

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри )

## Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОГО  
РЕСУРСУ ДЛЯ АДМІНІСТРУВАННЯ ПОСЛУГ З КЕЙТЕРИНГУ  
SOFTWARE IMPLEMENTATION OF AN  
INFORMATION RESOURCE TO MANAGE THE CATERING SERVICES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-229

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ВАЩЕНКО М.І.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
**Національний університет «Запорізька політехніка»**  
 (повне найменування закладу вищої освіти)

Факультет КНТ  
 Кафедра програмних засобів  
 Ступінь вищої освіти бакалавр  
 Спеціальність 121 Інженерія програмного забезпечення  
(код і найменування)  
 Освітня програма (спеціалізація) Інженерія програмного забезпечення  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.  
Сергій СУББОТІН  
 “      ”                      2023 року

**З А В Д А Н Н Я**

**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

ВАЩЕНКА Михайла Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інформаційного ресурсу для адміністрування послуг з кейтерингу. Software Implementation of an Information Resource to Manage the Catering Services

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,  
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

2. Строк подання студентом проєкту (роботи) 01 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | ПРІЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | ГОФМАН Є.А., доцент                       |                |                           |
| Нормоконтроль       | ЛИПОВЕЦЬ М.В., асистент                   |                |                           |

7. Дата видачі завдання “04” квітня 2023 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                             | Строк виконання етапів проєкту (роботи) | Примітка     |
|-------|----------------------------------------------------------------------|-----------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                          | 1 тиждень                               | Завдання, ТЗ |
| 2     | Аналіз предметної області.                                           | 1 тиждень                               | Розділ 1     |
| 3     | Вибір мови програмування та інших технологій розробки.               | 2 тиждень                               | Розділ 2     |
| 4     | Розробка структури програми.                                         | 2 тиждень                               | Розділ 3     |
| 5     | Розробка програми.                                                   | 3-4 тижні                               | Розділи 3, 4 |
| 6     | Тестування та експериментальне дослідження програмного забезпечення. | 5 тиждень                               | Розділ 4     |
| 7     | Оформлення пояснювальної записки та документів до неї.               | 6 тиждень                               | Додатки      |
| 8     | Нормоконтроль та рецензування.                                       | 7 тиждень                               |              |
| 9     | Захист роботи.                                                       | 8 тиждень                               |              |

Студент(ка)

\_\_\_\_\_  
( підпис )      Михайло ВАЩЕНКО  
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_  
( підпис )      Євгеній ГОФМАН  
(Ім'я ПРІЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:  
97 с., 3 табл., 40 рис., 4 дод., 25 джерел.

ІНФОРМАЦІЙНИЙ РЕСУРС, КЕЙТЕРИНГ, ГРОМАДСЬКЕ  
ХАРЧУВАННЯ, СФЕРА ПОСЛКГ, РОЗРОБКА, ВЕБЗАСТОСУНОК, МОВА  
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес програмної реалізації інформаційного ресурсу для адміністрування послуг з кейтерингу.

Предмет дослідження – програмні засоби для реалізації інформаційного ресурсу для адміністрування послуг з кейтерингу.

Метою роботи є програмна реалізація інформаційного ресурсу для адміністрування послуг з кейтерингу у вигляді кроссплатформного веб-застосунку.

Матеріали, методи та технічні засоби: мова програмування PHP та фреймворк Symfony.

Результати. Розроблено інформаційний ресурсу для адміністрування послуг з кейтерингу.

Практична цінність роботи полягає у програмній реалізації інформаційного ресурсу для адміністрування послуг з кейтерингу у вигляді кроссплатформного вебзастосунку.

Висновки. Реалізовано інформаційного ресурсу для адміністрування послуг з кейтерингу у вигляді кроссплатформного вебзастосунку.

Галузь використання – підприємства та працівники, організатори свят.

## **ABSTRACT**

Explanatory note to the diploma qualifying work of the bachelor: 97 pages, 3 tables, 40 figures, 4 appendixes, 25 sources.

INFORMATION RESOURCE, CATERING, PUBLIC CATERING, POSLKG SPHERE, DEVELOPMENT, WEB APPLICATION, PROGRAMMING LANGUAGE, SOFTWARE

The object of research is the process of software implementation of an information resource for the administration of catering services.

