Manhattan Jack	397,00 шт	2,35%	43 265,48 €	3,03%	34 991,24 €	3,83%	A	B	C		
9	Borjomi	421,00 шт	2,49%	42 100,00 €	2,95%	36 079,65 €	3,95%	A	B	C		
10	Салат с морепродуктами	391,00 шт	2,31%	41 055,00 €	2,88%	34 644,90 €	3,79%	A	B	C		
11	Роллы	404,00 шт	2,39%	36 360,00 €	2,55%	31 883,82 €	3,49%	A	B	C		
12	Пизца (Сыр + 2, Грибы, Ов, колбаски)	155,00 шт	0,92%	35 650,00 €	2,50%	33 491,64 €	3,66%	B	B	C		

Зробіть ваше меню ефективним із ABC-аналізом

Дізнайтесь, які страви мають попит та приносять більше виручки закладу — проводьте ABC-аналіз просто в Poster. Вирішіть, які страви краще вилучити з меню, а на які достатньо змінити ціну.

Рис. 1.3 – Приклад сторінки обліку різних продуктових позицій в додатку Poster

Проаналізувавши ще кілька аналогів, дійшов висновку, що подібні системи не зможуть задовільнити потреби підприємства через те, що вони не закривають повністю потреби підприємства та є платними. Тому було прийнято рішення розробляти індивідуальну інформаційну систему.

Індивідуальна інформаційна система може впорядкувати процеси на кухні, автоматизувавши рутинні завдання, звільнивши цінний час і ресурси, які можна перенаправити на інші сфери бізнесу. Система також може допомогти оптимізувати робочий процес, надаючи дані та інформацію в режимі реального часу, які можна використовувати для прийняття обґрунтованих рішень.

Однією з ключових переваг індивідуальної системи управління кухнею є можливість збирати та аналізувати дані в режимі реального часу. Це може допомогти визначити сфери для вдосконалення та прийняти рішення на основі даних, які можуть підвищити загальну продуктивність кухні..

Крім того, індивідуальне рішення може допомогти знизити витрати за рахунок зменшення відходів і підвищення ефективності. Наприклад, автоматизувавши управління запасами, система може допомогти забезпечити використання інгредієнтів до того, як вони зіпсуються, зменшити кількість харчових відходів і заощадити гроші. Крім того, оптимізуючи графік роботи співробітників і мінімізуючи час простою, система може допомогти підвищити загальну продуктивність.

На закінчення, хоча на ринку існує багато систем управління кухнею, індивідуальне рішення, розроблене спеціально для державного підприємства, може принести значні переваги. Воно може впорядкувати кухонні процеси, оптимізувати робочий процес і надати цінні дані та інформацію, які допоможуть підвищити загальну ефективність і продуктивність.

1.3 Вибір архітектури веб-сервісу

В процесі розгляду різних архітектурних патернів, було визначено основні характеристики та переваги кождої з них. Нижче наведений стислий порівняльний огляд таких архітектур:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						13

- Класична трирівнева архітектура (схема зображена на рис. 1.4), яку описується у попередньому тексті, складається з трьох рівнів: рівня презентації, бізнес-шару та рівня доступу до даних. Рівень презентації відповідає за інтерфейс між користувачем і програмою, відображення даних та збір введених користувачем даних. Бізнес-шар виконує обчислення, реалізує логіку додатку і взаємодіє з рівнем доступу до даних. Рівень доступу до даних забезпечує зберігання та отримання даних з бази даних [3-5].

Крім класичної трирівневої архітектури, існують інші типи архітектур, такі як монолітна архітектура, мікросервісна архітектура та сервіс-орієнтована архітектура (SOA).

- Монолітна архітектура (схема зображена на рис. 1.5) характеризується об'єднанням всіх компонентів програмного продукту в одну монолітну структуру. Усі код, бази даних та логіка розміщуються в одному місці [6-7].
- Мікросервісна архітектура (схема зображена на рис. 1.6) передбачає розбиття програмного продукту на невеликі незалежні мікросервіси, кожен з яких відповідає за свою функціональність і має власну базу даних та інтерфейси взаємодії з іншими сервісами [8-10].
- Сервіс-орієнтована архітектура (SOA) (схема зображена на рис. 1.7) включає незалежні сервіси, які надають послуги через стандартизовані інтерфейси комунікації, наприклад, веб-служби. Кожен сервіс виконує певні функції та взаємодіє з іншими сервісами за необхідності [11-13].

Зважаючи на особливості специфічного додатка, було обрано трирівневу архітектуру.

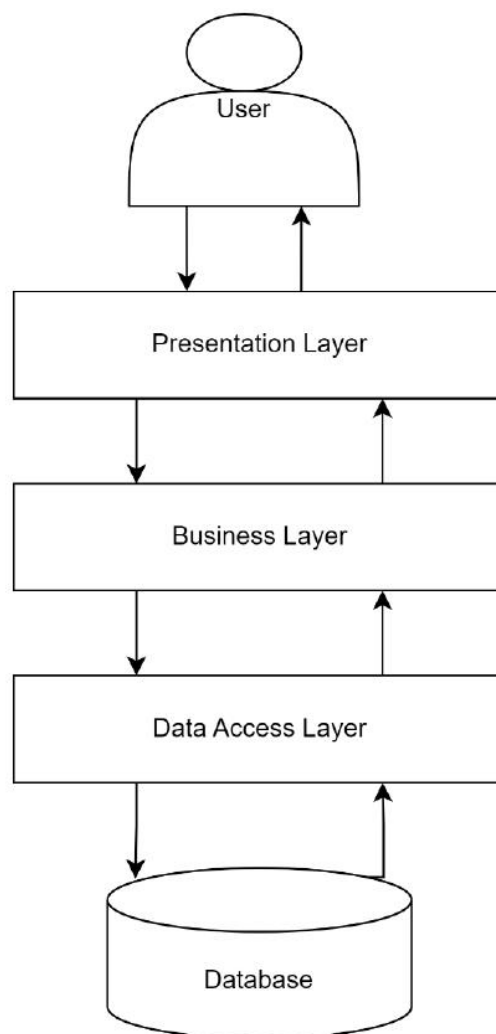


Рис. 1.4 – Схема трирівневої архітектури

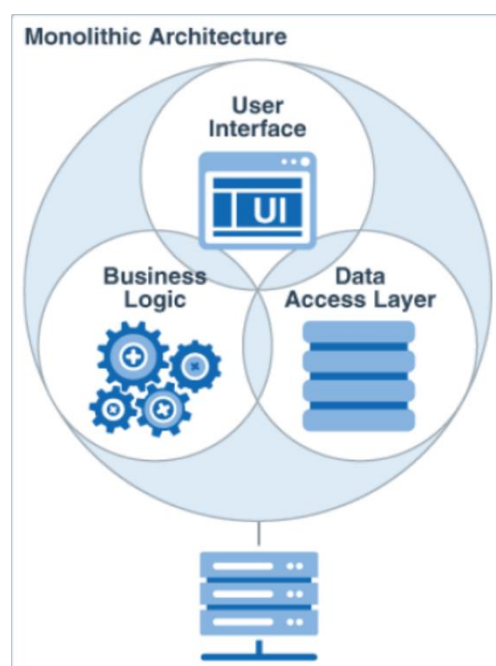


Рис. 1.5 – Схема монолітної архітектури

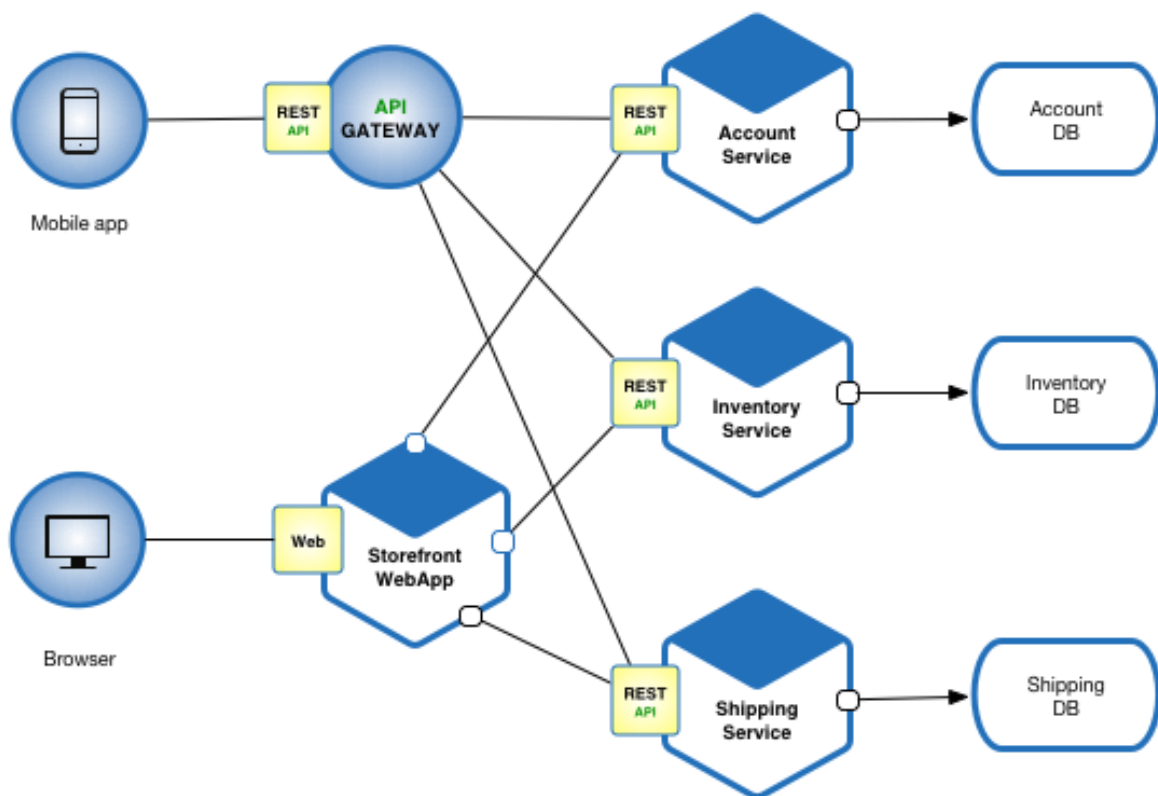


Рис. 1.6 – Схема мікросервісної архітектури

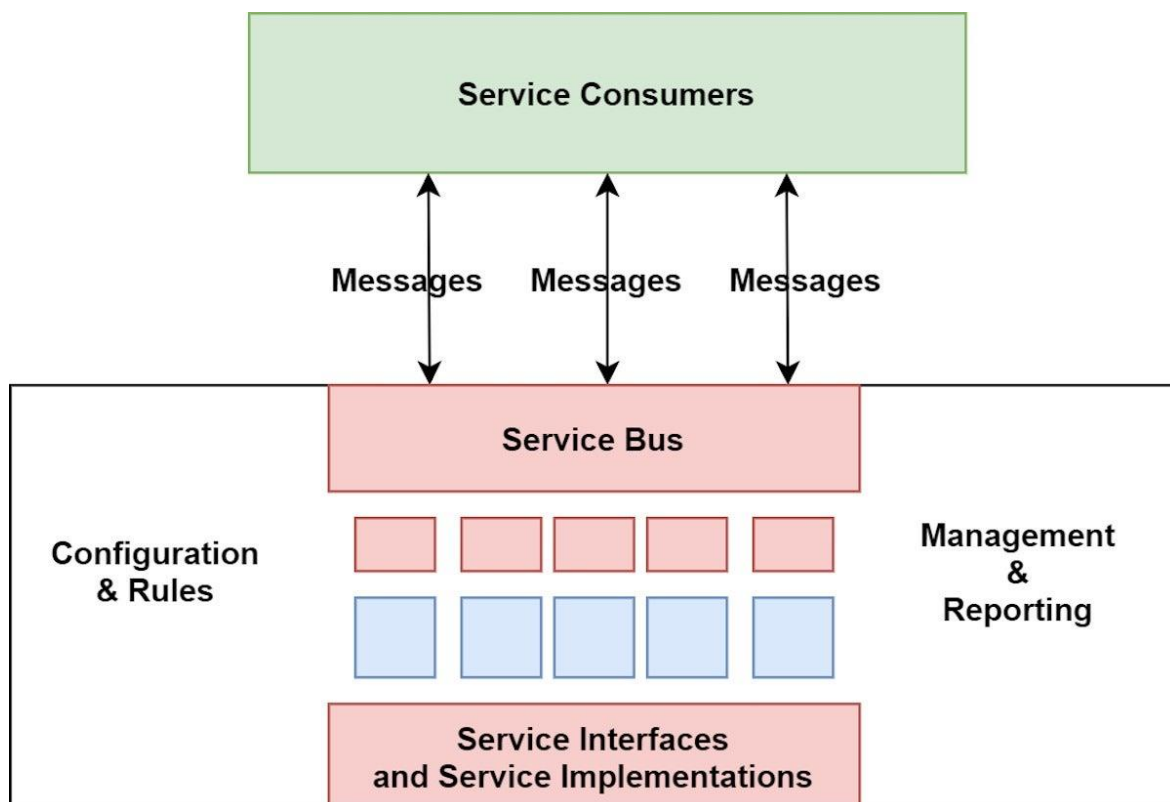


Рис. 1.7 – Схема SOA архітектури

Один з ключових аспектів вибору архітектури - це вибір технологій для реалізації кожного рівня з тришарової архітектури. У нашому випадку ми обрали наступну конфігурацію:

1. Користувацький інтерфейс

Для реалізації інтерфейсу я обрав фреймворк Angular з набором компонентів Material (розробки компанії Google, так само як і Angular). Angular є потужним фреймворком для побудови веб-інтерфейсів, який надає широкі можливості для створення інтерактивних та динамічних додатків. За допомогою Angular ми зможемо побудувати модульний та масштабований фронтенд для нашої веб-системи.