The subject of the research is software tools for implementing an information resource for the administration of catering services.

The aim of the work is to provide a software implementation of an information resource for the administration of catering services in the form of a cross-platform web application.

Materials, methods, and technical tools: the PHP programming language and the Symfony framework.

Results. An information resource has been developed for the administration of catering services.

The practical value of the work lies in the software implementation of an information resource for the administration of catering services in the form of a cross-platform web application.

Conclusions. Implemented an information resource for the administration of catering services in the form of a cross-platform web application.

Scope of use: enterprises and employees, holiday organizers.

## ЗМІСТ

|                                                                   |    |
|-------------------------------------------------------------------|----|
|                                                                   | С. |
| Перелік скорочень та умовних позначок .....                       | 8  |
| Вступ.....                                                        | 9  |
| 1 Аналіз предметної області .....                                 | 13 |
| 1.1 Кейтерингове підприємництво .....                             | 13 |
| 1.2 Організація та адміністрування застосунку для кейтерингу..... | 16 |
| 1.3 Порівняння існуючих аналогів.....                             | 18 |
| 1.3.1 Служба «Живіт-кейтеринг» .....                              | 19 |
| 1.3.2 Сервіс «Glovo» .....                                        | 20 |
| 1.3.3 Сервіс «Bolt Food» .....                                    | 22 |
| 1.3.4 Порівняння існуючих аналогів .....                          | 24 |
| 1.4 Постановка завдання .....                                     | 25 |
| 1.4.1 Межі проєкту .....                                          | 25 |
| 1.4.2 Функціональність системи.....                               | 25 |
| 1.4.3 Вимоги до інтерфейсу .....                                  | 26 |
| 1.4.4 Вимоги до продуктивності.....                               | 26 |
| 1.4.5 Вимоги до безпеки .....                                     | 27 |
| 1.4.6 Експлуатаційні вимоги .....                                 | 27 |
| 1.4.7 Не функціональні вимоги до продукту.....                    | 27 |
| 1.5 Висновки за розділом 1 .....                                  | 28 |
| 2 Матеріали і методи .....                                        | 29 |
| 2.1 Вибір мови програмування та інших технологій розробки .....   | 29 |
| 2.2 Вибір та обґрунтування фреймворку.....                        | 32 |
| 2.3 Структурна схема розробки.....                                | 35 |
| 2.4 Висновки за розділом 2 .....                                  | 36 |
| 3 Основні рішення щодо реалізації компонентів системи .....       | 37 |
| 3.1 Проєктування моделей системи .....                            | 37 |
| 3.2 Проєктування інтерфейсу .....                                 | 53 |

|                                                                          |    |
|--------------------------------------------------------------------------|----|
| 3.3 Висновки за розділом 3 .....                                         | 57 |
| 4 Експлуатація, тестування та експериментальне дослідження програми..... | 58 |
| 4.1 Призначення й умови застосування програми .....                      | 58 |
| 4.2 Умови виконання програми.....                                        | 59 |
| 4.3 Інструкція по експлуатації програми.....                             | 60 |
| 4.3.1 Характеристики програми.....                                       | 60 |
| 4.3.2 Звернення до програми.....                                         | 61 |
| 4.3.3 Вхідні й вихідні дані .....                                        | 61 |
| 4.4 Методика та результати тестування .....                              | 61 |
| Висновки .....                                                           | 64 |
| Перелік джерел посилань .....                                            | 65 |
| Додаток А Технічне завдання .....                                        | 68 |
| Додаток Б Опис програми .....                                            | 72 |
| Додаток В Текст програми .....                                           | 76 |
| Додаток Д Слайди презентації .....                                       | 90 |

## **ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

UML – Unified Modeling Language;

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СКБД – система керування базами даних.

## ВСТУП

Розробка вебзастосунок для кейтерингової служби може бути нагальним завданням з кількох причин. Слід відзначити деякі ключові причини, чому це так [1], [2].

Підприємства громадського харчування чутливі до часу: підприємства громадського харчування працюють у швидко мінливому та чутливому до часу середовищі. Постачальники продовольства повинні мати можливість швидко та ефективно керувати замовленнями, меню та доставкою, а також переконатися, що продукти свіжі та доставлені вчасно. Вебзастосунок, що дозволяє легко замовляти та відстежувати доставку, може значно підвищити ефективність та точність роботи підприємств громадського харчування [1], [2].