2. Бізнес шар:

Для реалізації бізнес шару буде використовуватись Angular, так як він дозволяє будувати репозиторії, сервіси та компоненти з коробки, а Typescript робить його типізований фреймворком, на якому можна зручно будувати бізнес логіку майже будь-якого рівня складності.

3. Шар даних (шар взаємодії з базою даних):

Цю роль на себе візьме також Angular та стандартна бібліотека для роботи з базою даних від Supabase

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Оберемо інструментальні засоби для розробки веб-сервісу:

Angular: Як вже було сказано, Angular буде використовуватися для побудови інтерфейсу програми. Angular - це популярний JavaScript-фреймворк з відкритим вихідним кодом для створення веб-додатків. Розроблений і підтримуваний компанією Google, він дозволяє розробникам з легкістю створювати динамічні та адаптивні користувацькі інтерфейси. Angular використовує компонентну архітектуру, що дозволяє розробникам створювати багаторазові компоненти інтерфейсу, які можна легко комбінувати для створення складних додатків.

Однією з ключових переваг Angular є двостороння прив'язка даних, яка усуває необхідність ручної маніпуляції з DOM і полегшує синхронізацію представлення та моделі. Angular також має потужну мову шаблонів, яка

дозволяє розробникам декларативно описувати структуру та поведінку користувацького інтерфейсу. Це, в свою чергу, полегшує створення та підтримку складних додатків.

Ще однією перевагою Angular є його модульна архітектура, яка дозволяє легко керувати залежностями між різними частинами програми. Це полегшує підтримку та тестування коду, а також дозволяє розробникам створювати та повторно використовувати модульні компоненти.

Angular також надає повний набір інструментів і сервісів для створення і тестування додатків, включаючи потужний інструмент CLI (інтерфейс командного рядка), великий набір готових компонентів і фреймворк для тестування.

Отже, Angular - це потужний і гнучкий фреймворк для створення веб-додатків, який широко використовується розробниками по всьому світу. Незалежно від того, чи створюєте ви невеликий односторінковий додаток, чи велику складну систему, Angular надає інструменти та фреймворк, необхідні для створення, тестування та розгортання високоякісних додатків [14-15].

Firestore: Firestore буде використовуватися як онлайн-хостинг. Firestore - це хмарна платформа для створення та управління веб і мобільними додатками. Вона була розроблена компанією Google і вперше запущена в 2011 році. Firestore надає широкий спектр послуг та функцій, які роблять її популярним вибором для створення масштабованих, продуктивних додатків, що працюють в режимі реального часу. Деякі з ключових особливостей Firestore включають

База даних в режимі реального часу: Firestore надає базу даних в режимі реального часу, яка дозволяє розробникам зберігати та отримувати дані в режимі реального часу. Це полегшує створення додатків, які потребують оновлення користувацького інтерфейсу при зміні даних, таких як додатки для чату, соціальні мережі та онлайн-ігри.

Аутентифікація: Firestore надає систему автентифікації, яка спрощує управління обліковими записами користувачів і забезпечує безпечний доступ до даних. Вона підтримує кілька постачальників автентифікації, таких як

електронна пошта і пароль, номер телефону та соціальні логіни, такі як Google і Facebook.

Хостинг: Firebase надає хостингову платформу, яка полегшує розгортання та розміщення веб-додатків. Хостинг Firebase забезпечує швидке, масштабоване та безпечне середовище для обслуговування веб-вмісту, а також включає такі функції, як шифрування SSL, власні домени та автоматичне кешування.

Хмарні функції: Firebase Cloud Functions - це безсерверна платформа, яка дозволяє розробникам запускати внутрішній код у відповідь на події, викликані службами Firebase. Це дає можливість створювати потужні та масштабовані бекенд-версії без необхідності керувати будь-якою інфраструктурою.

Хмарне сховище: Firebase надає хмарне рішення для зберігання даних, яке дозволяє розробникам зберігати та обслуговувати створений користувачем контент, наприклад, зображення та відео. Firebase Cloud Storage забезпечує масштабоване та продуктивне сховище, а також надійні функції безпеки та контролю доступу.

Аналітика: Firebase надає аналітичні можливості, які дозволяють розробникам отримати уявлення про те, як використовуються їхні додатки. Firebase Analytics надає інформацію в режимі реального часу про поведінку користувачів, таку як використання додатків, демографічні дані користувачів та відстеження подій.

ML Kit: Firebase також надає набір можливостей машинного навчання через свій ML Kit. Це дозволяє розробникам інтегрувати штучний інтелект і машинне навчання в свої додатки, наприклад, маркування зображень, розпізнавання облич і розпізнавання тексту.

Firebase надає простий і зрозумілий API, який дозволяє розробникам легко інтегрувати його сервіси в свої додатки. Він також надає веб-панель для управління та моніторингу додатків, що полегшує командам співпрацю та ефективну роботу.

Загалом, Firebase надає комплексну та багатфункціональну платформу для створення та управління веб- та мобільними додатками, що робить її

популярним вибором для розробників та організацій, які прагнуть створювати масштабовані, продуктивні та працюючі в режимі реального часу додатки [16-19].

Superbase – це реляційна база даних, яка пропонує розробникам потужні можливості для зберігання та керування даними в їх додатках. Вона забезпечує надійне зберігання даних, а також швидкий доступ до них з різних пристроїв і платформ. Ми будемо використовувати її в якості бази даних для нашого додатка.

Основні функції Superbase включають:

Реляційна база даних: Superbase використовує реляційну модель даних, що дозволяє структурувати дані у вигляді таблиць зі стовпцями та рядками. Це дає змогу ефективно організовувати дані та виконувати складні операції над ними, такі як пошук, сортування та зв'язки між таблицями.

Швидкий доступ до даних: Superbase підтримує швидкий доступ до даних з різних пристроїв та платформ, включаючи веб, мобільні пристрої та настільні комп'ютери. Це дозволяє розробникам створювати додатки, які працюють на будь-якому пристрої та забезпечують користувачам однаковий доступ до даних.

Розширені можливості синхронізації: Superbase надає розробникам потужні інструменти для синхронізації даних між різними пристроями та користувачами. Це означає, що якщо користувач змінює дані на одному пристрої, зміни автоматично синхронізуються з іншими пристроями. Це дозволяє створювати додатки з функцією офлайн-режиму та забезпечувати неперервний доступ до даних навіть при відсутності Інтернет-з'єднання.

Автентифікація та управління користувачами: Superbase має вбудовану систему автентифікації та управління користувачами. Це дозволяє розробникам створювати безпечні додатки, які вимагають авторизації для доступу до даних. Крім того, Superbase надає можливість використовувати різні методи аутентифікації, такі як соціальні мережі, електронна пошта або користувач і пароль.

Розширені запити та фільтрація: Superbase дозволяє розробникам створювати складні запити до бази даних, використовуючи різноманітні фільтри, сортування та групування даних. Це допомагає здійснювати потужний аналіз даних та отримувати необхідну інформацію з бази даних швидко та ефективно.

Інтеграція з іншими інструментами: Superbase можна легко інтегрувати з іншими інструментами та сервісами, такими як Node.js, Python, JavaScript, React і багато інших. Це дозволяє розробникам використовувати свої улюблені інструменти для роботи з базою даних та інтегрувати її з існуючими системами.

Superbase надає розробникам потужні можливості для створення додатків з надійним зберіганням та керуванням даними. З його допомогою розробники можуть швидко створювати сучасні додатки, які забезпечують користувачам швидкий та безпечний доступ до даних [20-22].

Visual Studio Code: Цей редактор коду можна використовувати для написання та налагодження коду для Angular та Firebase компонентів додатку. Visual Studio Code (VSCode) - це безкоштовний редактор коду з відкритим вихідним кодом, розроблений компанією Microsoft. Він призначений для Windows, macOS та Linux і широко використовується розробниками для широкого спектру мов і технологій програмування.

Однією з ключових особливостей VSCode є його розширюваність. Він має велику і постійно зростаючу екосистему розширень, які надають додаткову функціональність, таку як налагодження, лінчування, завершення коду тощо. Це дозволяє розробникам налаштовувати своє середовище розробки відповідно до їхніх конкретних потреб та робочих процесів.

VSCode також надає впорядкований користувацький інтерфейс, який полегшує навігацію кодом та пошук потрібної інформації. Він має потужний редактор коду з підтримкою підсвічування синтаксису, завершення коду та перевірки помилок, а також інтегровану підтримку налагодження та Git.

Ще однією перевагою VSCode є його продуктивність. Незважаючи на те, що це багатифункціональний редактор коду, він залишається швидким і легким,

а також швидко запускається. Це робить його ідеальним для використання як на настільних ПК, так і на ноутбуках.

Отже, Visual Studio Code - це універсальний і потужний редактор коду, який широко використовується розробниками по всьому світу. Незалежно від того, чи працюєте ви над особистим проектом, чи є частиною великої команди розробників,

VSCode надає інструменти та функціональність, необхідні для ефективної та результативної роботи [23-25].

Git: Git можна використовувати для контролю версій та спільної роботи. Git - це широко використовувана система контролю версій з відкритим вихідним кодом для розробки програмного забезпечення. Вона дозволяє розробникам відстежувати зміни, які вони вносять до свого коду, співпрацювати з іншими розробниками та керувати різними версіями своєї кодової бази.

За допомогою Git'у розробники можуть створювати локальні сховища свого коду на власних комп'ютерах, а потім переміщувати ці зміни до віддаленого сховища для спільного використання та співпраці. Це полегшує роботу кількох людей над однією і тією ж кодовою базою, а командам - управління змінами та вирішення конфліктів.

Git також надає надійну систему розгалуження та злиття, яка дозволяє розробникам працювати над декількома функціями або виправленнями помилок паралельно, не впливаючи на основну кодову базу. Коли вони будуть готові, вони можуть об'єднати свої зміни назад в основну гілку, а Git автоматично вирішить будь-які конфлікти, що можуть виникнути.

Ще однією ключовою перевагою Git'a є його розподілена природа. На відміну від централізованих систем контролю версій, де єдиний репозиторій виступає джерелом істини, Git дозволяє розробникам підтримувати кілька незалежних репозиторіїв. Це полегшує роботу в автономному режимі і забезпечує рівень надмірності, який може бути цінним у разі збоїв або втрати даних [26-27].