Очікування клієнтів: сучасні клієнти очікують зручності і простоти у використанні при замовленні страв. Вебзастосунок, який дозволяє клієнтам легко переглядати меню, розміщувати замовлення та відстежувати доставку, може значно підвищити задоволеність та утримання клієнтів [1], [2]. Крім того, клієнти все частіше використовують мобільні пристрої для доступу до Інтернету, а вебзастосунок, адаптований до мобільних пристроїв, може забезпечити доступність бізнесу громадського харчування для широкого кола клієнтів [1], [2].

Конкуренція: індустрія громадського харчування відрізняється високою конкуренцією, на ринок постійно виходять нові підприємства. Вебзастосунок, що забезпечує безперебійний процес замовлення та доставки, може дати підприємству громадського харчування конкурентну перевагу перед іншими підприємствами, які покладаються на традиційні методи оформлення замовлень [1], [2].

Управління даними: вебзастосунок також може надати цінну інформацію про поведінку та уподобання клієнтів, дозволяючи підприємствам громадського харчування приймати рішення на основі даних щодо пунктів меню, цін та маркетингових стратегій [1], [2]. Використовуючи аналітику

даних, підприємства громадського харчування можуть виявляти тенденції та закономірності в поведінці клієнтів та відповідно коригувати свої бізнес-стратегії [1], [2].

Масштабованість: вебзастосунок також може допомогти підприємствам громадського харчування масштабувати свої операції та обробляти великі обсяги замовлень [1], [2]. У міру зростання бізнесу вебзастосунок можна легко модифікувати та розширювати відповідно до нових меню, місць розташування та варіантів доставки [1], [2].

Безпека та гігієна: пандемія COVID-19 підкреслила важливість безпеки та гігієни у сфері громадського харчування. Вебзастосунок, що дозволяє здійснювати безконтактне замовлення та доставку, може допомогти забезпечити безпеку клієнтів та співробітників. Крім того, вебзастосунок може надавати клієнтам детальну інформацію про правила безпеки та гігієни громадського харчування, що може допомогти зміцнити довіру та лояльність [1], [2].

Підводячи підсумок, розробка вебзастосунку для кейтерингу може бути нагальним завданням через чутливий до часу характер індустрії громадського харчування, очікування клієнтів щодо зручності та простоти використання, конкуренції, управління даними, масштабованості та міркувань безпеки та гігієни. Добре розроблений вебзастосунок може значно підвищити ефективність, прибутковість та задоволеність клієнтів у сфері громадського харчування [1], [2].

Мета дипломної кваліфікаційної роботи бакалавра виконати програмну реалізацію інформаційного ресурсу для адміністрування послуг з кейтерингу.

Ось деякі з ключових завдань, які необхідно виконати, щоб вебзастосунок був високоякісним і відповідав потребам як громадського харчування, так і його клієнтів.

Визначення вимог: першим кроком у розробці вебпрограми є визначення вимог до програми. Це включає визначення функцій, якими повинен володіти застосунок для задоволення потреб підприємств кейтерингу та їх клієнтів. Це

може включати такі функції, як управління меню, онлайн-замовлення, відстеження доставки та обробка платежів.

Розробка інтерфейсу користувача: інтерфейс користувача вебпрограми має вирішальне значення для успіху програми. Інтерфейс користувача повинен бути простим у використанні, візуально привабливим та доступним на кількох пристроях. Дизайн інтерфейсу користувача повинен базуватися на дослідженнях та тестуванні користувачів, щоб гарантувати, що він відповідає потребам та очікуванням цільової аудиторії.

Розробка серверної частини: серверна частина вебдодатки відповідає за зберігання, обробку і витяг даних. Це включає розробку схеми бази даних, налаштування сценаріїв на стороні сервера та створення API, які будуть використовуватися для взаємодії між інтерфейсом та серверною частиною програми.

Перевірка та тестування застосунку: тестування є важливою частиною процесу розробки. Вебзастосунок повинен бути ретельно протестований, щоб переконатися, що він працює належним чином, безпечний і не містить помилок. Тестування повинно включати функціональне тестування, тестування зручності використання, тестування продуктивності та тестування безпеки.

Розгортання програми: як тільки застосунок буде протестовано та готове до роботи, його слід розгорнути у виробничому середовищі. Це включає налаштування серверної інфраструктури, налаштування бази даних та забезпечення доступності програми для користувачів.

Забезпечення поточного обслуговування та підтримки: після розгортання вебпрограми важливо забезпечити поточне обслуговування та підтримку, щоб гарантувати, що вона продовжує ефективно працювати. Це включає моніторинг програми на наявність помилок, застосування оновлень безпеки та надання технічної підтримки користувачам.

Таким чином, розробка якісного вебзастосунку для кейтерингу вимагає виконання цілого ряду завдань, включаючи визначення вимог, проєктування інтерфейсу користувача, розробку серверної панелі, тестування програми,

розгортання програми та забезпечення поточного обслуговування та підтримки. Ефективно виконуючи ці завдання, можна розробити високоякісне веб-застосунок, що відповідає потребам як підприємства громадського харчування, так і його клієнтів.

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перший розділ буде присвячено детальному ознайомленню із предметною областю надання послуг кейтерингу, через аналіз сфери послуг кейтерингу, аналіз існуючих рішень, виявлення та опис їх переваг та недоліків. Такий аналіз дозволить сформулювати низку вимог до нового рішення, що буде позбавлено існуючих недоліків та буде відрізнятися новим підходом до вирішення проблеми..

### 1.1 Кейтерингове підприємництво

Ідея кейтерингу здається досить простою та не хитрою – це коли шеф-кухар та його команда готують, розігрівають та подають їжу, чи не так, але насправді, це набагато більше [3]-[6].

Так, служби кейтерингу координують приготування, створення, доставку та презентацію страв для клієнтів. Наприклад, багато людей відвідують: весільні прийоми, благодійні збори коштів, репетиційні вечери та інші родинні свята, де були смачно приготовані та подані страви, швидше за все, ця подія обслуговувалася службами кейтерингу [3]-[6]. Додаткові обов'язки такої компанії на заходах часто включають оренду, встановлення та прибирання декорацій, столів, стільців, музики та освітлення [3]-[6].

Кейтеринг для заходів – існує три основних типи постачальників провізії:

- мобільний кейтеринг або доставки їжі;
- кейтеринг в готелях або ресторанах;
- приватні служби кейтерингу з повним спектром послуг.

Мобільний кейтеринг або доставка їжі. Цей тип організації харчування для заходів часто здійснюється у вигляді фуд-трака, візка з їжею або служби доставки [3]-[6]. Цей варіант часто є менш дорогим і менш трудомістким. Процес, в більшій чи меншій мірі залежить від потреб клієнта, включає в себе

узгодження заздалегідь визначеного меню і організацію доставки страв. Найчастіше потрібен мінімальний штат співробітників, що також знижує витрати [3]-[6].

Кейтеринг в готелях або ресторанах. Роль цього виду кейтерингових послуг полягає в тому, щоб виступати посередниками між кухонним персоналом і клієнтом при плануванні власного заходу. В їх обов'язки входить: розробка і підготовка меню, сервіровка столу, організація банкету, доставка декорацій, а також підбір персоналу в міру необхідності [3]-[6].

Приватні служби кейтерингу з повним спектром послуг. Всі приватні компанії з організації харчування заходів є незалежними підприємствами, що знаходяться у власності та управлінні. Приватні служби кейтерингу часто наймають для проведення ділових та особистих ювілеїв, вечірок на день народження, весільних прийомів, ділових свят, а також багатьох інших заходів [3]-[6]. Тип кейтерингової служби часто відповідає за всі аспекти планування події. Найчастіше це включає в себе (але не обмежується цим): розробку і приготування меню, створення і презентацію страв, прокат тарілок і столових приладів, а також підбір персоналу для підготовки до Заходу і прибирання приміщень [3]-[6].