Отже, Git є критично важливим інструментом для сучасної розробки програмного забезпечення, і ним користуються мільйони розробників та організацій по всьому світу. Незалежно від того, чи працюєте ви над невеликим особистим проектом, чи над великим корпоративним додатком, Git надає функції контролю версій та спільної роботи, необхідні для ефективного управління кодовою базою.

Angular був вибраний для реалізації даного проекту через те, що це дуже гнучкий сучасний фреймворк, який має широкий функціонал і чудово підійде для того, що розробити функціональний та приємний бізнес-додаток. Також розглядались React та Vue, але Angular сподобався своїми ООП підходами та відносною простотою входу, для людей, які раніше працювали з об'єктно орієнтованими мовами. **Firebase** був вибраний як сучасна система для хостингу. **Superbase** надає гнучкі умови для розробників та відносну простоту розгортання середовища. Плюсом такого підходу є те, що не потрібно розгортати базу локально та займатись розгортанням сервера.

Системні вимоги: так як це веб-додаток, то основною вимогою є підтримка браузером HTML5 та CSS3. Рекомендовані браузери: Google Chrome, Mozilla Firefox, Safari or Microsoft Edge.

Висновки до розділу 1

У першому розділі основної частини нашої роботи, який має теоретичний характер, ми провели аналіз інформаційної бази, літературних джерел та результатів переддипломної практики для визначення основних завдань впровадження інформаційних технологій у нашому дослідженні.

В ході аналізу існуючих аналогів програмного продукту було визначено стандартний функціонал, який присутній у програмних продуктах відповідного класу. Однак, виявлено, що дані продукти не можуть задовольнити потреби підприємства в повному обсязі.

Надалі, була розроблена архітектура розроблюваного додатка, що включає укрупнену структуру системи. Визначені компоненти, їх взаємозв'язок та

					ІПЗ.КР.Б – 121 – 23 – ПЗ	23

функціональні можливості, що сприятимуть ефективній реалізації та функціонуванню програмного комплексу.

Для реалізації програмного продукту ми обрали Angular як інструментарій розробки. Цей вибір обґрунтований його масштабованістю, компонентним підходом, оптимізованою швидкодією, розширюваністю та підтримкою від Google. Angular надає нам потужний фреймворк для швидкої та стабільної розробки нашого додатка.

Також, в першому розділі були визначені необхідні інструментальні засоби для реалізації програмного продукту та поставлені вимоги до апаратних засобів, необхідних для ефективного функціонування нашого додатка.

У параграфах цього розділу ми детально розглянули:

1. Перелік задач, які підлягають автоматизації у нашій предметній області.
2. Існуючі аналоги програмного продукту.
3. Архітектуру та укрупнену структуру майбутнього програмного комплексу.
4. Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів.

Кожен параграф може містити окремі змістовні підпункти, що деталізують відповідні аспекти нашого дослідження.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-СИСТЕМИ ДЛЯ КУХНІ ЇДАЛЬНІ НА ПІДПРИЄМСТВІ

2.1 Визначення варіантів використання та структури системи.

Веб-система створюється для спрощення роботи комірника кухні комунального освітнього підприємства.

Вимоги користувачів.

Користувачам надається можливість скористатися наступними функціями:

1. Створення та редагування продуктів;
2. Редагування кількості продуктів на складі;
3. Створення меню;
4. Створення та перегляд страв;
5. Експорт меню.

Варіанти використання веб-сервісу зображені на рис. 2.1.

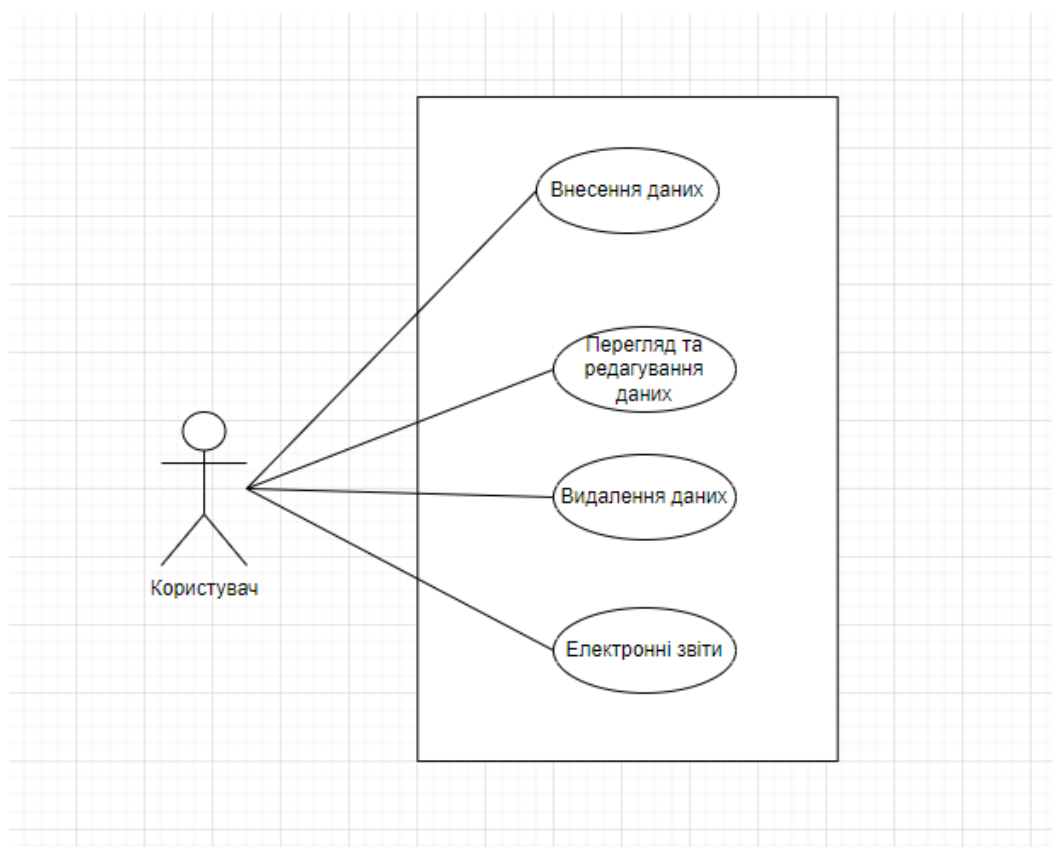


Рис. 2.1 – Варіанти використання

Функціональні вимоги:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	25

1. Додавання продуктів - користувач повинен мати можливість додавати нові найменування продуктів в систему.
2. Редагування продуктів - користувач повинен мати можливість редагувати уже додані найменування продуктів в системі.
3. Видалення продуктів – користувач повинен мати можливість видаляти додані найменування продуктів.
4. Додавання продуктів на склад – користувач повинен мати можливість додавати продукти на склад.
5. Створення меню – користувач повинен мати можливість створювати меню на день та обирати тип цього меню.
6. Створення страв – користувач повинен мати можливість створювати страви прямо на сторінці меню.
7. Експорт меню – користувач повинен мати можливість експортувати меню, для того щоб його можна було надрукувати та вивісити в залі.

Організаційну структуру додатка можна представити у вигляді діаграми (рис. 2.2).

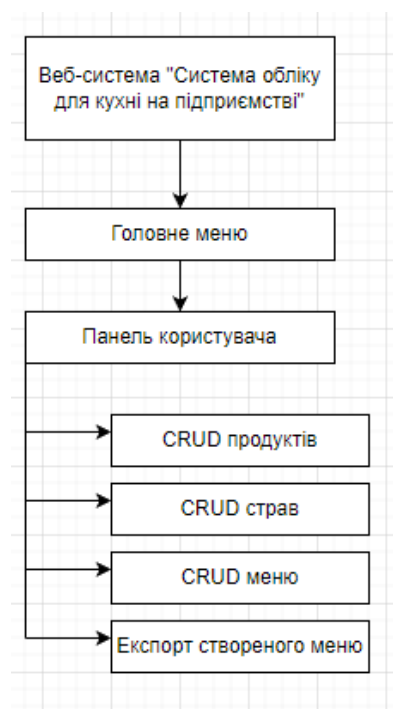


Рис. 2.2 – Організаційна структура

Нефункціональні вимоги веб-системи обліку для кухні їдальні на підприємстві:

1. Зручність інтерфейсу користувача: Система повинна забезпечувати простоту роботи з нею, так як користуватись додатком будуть невпевнені користувачі ПК.
2. Сумісність: Система повинна бути сумісною з різними платформами та браузерами, щоб користувачі могли отримати доступ до неї з різних пристроїв та операційних систем.
3. Масштабованість: Система повинна бути легко масштабованою, щоб відповідати зростаючим потребам підприємства. Вона повинна підтримувати збільшення обсягу даних, кількості користувачів та оптимально працювати навіть при збільшенні навантаження.
4. Надійність: Система повинна бути надійною та стабільною, забезпечуючи безперебійну роботу навіть при високих навантаженнях. Вона повинна бути захищеною від відмов та забезпечувати резервне копіювання даних для запобігання втраті інформації.
5. Ефективність: Система повинна працювати швидко та ефективно, забезпечуючи швидкий доступ до інформації та оптимальну продуктивність. Це важливо для забезпечення швидкого оброблення замовлень та зменшення часу очікування для користувачів.

Модуль «Продукти» використовує діаграму класів, зображену на рис. 2.3, для втілення сутностей, виявлених під час аналізу функціональних вимог, що забезпечують реалізацію програмного комплексу системи.

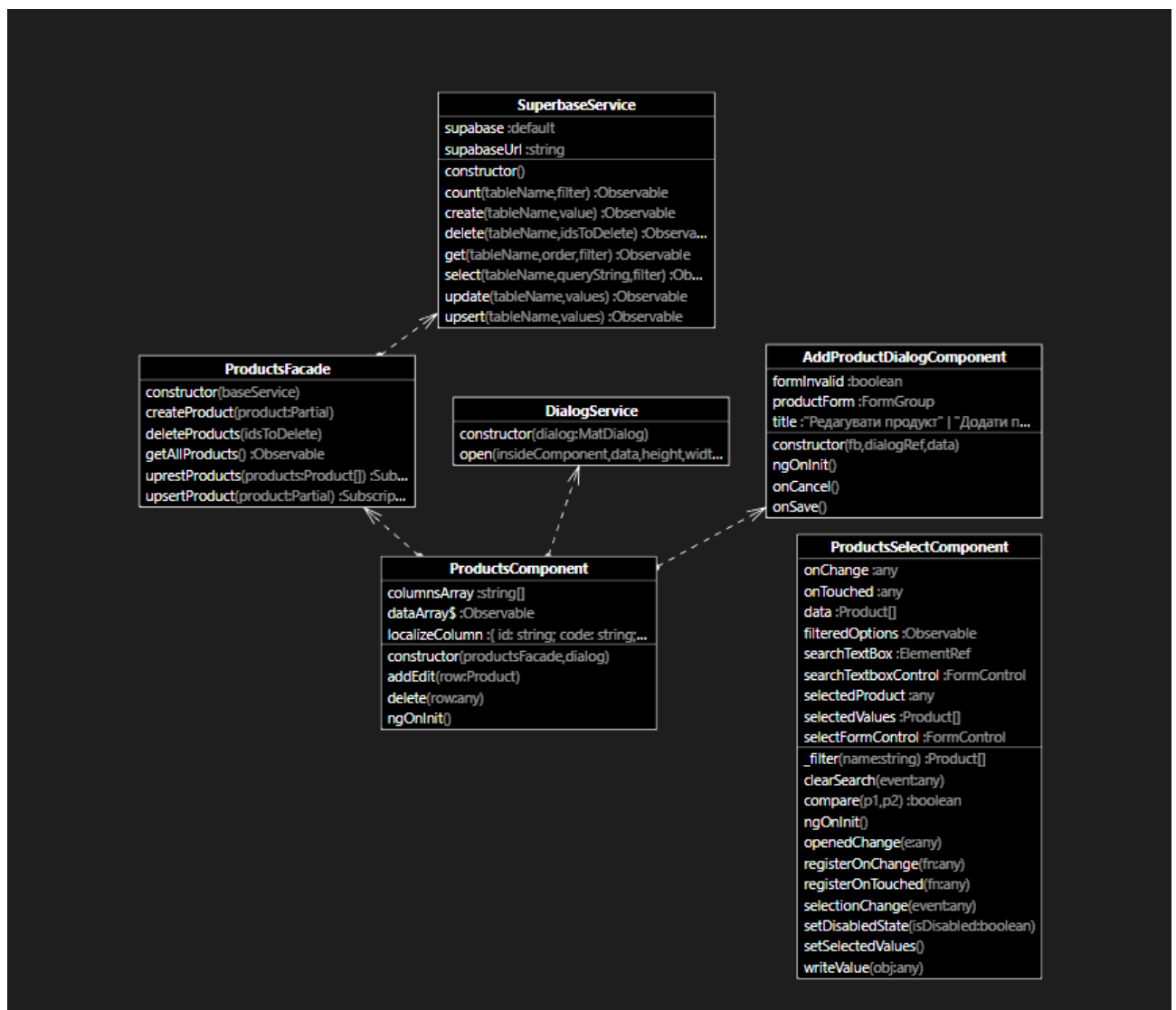


Рис. 2.3 - Діаграма класів модуля «Продукти»

ProductsFacade:

- `getAllProducts(): Observable<Product[]>` - отримує всі продукти шляхом виклику методу `get` базового сервісу з вказаною таблицею та сортуванням по ідентифікатору.
- `createProduct(product: Partial<Product>)` - створює новий продукт шляхом виклику методу `create` базового сервісу з вказаною таблицею та даними продукту.
- `deleteProducts(idsToDelete: number[])` - видаляє вказані продукти шляхом виклику методу `delete` базового сервісу з вказаною таблицею та списком ідентифікаторів продуктів.
- `upsertProduct(product: Partial<Product>)` - оновлює або створює

продукт шляхом виклику методу upsert базового сервісу з вказаною таблицею та даними продукту.

- uprestProducts(products: Partial<Product[]>) - оновлює або створює декілька продуктів шляхом виклику методу upsert базового сервісу з вказаною таблицею та списком даних продуктів.

ProductsComponent:

- delete(row: any) - видаляє вказаний рядок (продукт) шляхом виклику методу deleteProducts з екземпляра productsFacade і передачі його ідентифікатора.
- addEdit(row?: Product) - відкриває діалогове вікно для додавання або редагування продукту. Якщо переданий параметр row не є undefined, він передається як дані для діалогового вікна. Після закриття діалогового вікна, якщо отримано об'єкт продукту, викликається метод upsertProduct з екземпляра productsFacade для оновлення або створення продукту. Після цього оновлюється змінна dataArray\$, яка містить список всіх продуктів шляхом виклику методу getAllProducts з екземпляра productsFacade.

ProductsSelectComponent:

- ngOnInit(): Ініціалізує filteredOptions, використовуючи valueChanges з searchTextboxControl, і пропускає його через функцію _filter, щоб відфільтрувати опції на основі введеного значення.
- private _filter(name: string): Product[]: Фільтрує список продуктів на основі введеного значення name. При цьому встановлює selectedValues і відповідно оновлює значення selectFormControl. Повертає відфільтрований список продуктів.
- selectionChange(event: any): Обробляє зміну вибору елемента. Якщо зміна викликана користувачем і вибраний елемент відзначений як "не вибраний", видаляє його з selectedValues. Потім викликає onChange з selectedValues.
- openedChange(e: any): Обробляє зміну стану відкриття

комбінованого списку. Якщо він відкривається (`e == true`), фокусується на полі пошуку `searchTextBox`.

- `clearSearch(event: any)`: Очищує значення `searchTextboxControl`.
- `setSelectedValues()`: Встановлює `selectedValues` на основі значень `selectFormControl`. Додає відсутні значення до `selectedValues`.
- `writeValue(obj: any): void`: Встановлює значення `selectFormControl` на основі `obj`.
- `registerOnChange(fn: any): void`: Реєструє функцію `fn` як зворотний виклик `onChange`.
- `registerOnTouched(fn: any): void`: Реєструє функцію `fn` як зворотний виклик `onTouched`.
- `compare(p1: Product, p2: Product)`: Порівнює два об'єкти продукту `p1` і `p2` за їх ідентифікаторами (`id`). Повертає `true`, якщо вони рівні.
- `setDisabledState?(isDisabled: boolean): void`: Не реалізовано.
- `private onChange: any = () => {};`: Змінна для збереження зворотного виклику `onChange`.
- `private onTouched: any = () => {};`: Змінна для збереження зворотного виклику `onTouched`.

Модуль Склад вміщає у собі класи, зображені на рис. 2.4.



Рис. 2.4 – Діаграма класів модуля «Склад»

WarehouseComponent:

Метод addWeight(item: any) додає вагу до продукту і оновлює список продуктів на складі.

EditWeightDialogComponent:

- ngOnInit(): void - Ініціалізує форму weightForm для редагування ваги продукту.
- onCancel(): void - Закриває діалогове вікно без збереження змін.
- onSave(): void - Зберігає значення форми weightForm та закриває діалогове вікно.
- get title(): string - Повертає заголовок діалогового вікна на основі назви продукту.
- get formInvalid(): boolean - Повертає true, якщо форма weightForm недійсна.

WarehouseProductFacade:

- `getWarehouseProducts(): Observable<WarehouseProduct[]>` - отримує список продуктів на складі з бази даних.
- `createProduct(product: Partial<WarehouseProduct>)` - створює новий продукт на складі.
- `deleteProducts(idsToDelete: number[])` - видаляє вказані продукти зі складу.
- `upsertProduct(product: Partial<WarehouseProduct>)` - оновлює або створює продукт на складі.
- `ngOnInit(): void` - ініціалізує форму для редагування ваги продукту.
- `onCancel(): void` - закриває діалогове вікно без збереження змін.
- `onSave(): void` - зберігає значення форми та закриває діалогове вікно.
- `getTitle(): string` - повертає заголовок діалогового вікна на основі назви продукту.
- `get formInvalid(): boolean` - повертає true, якщо форма недійсна.

SuperbaseService:

- `get<T>(tableName: TableName, order?: string, filter?: FilterParams): Observable<T[]>` - отримує дані з вказаної таблиці з можливістю сортування та фільтрації.
- `select<T>(tableName: TableName, queryString: string, filter?: FilterParams)` - виконує власний запит до вказаної таблиці з можливістю фільтрації.
- `count(tableName: TableName, filter?: FilterParams): Observable<number>` - отримує кількість записів у вказаній таблиці з можливістю фільтрації.
- `create<T>(tableName: TableName, value: Partial<T>)` - створює новий запис у вказаній таблиці з вказаними значеннями.
- `delete(tableName: TableName, idsToDelete: number[])` - видаляє вказані записи з вказаної таблиці за їх ідентифікаторами.

- `update<T>(tableName: TableName, ...values: T[])` - оновлює вказані записи у вказаній таблиці з новими значеннями.
- `upsert<T>(tableName: TableName, values: T[])` - оновлює або створює записи у вказаній таблиці з новими значеннями.

Класи модуля Меню зображено на рис. 2.5.

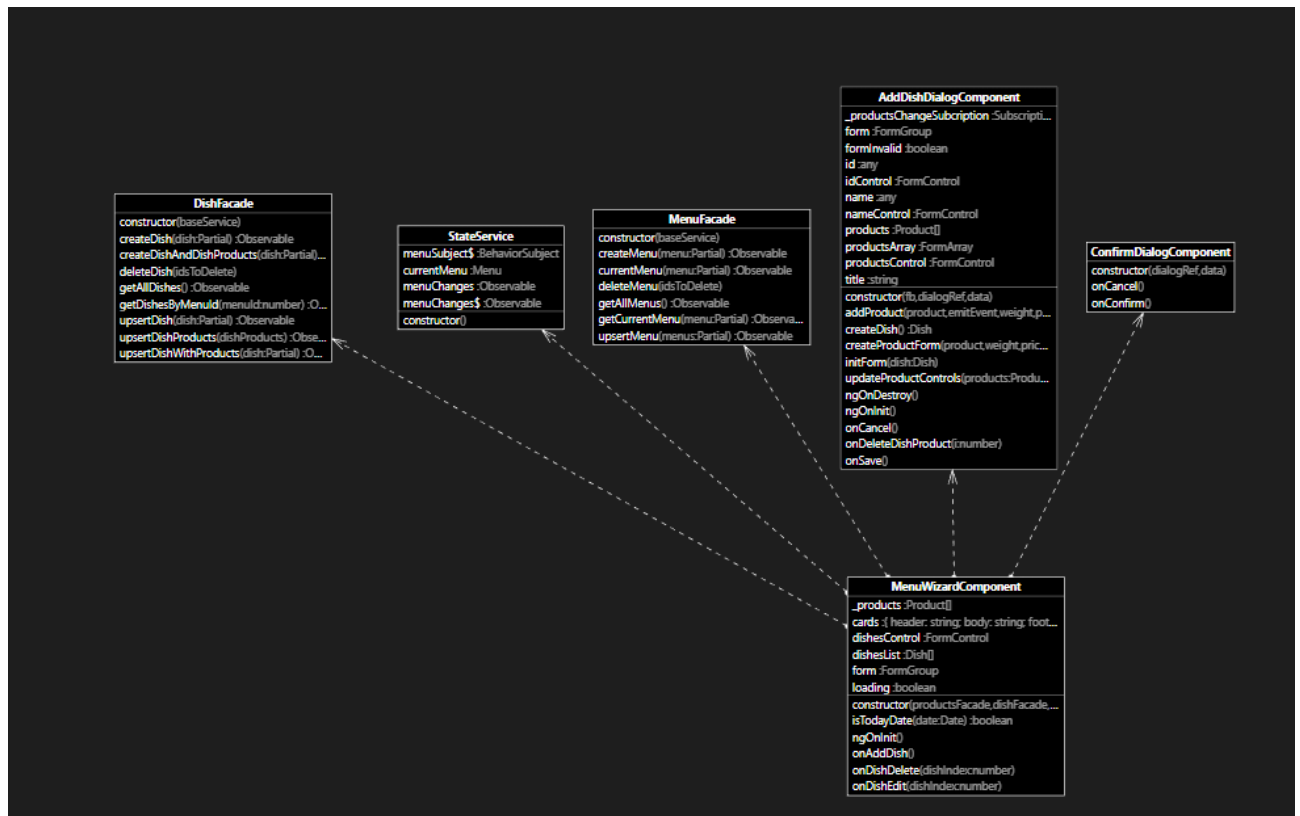


Рис. 2.5 – Діаграма класів модуля «Меню»

MenuWizardComponent:

- `ngOnInit(): void`: Ініціалізує компонент. Підписується на зміни меню (`menuChanges$`) та отримує поточне меню (`getCurrentMenu`) та страви, пов'язані з цим меню (`getDishesByMenuId`). Встановлює значення `dishesControl` на отримані страви.
- `onAddDish()`: Відкриває діалогове вікно (`AddDishDialogComponent`) для додавання нової страви. Після закриття діалогу, якщо було отримано страву (`d`), викликає `upsertDishWithProducts` для додавання або оновлення страви та оновлює значення `dishesControl`.
- `onDishDelete(dishIndex: number)`: Відкриває діалогове вікно підтвердження (`ConfirmDialogComponent`) для підтвердження

видалення страви. Після закриття діалогу, якщо підтверджено видалення ($x === true$), видаляє страву зі списку `dishesList` та викликає `deleteDish` для видалення страви з бази даних.

- `onDishEdit(dishIndex: number)`: Відкриває діалогове вікно (`AddDishDialogComponent`) для редагування страви. Після закриття діалогу, якщо було отримано відредаговану страву (`d`), оновлює відповідну страву в `dishesList` та викликає `upsertDishWithProducts` для оновлення страви в базі даних.
- `public isTodayDate(date: Date)`: Перевіряє, чи вказана дата (`date`) є сьогоднішньою датою. Порівнює рік, місяць та день вказаної дати з поточною датою. Повертає `true`, якщо дата є сьогоднішньою, інакше - `false`.