Тепер, спираючись на вище наведену інформацію про кейтеринг, можна відмітити, що хороша служба кейтерингу повинна мати можливість швидко адаптуватися, оскільки як потреби клієнта, так і хід події постійно змінюються [3]-[6]. Протягом усього заходу команда кейтерингу відіграє важливу роль у бездоганному проведенні заходу, оскільки вони також є основною командою з прибирання та наведення порядку. Хороша команда кейтерингу забере всі брудні тарілки, напої без нагляду, підмітає розбите скло і навіть ретельно прибере приміщення в кінці вечора, щоб допомогти вам повернути свій депозит. Така відданість справі забезпечує безпеку заходу і його бездоганний зовнішній вигляд в будь-який час. Якщо все піде добре, клієнти навіть не будуть знати про всю закулісну роботу працівників кейтерингової служби [3]-[6].

Кейтеринг – еко-постачальники провізії. Від першої зустрічі клієнта та кейтерингової служби до великої події, обслуговування включає ретельне планування, підготовку, командну роботу, креативність та гнучкість [3]-[6].

Від концепції до останнього нез'їденого шматочка салату, що залишився у смітнику, команда кейтерингу тісно співпрацює зі своїм клієнтом, розробляючи на замовлення меню закусок, меню бару, меню вечері та, можливо, десертів, кави та пізніх закусок. Це включає обговорення персоналу, розкладу, презентації, договірних очікувань як весільної пари, так і кейтерингу, а також прибирання після заходу [3]-[6].

Крім того, деякі кейтеринг-компанії для заходів звертаються до своїх місцевих виробників органічних продуктів і резервують сезонні фрукти, овочі та інші продукти для майбутньої події. Сезонні страви часто визначають вибір меню як для гостей, так і для шеф-кухаря [3]-[6].

Кейтеринг також може допомогти клієнту забронювати оренду для проведення заходу. Сюди входять, але не обмежуються ними: скатертини, столи для коктейлів або подарунків, або тортів, стільці, посуд, Барні келихи, обігрівачі і так далі [3]-[6].

Як тільки всі деталі будуть узгоджені, кейтерингова компанія підготує проєкт контракту для розгляду. Хороша кейтерингова компанія буде прозорою щодо своїх витрат і пояснить клієнтам, чому вони прийняли ті чи інші рішення, наприклад, найняли певну кількість офіціантів або віддали перевагу одному вину перед іншим [3]-[6].

День заходу – до прибуття гостей керівник кейтерингу і його команда докладуть всіх зусиль, щоб розставити столи, стільці, барну стійку і приступити до приготування страв. Як тільки подія розпочнеться, керівник працюватиме з іншими постачальниками, щоб залишатися в курсі розкладу події, прибирати брудний або викинутий посуд, прибирати розбиті келихи та ввічливо задовольняти потреби всіх гостей [3]-[6].

Після заходу – команда кейтерингу найчастіше відповідає за вивезення сміття, організацію прокату для самовивозу і виїзд з місця проведення в установленому порядку [3]-[6].

Існує безліч сумнівних кейтеринг компаній. Корисно вивчити їх онлайн-відгуки, і зазвичай перед бронюванням запитують рекомендації у минулих клієнтів. Крім того, переконайтеся, що компанія громадського харчування, яку замовляють, має страховку та ліцензію на подачу їжі [3]-[6].

Вид чистої, організованої кухні багато говорить про кейтерингову компанію. Якщо все виглядає хаотично і брудно, швидше за все, таким же буде і захід. Погана кейтерингова компанія покаже своє справжнє обличчя в процесі підготовки. Пунктуальність є ключовою якістю хорошої кейтерингової компанії, оскільки часто вони є першим і останнім постачальником, який опиняється на місці [3]-[6]. Оскільки більшість заходів зосереджені на їжі, коктейлях та прибиранні, найм поганої кейтеринг компанії сильно вплине на якість події. Можливо, їм буде важко вчасно приготувати вечерю, можливо, у них млявий персонал, який не справляється з прибиранням за гостями, в результаті чого захід виглядає неохайним. Вони можуть зрізати кути і скоротити штат вашого заходу [3]-[6].

Коли справа доходить до найму кейтерингу, виконання домашніх завдань заощадить багато часу, грошей, а головне, таке святкування увійде в історію як одне з найказковіших заходів, які коли-небудь відвідували гості, а не як ганебне [3]-[6].