AddDishDialogComponent:

- `ngOnInit()`: Ініціалізує компонент. Створює форму (`form`) з відповідними контролами для полів `id`, `name`, `products`, `productsFormArray`. Якщо передана страву (`this.data.dish`), ініціалізує форму зі значеннями страви та встановлює заголовок діалогового вікна (`title`) на "Редагувати страву". Інакше, встановлює заголовок на "Додати страву". Створює підписку на зміни значення `productsControl`.
- `onCancel(): void`: Закриває діалогове вікно без збереження змін.
- `onSave(): void`: Створює об'єкт страви зі значень полів форми та передає його назад до компонента, який відкрив діалогове вікно. Після цього закриває діалогове вікно.
- `onDeleteDishProduct(i: number)`: Видаляє продукт зі списку продуктів страви за індексом `i`. (Закоментований код видаляє продукт із списку `selectedProducts`, але не здійснює оновлення значення `productsControl` та видалення елемента з `productsArray`.)

- private addProduct(product: Product, emitEvent: boolean = true, weight: string | number = "", price: string | number = ""): Додає продукт до списку продуктів страви, створюючи відповідний FormControl для продукту з вказаною вагою (weight) та ціною (price). Опціонально вмикає або вимикає події емітовання.
- private updateProductControls(products: Product[]): Оновлює контроли продуктів відповідно до переданого списку продуктів. Додає нові продукти, які не містяться в productsArray, та видаляє продукти, які не містяться в переданому списку products.
- private initForm(dish: Dish): Ініціалізує форму значеннями переданої страви. Встановлює значення idControl, nameControl та productsControl. Додає продукти до productsArray з відповідними значеннями ваги та ціни.
- private createProductForm(product: Product, weight: string | number = "", price: string | number = ""): FormGroup: Створює FormGroup для продукту з вказаними значеннями ваги (weight) та ціни (price).
- private createDish(): Dish: Створює об'єкт страви зі значень полів форми. Встановлює ідентифікатори продуктів та посилання на об'єкти продуктів у відповідних DishProduct. Повертає створений об'єкт страви.
- ngOnDestroy(): Відписується від підписки на зміни значення productsControl при знищенні компонента.

MenuFacade:

- currentMenu(menu: Partial<Menu>): Observable<Menu>: Повертає поточне меню на основі переданого menu. Спочатку спробує отримати поточне меню (getCurrentMenu(menu)). Якщо меню не існує, створить нове (createMenu(menu)) та поверне його. Якщо меню вже існує, оновить його servingsCount та поверне оновлене меню.

- `getCurrentMenu(menu: Partial<Menu>): Observable<Menu>`: Повертає поточне меню на основі переданого `menu`. Виконує запит до бази даних, шукаючи запис, який має відповідати `menuType`, `preferential` та `date`. Повертає перше знайдене меню або `null`, якщо меню не знайдено.
- `getAllMenus(): Observable<Menu[]>`: Повертає всі меню з бази даних.
- `createMenu(menu: Partial<Menu>): Observable<any>`: Створює нове меню в базі даних на основі переданого `menu`. Повертає результат операції створення.
- `deleteMenu(idsToDelete: number[])`: Видаляє меню з бази даних за заданими ідентифікаторами `idsToDelete`.
- `upsertMenu(menus: Partial<Menu>[])`: Додає або оновлює (`upsert`) меню в базі даних на основі переданого масиву `menus`.

2.2 Розробка бази даних системи

Наступним кроком буде розробка бази даних. В нашому проєкті використовується Supabase. `ConnectionString` для бази. Конфігурація бази даних зображена на рис. 2.6.

```

src > environments > TS environment.ts > [⌕] environment
1  // This file can be replaced during build by using the `fileReplacements` array.
2  // `ng build` replaces `environment.ts` with `environment.prod.ts`.
3  // The list of file replacements can be found in `angular.json`.
4
5  export const environment = {
6    production: false,
7
8    API_KEY: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdXBhYmFzZSI6InJlZiI
9  };
10
11  /*
12   * For easier debugging in development mode, you can import the following file
13   * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.i
14   *
15   * This import should be commented out in production mode because it will have a

```

Рис. 2.6 - Конфігурація бази даних

Структура бази даних зображена на рис. 2.7.

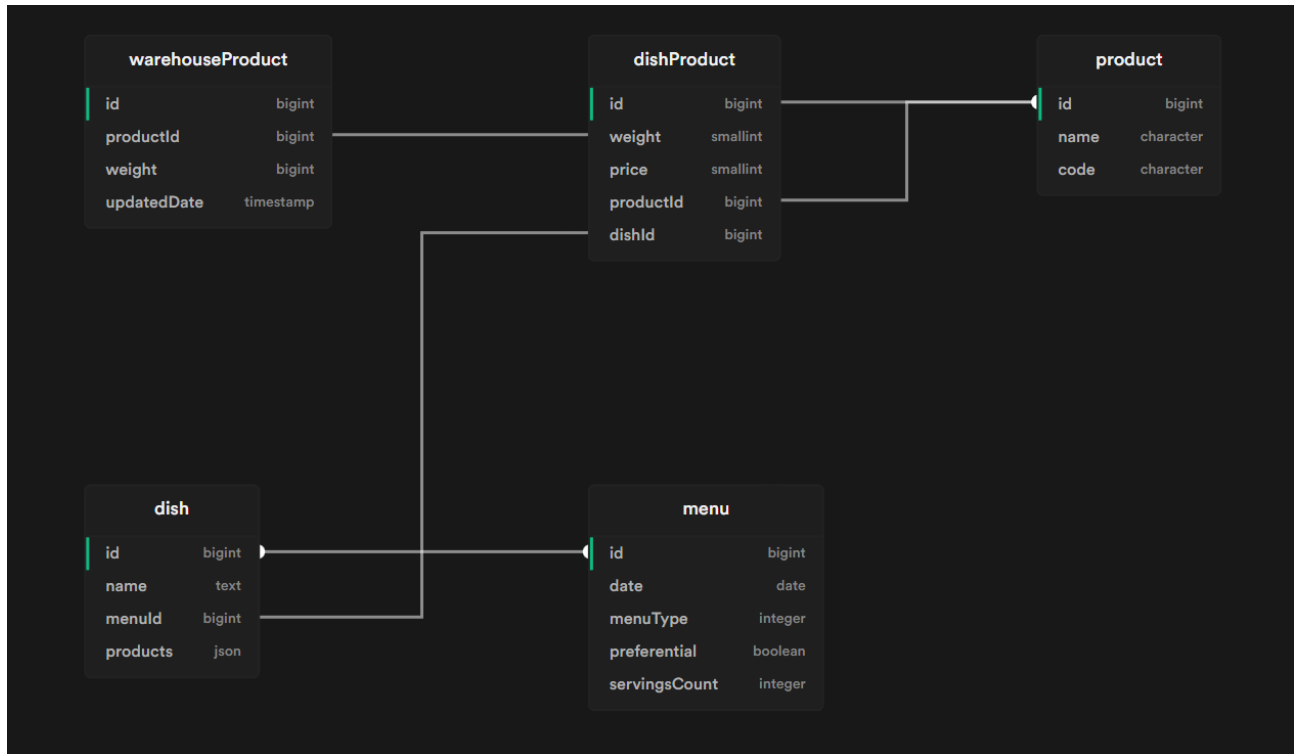


Рис. 2.7 – Структура бази даних

2.3 Опис взаємодії елементів системи під час роботи програмного комплексу

Під час роботи веб-додатку для кухні, взаємодія компонентів та бази даних здійснюється через базовий сервіс, який передбачає наступну послідовність дій для користувача.

Спочатку користувачу потрібно налаштувати продукти. Це означає, що він визначає перелік продуктів, які будуть використовуватись у системі, такі як овочі, фрукти, м'ясо, приправи та інші. Користувач може внести дані про кожен продукт, включаючи назву, опис, категорію та характеристики, які будуть використовуватись під час створення страв.

Після налаштування продуктів користувачу потрібно налаштувати склад. Склад відображає наявність продуктів і їх кількість на кухні. Користувач може внести дані про кожен продукт, вказуючи його кількість або залишок. Це допомагає системі контролювати наявність продуктів та попереджати про їхню нестачу.

Після успішного налаштування продуктів та складу користувач може використовувати просту форму для додавання меню та страв. Він може обрати день, на який створюється меню, та додавати страви, вибираючи продукти з попередньо налаштованого списку. Користувач може внести назву страви, опис, склад і кількість необхідних продуктів. Коли користувач зберігає страву, дані про неї записуються в базу даних через базовий сервіс.

Таким чином, взаємодія компонентів, таких як форма користувача, базовий сервіс та база даних, забезпечує плавну та ефективну роботу системи. Користувач може легко взаємодіяти з додатком, налаштовувати продукти, склад, створювати меню та додавати страви, а всі необхідні дані зберігаються в базі даних для подальшого використання.

2.4 Реалізація веб-системи обліку для кухні їдальні на підприємстві

Головним модулем нашого додатка є модуль Меню.

В табл. 2.1 наведено назву методів та покроковий опис того, що робить кожен метод у класі `AddDishDialogComponent`:

Таблиця 2.1

Методи `AddDishDialogComponent`

Назва методу	Опис
ngOnInit	Метод використовується для виклику методів після ініціалізації змінних, що приходять через конструктор або <code>@Input()</code> параметри
onCancel	Закриває діалог без збереження результату
onSave	Закриває діалог зі збереженням результату
addProduct	Додає продукт в форму
initForm	Ініціалізує форму після відкриття діалогу
createProductForm	Використовується для динамічного створення форми продукту
createDish	Створює сутність страви з форми перед закриттям діалогу

Код цього діалогу наведено нижче:

```
@Component({
  selector: 'app-add-dish-dialog',
  templateUrl: './add-dish-dialog.component.html',
  styleUrls: ['./add-dish-dialog.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class AddDishDialogComponent {
  form: FormGroup;
  private _productsChangeSubscription: Subscription;
```

```

title: string = "";

constructor(
  private fb: FormBuilder,
  private dialogRef: MatDialogRef<AddDishDialogComponent>,
  @Inject(MAT_DIALOG_DATA) public data: any
) { }

ngOnInit() {
  this.form = this.fb.group({
    id: ["", Validators.required], //DON'T FORGET TO ADD VALIDATORS.PATTERN
    products: [[], Validators.required],
    productsFormArray: this.fb.array([])
  });

  if(this.data.dish){
    this.title = 'Редагувати страву';
    this.initForm(this.data.dish);
  }

  this.title = 'Додати страву';

  setTimeout(()=>{
    this._productsChangeSubscription = this.productsControl.valueChanges.subscribe((p: Product[]) => {
      this.updateProductControls(p);
    });
  });

  onCancel(): void {
    this.dialogRef.close();
  }

  onSave(): void {
    const dish = this.createDish();
    this.dialogRef.close(dish);
  }

  onDeleteDishProduct(i: number){;
  }

  private addProduct(product: Product, emitEvent: boolean = true, weight: string | number = "", price: string | number = "") {
    this.productsArray.push(this.createProductForm(product, weight, price), {emitEvent: emitEvent});
  }

  private updateProductControls(products: Product[]){
    products.forEach(p => {
      if(!this.productsArray.value.find((x: any) => x.name == p.name)){
        this.addProduct(p);
      }
    });

    this.productsArray.value.forEach((pc: any, i: number) => {
      if(!products.find((x) => x.name == pc.name)){
        this.productsArray.removeAt(i);
      }
    })
  }

  private initForm(dish: Dish){
    this.idControl.setValue(dish.id);
  }

```

```

this.nameControl.setValue(dish.name, {emitEvent: false});
this.productsControl.setValue(dish.products.map(p => new Product(p.product)));
dish.products.forEach(dp => {
  this.addProduct(dp.product, false, dp.weight, dp.price);
});
}

private createProductForm(product: Product, weight: string | number = "", price: string | number = ""): FormGroup{
  const productForm = this.fb.group({
    id: [product.id],
    name: [product.name],
    weight: [weight, Validators.required],
    price: [price, Validators.required]
  });

  return productForm;
}

private createDish(): Dish{
  let dishProducts = this.productsArray.value;
  dishProducts.map((dp: DishProduct, i: number) => {
    dp.productId = this.products[i].id;
    dp.product = this.products[i];
  });
  let dish = new Dish();
  if(this.id){
    dish.id = this.id;
  }
  dish.products = dishProducts;
  dish.name = this.name;

  return dish;
}

ngOnDestroy(){
  this._productsChangeSubscription.unsubscribe();
}
}

```

Також хочу навести приклад методу, який зберігає повернуте з діалога значення в базу даних.

```

onAddDish(){
  this.dialog.open(AddDishDialogComponent, {products: this._products}).afterClosed().subscribe((d: Dish) => {
    if(d){
      d.menuId = this.state.currentMenu.id;
      this.dishFacade.upsertDishWithProducts(d).subscribe(x => {
        let dishesList = this.dishesList;
        d.id = x.id;
        dishesList.push(d);
        this.dishesControl.patchValue(dishesList);
      });
    }
  });
}

```

Сам метод upsert`у виглядає так

```

upsert<T>(tableName: TableName, values: T[]){
  let query = this.supabase.from(tableName).upsert(values).select();
  return from(query).pipe(map(value => value.data));
}

```

Висновки до розділу 2

В другому розділі нашої роботи було проведено аналіз вимог до розроблюваного програмного продукту. Результати аналізу показали, що існують функціональні та нефункціональні вимоги від користувачів різних типів. Були враховані вимоги до часових характеристик реакції системи на дії користувача. Варіанти використання системи були специфіковані, а також наведена узагальнена схема елементів системи разом з описом їх структури та методів.