## **1.2 Організація та адміністрування застосунку для кейтерингу**

В останні роки харчова промисловість зробила революцію, впровадивши програмне забезпечення (ПЗ) для доставки їжі на вимогу [7]-[9]. Ці програми змінили спосіб, яким люди замовляють їжу, роблячи це простішим та зручнішим. Під час розробки такого застосунку, є кілька речей, які слід враховувати.

По-перше, варто визначити, які функції слід включити у свій застосунок. Деякі ключові функції включають меню із зображеннями та описами, можливість оформити замовлення та здійснити оплату, відстеження замовлень у режимі реального часу та підтримку клієнтів [7]-[9].

Також потрібно враховувати різні типи користувачів, які використовують застосунок, такі як клієнти, власники ресторанів та водії доставки. Кожен тип користувачів матиме різні потреби, тому розробники повинні адаптувати свою програму відповідно до цих потреб [7]-[9].

При розробці ПЗ важливо переконатися, що воно просте у використанні та навігації. Клієнти повинні мати можливість швидко і легко оформити замовлення, а власники ресторанів і водії служб доставки повинні мати можливість управляти своїми замовленнями і доставками без будь-яких ускладнень [7]-[9].

Нарешті, треба ефективно просувати свій застосунок, щоб він досяг вашої цільової аудиторії. Залежно від бюджету та цільової аудиторії, це може включати маркетинг у соціальних мережах, платну рекламу або маркетинг впливу [7]-[9].

Розробка додатків для кейтерингу може бути складним, але корисним заняттям. Дотримуючись цих порад та враховуючи потреби вашої цільової аудиторії, ви можете створити застосунок, який виділяється на конкурентному ринку та забезпечує чудовий досвід користувачів для всіх своїх користувачів [7]-[9].

Мобільний застосунок для кейтерингу – це як модна програма, яку можна завантажити на свій телефон або планшет і яка допоможе замовити їжу у компанії громадського харчування. Він складається з трьох основних частин: тієї, яку користувач бачить на своєму телефоні, тієї, яка стежить за тим, щоб все працювало за лаштунками, і тієї, яка допомагає доставляти їжу [7]-[9].

Коли користувач користується додатком, ви можете заглянути в меню, вибрати те, що вам потрібно, оплатити це і відстежити, коли воно надійде.

Можливо, там навіть є якісь цікаві функції, такі як заохочувальні програми або спеціальні пропозиції [7]-[9].

За лаштунками знаходиться сервер, який забезпечує безперебійну роботу всього обладнання. Він відстежує всі замовлення, стежить за тим, щоб платежі проходили успішно, і оновлює запаси, коли товари закінчуються. Іноді він взаємодіє з іншими службами, такими як банки або картографічні програми, щоб переконатися, що все працює належним чином [7]-[9].

Нарешті, частина доставки програми гарантує, що їжа потрапить до замовника. Це допомагає водіям знайти оптимальний маршрут і гарантує, що замовник буде знати, коли його їжа буде в дорозі [7]-[9].

Усі ці частини використовують різні комп'ютерні мови та інструменти для співпраці. Весь сенс програми в тому, щоб зробити замовлення кейтерингу простим і зручним для всіх учасників [7]-[9].

### **1.3 Порівняння існуючих аналогів**

Кейтерингові вебзастосунки стають все більш популярними в останні роки, і з поважних причин. Одним з головних переваг цих додатків є підвищена ефективність. Автоматизуючи багато аспектів бізнесу громадського харчування, таких як обробка замовлень, відстеження доставки та управління меню, підприємства можуть працювати ефективніше та з меншою кількістю помилок. Це може заощадити час і гроші, а також підвищити загальну якість обслуговування. Крім того, доступ до вебдодатків можливий з будь-якого місця, що робить їх зручним варіантом для клієнтів, які хочуть розміщувати замовлення в дорозі.

Однак є також деякі потенційні недоліки, які слід враховувати. Розробка та підтримка вебпрограми може бути дорогим заняттям, особливо для невеликих підприємств громадського харчування. Також потрібні технічні знання, що може стати проблемою для підприємств, які не мають спеціальної ІТ-команди. Ще однією проблемою є ризики безпеки, оскільки вебпрограми

вразливі до злому та порушення даних. Нарешті, адаптація користувачів може бути непростим завданням, особливо для клієнтів, які звикли робити замовлення по телефону або особисто. Загалом, хоча вебдодатки для громадського харчування пропонують багато переваг, підприємства повинні ретельно зважити потенційні недоліки та визначити, чи є вебзастосунок правильним вибором для їхніх потреб.