У другому параграфі була представлена фізична схема даних та опис структури збереження даних. Це дозволило нам розуміти, як дані будуть організовані та зберігатись в системі. Деталізований опис наведено у додатку М.

Третій пункт містить опис узагальнених алгоритмів роботи системи та взаємодії елементів системи під час роботи програмного комплексу. Цей опис дозволяє нам зрозуміти загальний принцип роботи системи та взаємодію компонентів під час її функціонування.

У четвертому параграфі була надана детальна реалізація програмного комплексу. Були представлені фрагменти коду, які є складними з точки зору реалізації та вимагають пояснень. Посилання на додатки, де міститься відповідний код, дозволяють отримати додаткові пояснення та зрозуміти процес розробки.

Загалом, другий розділ нашої роботи надав детальну інформацію про вимоги до програмного продукту, фізичну схему даних, алгоритми роботи системи та реалізацію програмного комплексу. Ця інформація є основою для подальшого проектування та розробки системи.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ

3.1 Порядок встановлення та налаштування системи.

Веб-система потребує лише розгортання на сервері, користувачам лише потрібно подбати, щоб їх пристрій задовольняв мінімальні системні вимоги, що наведені в табл. 3.1.

Таблиця 3.1

Компонент	Мінімальні вимоги
Операційна система	Windows 7 або новіше, macOS 10.13 або новіше, Linux, iPadOS, Android 9.0
Процесор	Intel Core 2 Duo або еквівалентний
Оперативна пам'ять	Мінімум 1 ГБ RAM
Відеокарта	Сумісна з DirectX 10 або OpenGL 3.3
Роздільна здатність екрану	Мінімум 412х915 пікселів
Браузер	Google Chrome, Mozilla Firefox, Safari, Microsoft Edge з версії, що підтримує JavaScript

3.2 Структура інтерфейсу веб-системи обліку для кухні їдальні на підприємстві

Система працює за схемою, зображеною на рис. 3.1.

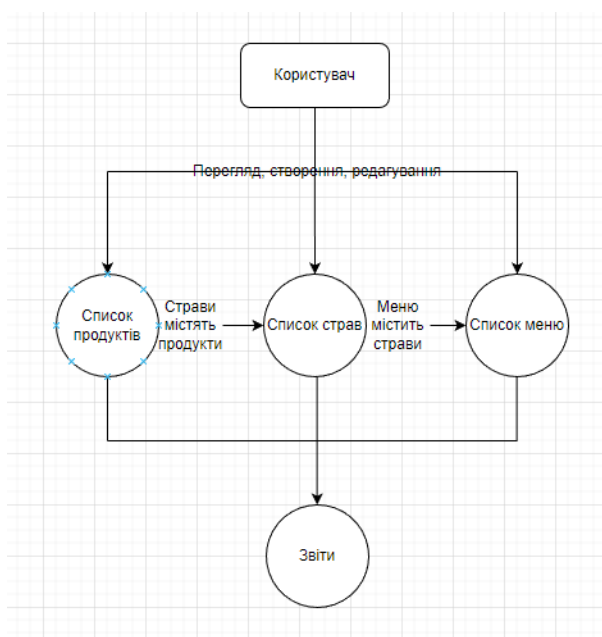


Рис. 3.1 – Схема роботи веб-системи

На головній сторінці користувач бачить сторінку, що складається з 3х кроків: «Налаштуйте меню» (рис. 3.2), «Додайте страви» (рис. 3.3), «Завантажте меню» (рис. 3.4).

МенюПродуктиСклад

1Налаштуйте меню

2Додайте страви

3Завантажте меню

Оберіть дату*

18.05.2023

Ранкове

Кількість порцій*

100

☐ Пільгове меню

Далі →

Рис. 3.2 – Сторінка «Налаштуйте меню»

МенюПродуктиСклад

1Налаштуйте меню

2Додайте страви

3Завантажте меню

← Назад

+

Далі →

Рис. 3.3 – Сторінка «Додайте страви»

МенюПродуктиСклад

1Налаштуйте меню

2Додайте страви

3Завантажте меню

← Назад

Експортувати меню в DOC

Рис. 3.4 – Сторінка «Завантажте меню»

Детальніше зупинимось на сторінці «Додайте страви» (рис. 3.3). Сторінка дозволяє додавати страви до меню. При натисканні кнопки «Додати» відкривається діалог додавання страви (рис. 3.5).

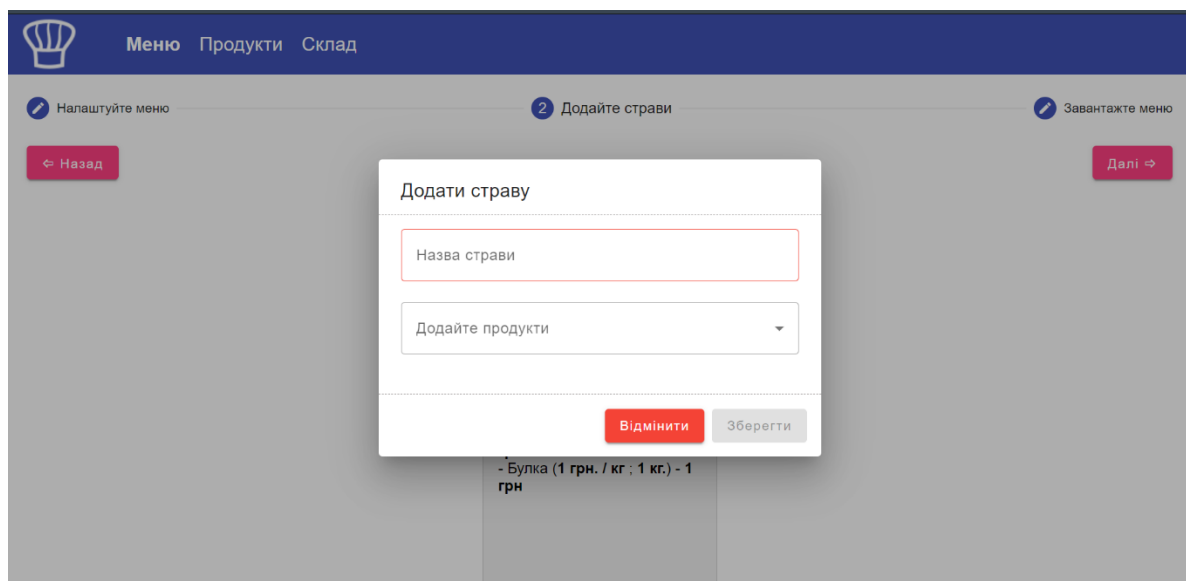


Рис. 3.5 – Діалог додавання страви

Форма створення страви є динамічною, тому заповнимо назву страви та додамо продукти. Контрол додавання продуктів, що передбачає вибір кількох варіантів та пошук по усім варіантам, зображений на рис. 3.6.

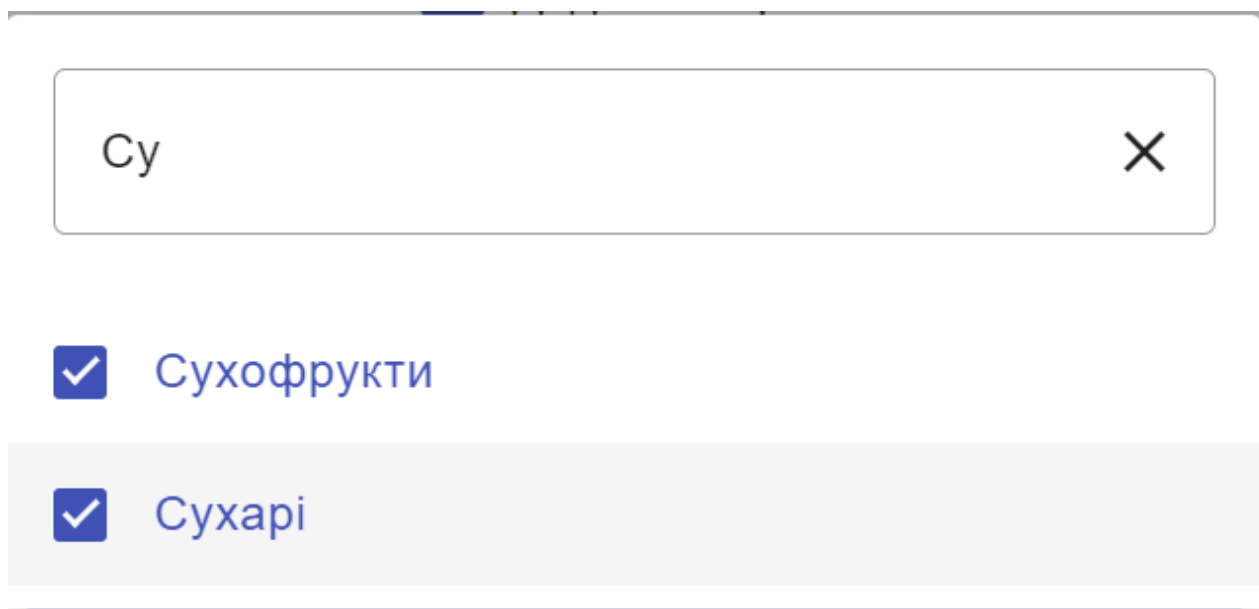


Рис. 3.6 – Контрол додавання декількох продуктів до страви з можливістю пошуку

В результаті форма розгортається та створює ще стільки форм, скільки продуктів вибрано користувачем (рис. 3.7).

Рис. 3.7 – Додаткові поля для введення ваги та ціни продукту для кожного вибраного на формі пролуку

Після створення страви, під кнопкою додавання можемо бачити картку страви (рис. 3.8).

Рис. 3.8 – Картка доданої страви

Картка має 2 кнопки: редагування та видалення страви. При натисканні на редагування, можемо побачити такий діалог, як при додавання, але з іншим заголовком (рис. 3.9).

Рис. 3.9 – Діалог редагування страви

При натисканні на кнопку видалення, отримаємо повідомлення з підтвердженням видалення (рис. 3.10).

Рис. 3.10 – Підтвердження видалення страви

Наступний крок «Завантажте меню» представляє собою сторінку з кнопкою (рис. 3.11), яка завантажує .docx файл з новоствореним меню.

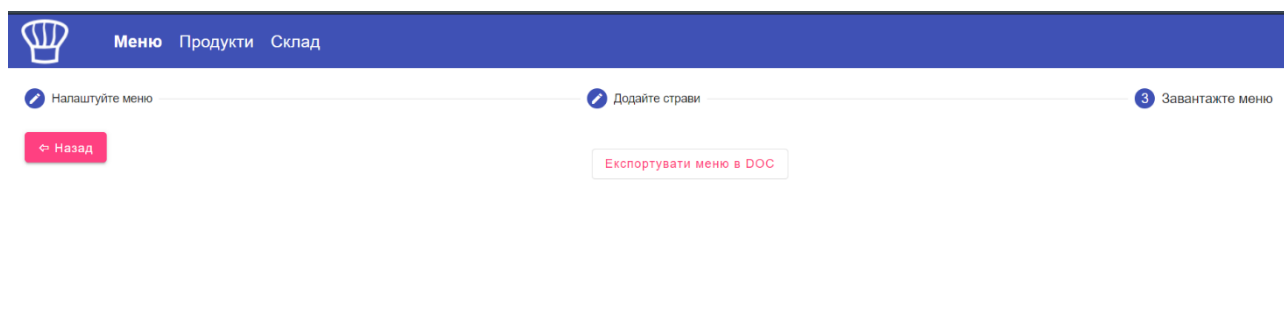


Рис. 3.11 – Сторінка завантаження меню

Після натискання кнопки «Експортувати меню в DOC», завантажиться файл з заповненими 2 однаковими таблицями «Меню» (рис. 3.12). Це було зроблено за вимогою комірника, так як меню завжди потрібне в двох екземплярах, а розміщення 2 меню на 1 сторінці формату А4 дозволить зекономити папір.

Затверджено
Директор ЛПЦЕЮ №4
Юлія БАБІЙ

МЕНЮ

На «18» травня 2023 р

Назва страви	Вихід (в гр)	Ціна (в грн)
Тестова страва	310	30.20

Кухар _____
Калькулятор _____

Затверджено
Директор ЛПЦЕЮ №4
Юлія БАБІЙ

МЕНЮ

На «18» травня 2023 р

Назва страви	Вихід (в гр)	Ціна (в грн)
Тестова страва	310	30.20

Кухар _____
Калькулятор _____

Рис. 3.12 – Заекспортований doc файл

Далі покажемо роботу допоміжних сторінок: продукти та склад.

Сторінка «Продукти» (рис. 3.13) представляє таблицю з можливістю додавати новий продукт, редагувати доданий та видаляти продукти.

 Меню Продукти Склад +			
Ідентифікатор	Код	Назва продукту	
1	102	Хліб	 
2	103	Булка	 
3	610301	Мука пшенична	 
4	611701	М'ясо ялов.	 
5	611702	М'ясо свин.	 
6	611201	Масло вершкове	 
7	611301	Олія соняшн.	 
8	611501	Молоко свіже	 
9	611901	Сметана	 
10	611601	Творог	 
11	612001	Сир тверд.	 
12	611401	Яйце	 

Рис. 3.13– Сторінка «Продукти»

При натисканні кнопки додавання, відкриється діалог додання продукту (рис. 3.14).

Додати продукт

Рис. 3.14 – Діалог додавання продукту

Натискання на редагування продукту відкриває діалог редагування (рис. 3.15).

Рис. 3.15 – Діалог редагування продукту

Наступною сторінкою є «Склад», де червоним підсвічені продукти, які відсутні на складі (рис. 3.16).

Меню Продукти Склад				
Ідентифікатор	Код	Назва продукту	Залишилось, кг.	
1	102	Хліб	115	
2	103	Булка		
3	610301	Мука пшенична		
4	611701	М'ясо ялов.		
5	611702	М'ясо свин.		
6	611201	Масло вершкове		
7	611301	Олія соняшн.		
8	611501	Молоко свіже		
9	611901	Сметана		
10	611601	Творог		
11	612001	Сир тверд.		
12	611401	Яйце		
13	610801	Сік фрукт.		

Рис. 3.16 – Сторінка «Склад»

3.3 Тестування роботи системи

У цьому розділі описуються методики, інструменти та процедури, використовувані для проведення тестування.

Планування тестування

У даному розділі висвітлюються стратегії, інструменти та процедури, які використовуються для проведення тестування. Описується процес розробки плану тестування, який включає в себе стратегію, обсяг та послідовність проведення тестів. Також визначаються основні функціональні блоки системи, які підлягають перевірці, а також критичні сценарії використання, які мають бути покриті тестами.

Функціональне тестування

Функціональне тестування спрямоване на перевірку відповідності функціональних можливостей системи вимогам та очікуванням користувачів. У цьому розділі описуються сценарії тестування, які включають перевірку реєстрації користувачів, пошуку та фільтрації профілів, взаємодії з повідомленнями тощо. Для цього використовуються автоматизовані тести, ручне тестування та моделювання реальних сценаріїв [30].

Тестування продуктивності

У цьому розділі проводиться оцінка продуктивності системи при різних навантаженнях та обсягах даних. Виконуються тести на швидкодію веб-сторінок, відповідь сервера, час завантаження профілів користувачів та інші важливі метрики продуктивності. Для вимірювання та аналізу продуктивності використовуються інструменти, такі як Apache JMeter, LoadRunner тощо [28].

Тестування безпеки

У цьому розділі проводиться перевірка системи на наявність потенційних вразливостей та заходів безпеки. Виконуються тести на перехоплення трафіку, захист від SQL-ін'єкцій, кросс-сайтових скриптів, аутентифікації та авторизації, обмеження доступу до конфіденційних даних тощо. Для цього використовуються відомі методи тестування безпеки, такі як пенетраційне тестування та сканування вразливостей [29].

Нижче наведено набір тест-кейсів для роботи основних модулів додатку.

Модуль: **Створення меню за датою та типом**

Тест-кейс: **Створення меню на конкретну дату**

Кроки:

- Вибрати дату, для якої потрібно створити меню.
- Вибрати тип меню (наприклад, сніданок, обід, вечеря).
- Додати страви до меню для вибраної дати та типу.

Очікуваний результат: Меню успішно створено для вибраної дати та типу.

Тест-кейс: **Валідація дати та типу меню**

Кроки:

- Вибрати неправильну або недопустиму дату.
- Вибрати неправильний або недопустимий тип меню.
- Спостерігати, як система обробляє недійсні дані.

Очікуваний результат: Система відхиляє неправильні дати та типи меню.

Модуль: **Завантаження меню**

Тест-кейс: **Завантаження повного меню**

Кроки:

- Вибрати опцію завантаження повного меню.
- Спостерігати, як система завантажує всі страви та інформацію про них.

Очікуваний результат: Повне меню успішно завантажено.

Тест-кейс: **Завантаження меню за конкретну дату**

Кроки:

- Вибрати дату, для якої потрібно завантажити меню.
- Спостерігати, як система завантажує страви, відповідні вибраній даті.

Очікуваний результат: Меню для вибраної дати успішно завантажено.

Модуль: **Створення страви**

Тест-кейс: **Створення нової страви**

Кроки:

- Вибрати опцію створення нової страви.
- Ввести назву, опис, інгредієнти та інші деталі страви.
- Зберегти страву.

Очікуваний результат: Нова страву успішно створена і збережена в системі.

Тест-кейс: **Валідація обов'язкових полів**

Кроки:

- Спробувати створити страву, не заповнивши обов'язкові поля (наприклад, назву).
- Спостерігати, як система обробляє недостатньо заповнену інформацію.

Очікуваний результат: Система не дозволяє створити страву без обов'язкових полів.

Модуль: **Редагування страви**

Тест-кейс: **Редагування існуючої страви**

Кроки:

- Вибрати опцію редагування існуючої страви.
- Внести зміни в назву, опис, інгредієнти або інші деталі страви.
- Зберегти зміни.

Очікуваний результат: Зміни внесені до страви успішно збережені в системі.

Тест-кейс: **Валідація змінених полів**

Кроки:

- Спробувати редагувати страву, внісши неправильну або недопустиму інформацію в поля.
- Спостерігати, як система обробляє недійсні дані.

Очікуваний результат: Система не дозволяє зберегти зміни з неправильними або недопустимими даними.

Модуль: **Видалення страви**

Тест-кейс: **Видалення існуючої страви**

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						52

Кроки:

- Вибрати опцію видалення існуючої страви.
- Підтвердити видалення страви.

Очікуваний результат: Страва успішно видалена з системи.

Модуль: **Створення продукту**

Тест-кейс: **Створення нового продукту**

Кроки:

- Вибрати опцію створення нового продукту.
- Ввести назву, опис та інші деталі продукту.
- Зберегти продукт.

Очікуваний результат: Новий продукт успішно створений та збережений в системі.

Тест-кейс: **Валідація обов'язкових полів**

Кроки:

- Спробувати створити продукт, не заповнивши обов'язкові поля (наприклад, назву).
- Спостерігати, як система обробляє недостатньо заповнену інформацію.

Очікуваний результат: Система не дозволяє створити продукт без обов'язкових полів.

Модуль: **Редагування продукту**

Тест-кейс: **Редагування існуючого продукту**

Кроки:

- Вибрати опцію редагування існуючого продукту.
- Внести зміни в назву, опис та інші деталі продукту.
- Зберегти зміни.

Очікуваний результат: Зміни внесені до продукту успішно збережені в системі.

Тест-кейс: **Валідація змінених полів**

Кроки:

- Спробувати редагувати продукт, внісши неправильну або недопустиму інформацію в поля.
- Спостерігати, як система обробляє недійсні дані.

Очікуваний результат: Система не дозволяє зберегти зміни з неправильними або недопустимими даними.

Модуль: **Експорт меню**

Тест-кейс: **Експорт меню в DOC-файл**

Кроки:

- Вибрати опцію експорту меню в DOC-файл.
- Перевірити, чи успішно згенерований DOC-файл з відповідним меню.

Очікуваний результат: DOC-файл з експортованим меню успішно згенерований та містить відповідні дані.

Результати тестування

Процес тестування підтвердив коректну реалізацію та відповідність вимогам основних функціональних можливостей веб-системи обліку для кухні їдальні на підприємстві. Усі виконані тест-кейси були успішними, що свідчить про правильну роботу системи та її здатність виконувати необхідні завдання.

Висновки до розділу 3

У цьому розділі було розглянуто процес встановлення та конфігурування системи, включаючи параметри доступу та підключення до бази даних. Ці кроки дозволяють забезпечити належне функціонування системи та готовність до використання.

Далі, була детально розглянута структура інтерфейсу веб-орієнтованої системи з точки зору користувача. Були описані основні елементи інтерфейсу, їх функціональні можливості та способи взаємодії з системою. Це сприяє зручному та ефективному використанню системи користувачами.

Окрім цього, було проведене тестування програмного комплексу з метою перевірки правильності реалізації функціональних та нефункціональних вимог,

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						54

а також виявлення можливих помилок та недоліків. Цей процес дозволяє переконатися, що програмний комплекс працює стабільно, відповідає вимогам та відповідає потребам користувачів.

В результаті проведення цього розділу можна зробити висновок, що веб-система готова до використання, налаштований з урахуванням потреб користувачів та пройшов успішне тестування.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було спроектовано та реалізовано веб-систему обліку для кухні їдальні на підприємстві:

- Проведено аналіз функціональних вимог та детальне дослідження архітектурних патернів та інструментальних засобів для розробки веб-системи обліку для кухні їдальні на підприємстві.
- В рамках аналізу предметної області було визначено основні функціональні можливості, які має мати веб-система обліку для кухні їдальні на підприємстві, такі як створення меню, страв, продуктів, робота зі складом продуктів.
- Були оглянуті існуючі на ринку аналоги, з'ясовано їхні переваги та недоліки, а також виявлено проблеми їх використання для нашого випадку.
- Обґрунтовано вибір інструментальних засобів.
- Для забезпечення ефективного зберігання та отримання даних була розроблена база даних системи. Використання бази даних Superbase було обрано з метою спрощення розгортання проекту, так як вона є хмарною.
- Проектування та реалізація алгоритмів роботи системи були проведені з урахуванням функціональних вимог, таких як створення меню, страв, продуктів, експорт меню. Були розроблені алгоритми, які забезпечують ефективну та зручну роботу системи.
- Було проведено аналіз інтерфейсу веб-додатка з урахуванням потреб користувачів. Описано основні компоненти інтерфейсу, їх функціональні можливості та способи взаємодії з системою. Це сприяє зручному та ефективному користуванню системою користувачами.
- Метою проведеного тестування програмного комплексу було перевірити, чи відповідає він функціональним та нефункціональним вимогам, а також виявити можливі помилки та недоліки. Це дозволяє переконатися, що програмний комплекс працює стабільно, відповідає вимогам та задовольняє потреби користувачів. Реалізований програмний комплекс готовий до використання.

Розроблена система веб-система обліку для кухні їдальні пропонує базовий набір функціональності, що включає створення меню, створення, редагування та видалення страв, створення, редагування та видалення продуктів та експорт меню. Однак, ця система має потенціал для подальшого розширення та вдосконалення. Може бути додана можливість експортувати інші дані, також можна додати імпорт меню та інших параметрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Посадова інструкція комірника в закладі освіти [Електронний ресурс] – Режим доступу до ресурсу: <https://ditsad.com.ua/doc20.htm>
2. Poster [Електронний ресурс] – Режим доступу до ресурсу: <https://joinposter.com/ua>
3. IBM: Тришарова архітектура [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ibm.com/topics/three-tier-architecture>
4. Medium: the Three Layered Architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/@deanrubin/the-three-layered-architecture-fe30cb0e4a6>
5. Canarys: 3 Layered Architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ecanarys.com/Blogs/ArticleID/76/3-Layered-Architecture>
6. N-iX: Microservices vs monolith: Which architecture is the best choice for your business? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/>
7. TechTarget: Monolith Architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techtarget.com/whatis/definition/monolithic-architecture>
8. Microsoft Learn: Microservice Architecture style [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>
9. Microservice Architecture: Microservice Architecture Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://microservices.io/patterns/microservices.html>
10. Microservice Architecture: Microservice Architecture Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://microservices.io/patterns/microservices.html>

11. Amazon AWS: SOA [Электронный ресурс] – Режим доступа до ресурсу:
https://aws.amazon.com/what-is/service-oriented-architecture/?nc1=h_ls
12. TechTarger: SOA [Электронный ресурс] – Режим доступа до ресурсу:
<https://www.techtarget.com/searchapparchitecture/definition/service-oriented-architecture-SOA>
13. GeekForGeeks: SOA [Электронный ресурс] – Режим доступа до ресурсу:
<https://www.geeksforgeeks.org/service-oriented-architecture/>
14. Angular Docs – Режим доступа до ресурсу: <https://angular.io/docs>
15. Dev Docs : Angular – Режим доступа до ресурсу: <https://devdocs.io/angular/>
16. Firebase: Docs [Электронный ресурс] – Режим доступа до ресурсу:
<https://firebase.google.com/docs?hl=en>
17. Techtarget: Google Firebase [Электронный ресурс] – Режим доступа до ресурсу:
<https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase>
18. Medium: What is Firebase? [Электронный ресурс] – Режим доступа до ресурсу:
<https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>
19. Educative: What is Firebase [Электронный ресурс] – Режим доступа до ресурсу: <https://www.educative.io/answers/what-is-firebase>
20. Supabase: Docs [Электронный ресурс] – Режим доступа до ресурсу:
<https://supabase.com/docs>
21. LogRocket: Firebase vs. Supabase [Электронный ресурс] – Режим доступа до ресурсу: <https://blog.logrocket.com/firebase-vs-supabase-choosing-right-tool-project/>
22. HappyTeam: Supabase – open source BaaS, can it work? [Электронный ресурс] – Режим доступа до ресурсу: <https://happyteam.io/blog/supabase-open-source-baas-can-it-work/>
23. VS Code [Электронный ресурс] – Режим доступа до ресурсу:
<https://code.visualstudio.com/docs>

24. Medium: VS Code [Электронный ресурс] – Режим доступа до ресурсу: <https://medium.com/@NowPixels/writing-documentation-in-vscode-a-handy-beginners-guide-57ad510fac99>
25. Flutter: Docs VS CODE [Электронный ресурс] – Режим доступа до ресурсу: <https://docs.flutter.dev/tools/vs-code>
26. Git: Docs [Электронный ресурс] – Режим доступа до ресурсу: <https://git-scm.com/docs>
27. Atlassian: Git [Электронный ресурс] – Режим доступа до ресурсу: <https://www.atlassian.com/git>
28. Tricentis: Performance testing [Электронный ресурс] – Режим доступа до ресурсу: <https://www.tricentis.com/learn/performance-testing>
29. Geeks for Geeks: Software testing | Security testing [Электронный ресурс] – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/software-testing-security-testing/>
30. BrowserStack: Functional Testing [Электронный ресурс] – Режим доступа до ресурсу: <https://www.browserstack.com/guide/functional-testing>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						61

Сервіс експорту

export.service.ts

```
import { Injectable } from '@angular/core';
import { IPatch, PatchType, TextRun, UnderlineType, patchDocument } from 'docx';
import { saveAs } from 'file-saver';

import { HttpClient } from '@angular/common/http';
import { ExportMenu } from './export-menu.model';
import { Observable, mergeMap, of } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ExportService {

  constructor(private httpClient: HttpClient) { }

  font: string = 'Times New Roman';

  public downloadDocument(exportMenu: ExportMenu): Observable<void> {
    const patches = this.exportMenuPatches(exportMenu);
    const templatePath = 'assets/menu_template.docx';
    const documentName =
`_${exportMenu.day}_${exportMenu.month}_${exportMenu.year}.docx`;

    return this.httpClient.get(templatePath, { responseType: 'arraybuffer' })
      .pipe(
        mergeMap(data => patchDocument(data, { patches})),
        mergeMap(doc => of(saveAs(new Blob([doc]), documentName)))
      );
  }

  exportMenuPatches(exportMenu: ExportMenu): { [key: string]: IPatch } {
    const dishes = exportMenu.dishes;
    let obj: any = {
      day: exportMenu.day.toString(),
      month: exportMenu.month,
      year: exportMenu.year.toString()
    };
  }
```

```

let i = 0;
dishes.forEach(dish => {
    i++;
    obj[`dish_name_${i}`] = dish.name.toString();
    obj[`w${i}`] = dish.weight.toString();
    obj[`p${i}`] = dish.price.toFixed(2).toString();
});

while (i++ <= 8) {
    obj[`dish_name_${i}`] = '';
    obj[`w${i}`] = '';
    obj[`p${i}`] = '';
}

return this.getPatches(obj, { day: true, month: true });
}

private getPatches(fields: { [key: string]: string }, underlined: any = null) {
    const patches: { [key: string]: IPatch } = {};

    for (const field in fields) {
        patches[field] = {
            type: PatchType.PARAGRAPH,
            children: [new TextRun({ text: fields[field], font: this.font, underline:
underlined[field] ? {type: UnderlineType.SINGLE} : undefined })],
        };
    }

    return patches;
}
}

```

Діалог додавання страви

add-dish-dialog.ts

```
import { ChangeDetectionStrategy, Component, Inject } from '@angular/core';
import { FormArray, FormBuilder, FormControl, FormGroup, Validators } from
'@angular/forms';
import { MAT_DIALOG_DATA, MatDialogRef } from '@angular/material/dialog';
import { Observable, Subscription } from 'rxjs';
import { Dish } from 'src/models/dish.model';
import { DishProduct } from 'src/models/dish.product.model';
import { Product } from 'src/models/product.model';

@Component({
  selector: 'app-add-dish-dialog',
  templateUrl: './add-dish-dialog.component.html',
  styleUrls: ['./add-dish-dialog.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class AddDishDialogComponent {
  form: FormGroup;
  private _productsChangeSubscription: Subscription;
  title: string = '';

  get productsArray(): FormArray{
    return this.form.controls['productsFormArray'] as FormArray;
  }

  get productsControl(): FormControl{
    return this.form.controls['products'] as FormControl;
  }

  get products(): Product[]{
    return this.productsControl.value;
  }

  get idControl(): FormControl{
    return this.form.controls['id'] as FormControl;
  }

  get nameControl(): FormControl{
    return this.form.controls['name'] as FormControl;
  }
}
```

```

    }

    get name(){
        return this.nameControl?.value;
    }

    get id(){
        return this.idControl.value;
    }

    get formInvalid(){
        return this.form.invalid;
    }

    constructor(
        private fb: FormBuilder,
        private dialogRef: MatDialogRef<AddDishDialogComponent>,
        @Inject(MAT_DIALOG_DATA) public data: any
    ) { }

    ngOnInit() {
        this.form = this.fb.group({
            id: [''],
            name: ['', Validators.required], //DON'T FORGET TO ADD VALIDATORS.PATTERN
            products: [[], Validators.required],
            productsFormArray: this.fb.array([])
        });

        if(this.data.dish){
            this.title = 'Редагувати страву';
            this.initForm(this.data.dish);
        }

        this.title = 'Додати страву';

        setTimeout(()=>{
            this._productsChangeSubscription =
this.productsControl.valueChanges.subscribe((p: Product[]) => {
                this.updateProductControls(p);
            });
        });
    }

```

```

    }

    onCancel(): void {
        this.dialogRef.close();
    }

    onSave(): void {
        const dish = this.createDish();
        this.dialogRef.close(dish);
    }

    onDeleteDishProduct(i: number){;
    }

    private addProduct(product: Product, emitEvent: boolean = true, weight: string |
number = '', price: string | number = '') {
        this.productsArray.push(this.createProductForm(product, weight, price),
{emitEvent: emitEvent});
    }

    private updateProductControls(products: Product[]){
        products.forEach(p => {
            if(!this.productsArray.value.find((x: any) => x.name == p.name)){
                this.addProduct(p);
            }
        });

        this.productsArray.value.forEach((pc: any, i: number) => {
            if(!products.find((x) => x.name == pc.name)){
                this.productsArray.removeAt(i);
            }
        })
    }

    private initForm(dish: Dish){
        this.idControl.setValue(dish.id);
        this.nameControl.setValue(dish.name, {emitEvent: false});
        this.productsControl.setValue(dish.products.map(p => new Product(p.product)));
        dish.products.forEach(dp => {
            this.addProduct(dp.product, false, dp.weight, dp.price);
        });
    }

```

```

    }

    private createProductForm(product: Product, weight: string | number = '', price:
string | number = ''): FormGroup{
        const productForm = this.fb.group({
            id: [product.id],
            name: [product.name],
            weight: [weight, Validators.required],
            price: [price, Validators.required]
        });

        return productForm;
    }

    private createDish(): Dish{
        let dishProducts = this.productsArray.value;
        dishProducts.map((dp: DishProduct, i: number) => {
            dp.productId = this.products[i].id;
            dp.product = this.products[i];
        });
        let dish = new Dish();
        if(this.id){
            dish.id = this.id;
        }
        dish.products = dishProducts;
        dish.name = this.name;

        return dish;
    }

    ngOnDestroy(){
        this._productsChangeSubscription.unsubscribe();
    }
}

```

add-dish-dialog.html

```

<h1 mat-dialog-title class="dialog-title">{{title}}</h1>
<div mat-dialog-content class="dialog-content">
    <form [formGroup]="form">
        <mat-form-field class="w-100">
            <input matInput formControlName="name" placeholder="Назва страви">
        </mat-form-field>

```

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	
						67

```

<app-products-select [data]="data.products" formControlName="products">
</app-products-select>

<div formArrayName="productsFormArray">
  <div *ngFor="let product of productsArray.controls; let i=index"
[formGroupName]="i">
    <div class="flex">
      <label>{{product.value.name}}</label>
      <!-- <mat-icon class="close-icon"
(click)="onDeleteDishProduct(i)">close</mat-icon> -->
    </div>
    <div class="flex space-between">
      <mat-form-field class="product-prop" placeholder="Вара продукту (кг)">
        <mat-label>Вара продукту (кг)</mat-label>
        <input type="number" min="0" placeholder="Вара продукту
(кг)" matInput id="productWeight{{i}}" formControlName="weight">
      </mat-form-field>
      <mat-form-field class="w-50">
        <mat-label>Ціна продукту (грн)</mat-label>
        <input type="number" min="0" matInput id="productPrice{{i}}"
formControlName="price" placeholder="Ціна продукту (грн)">
      </mat-form-field>
    </div>
  </div>
</div>
</form>
</div>

<div mat-dialog-actions class="dialog-buttons flex end">
  <button mat-raised-button (click)="onCancel()" color="warn">Відмінити</button>
  <button mat-raised-button color="primary" (click)="onSave()"
[disabled]="formInvalid">Зберегти</button>
</div>

```