### **1.3.1 Служба «Живіт-кейтеринг»**

Компанія є однією з найвідоміших компаній Києва, її постійними клієнтами є «Київський ювелірний завод», L'Oréal, Multiplex Cinema, Samsung і телеканал «Україна». Ті, хто хоч раз скористався Belly Catering, помітять індивідуальний підхід і професіоналізм до кожного клієнта [10], [11].

Компанія пропонує величезний вибір тем, які дозволяють легко вписати їх в концепцію будь-якого заходу [10], [11].

Тут ми переконані, що, купуючи таку складну послугу, як кейтеринг, клієнт повинен бути впевнений у можливостях виконавця. Тому «Живіт-кейтеринг» організовує дегустацію обраного меню та показує, як виглядає сервірування столу та обслуговуючий персонал.

Ще однією особливістю цієї кейтерингової компанії є те, що кухня тут розроблена спеціально під характеристики заміських ресторанів. Тому власники гарантують, що потужності кухні можуть приймати заходи до 2000 гостей [10], [11].

За минулий рік «Живіт-кейтеринг» успішно провів понад 270 заходів і обслужив понад 50 000 гостей. Найпомітніші з цих подій – Ukrainian Software Development Forum 4.0, прем'єра фільму «Легенда про Тарзана» та медична конференція ISAPS [10], [11].

Ефективна презентація. Ми надаємо концепції дизайну. Стиль, який ми витримуємо. Ми створюємо тематичні станції з приголомшливими ефектами. Як подавати устриці на освітлених свічками крижаних скульптурах або різати

шинку навколо живого моху та каміння [10], [11]. Приклад інтерфейсу служби «Живіт-кейтеринг» наведено на рис. 1.1.

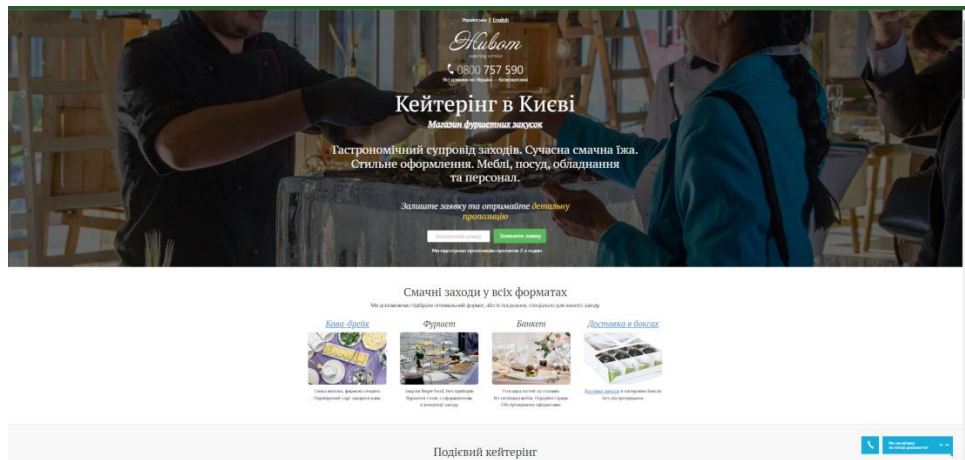


Рисунок 1.1 – Приклад інтерфейсу служби «Живіт-кейтеринг» [10]

Комфортний сервіс. Це коли немає аварій і кожен знає свою зону відповідальності. Кухарі смажать і нарізають овочі, офіціанти обслуговують гостей, а техніки збирають брудний посуд [10], [11].

Смачна їжа. В наявності тільки свіжі, якісні продукти. Контролювати правила зберігання і транспортування. Рекомендуємо правильний вихід на 1 гостя, підбираємо меню, страви та оформлення відповідно до концепції заходу [10], [11].

### 1.3.2 Сервіс «Glovo»

Застосунок Glovo дозволяє замовити будь-який тип продукту: користувач може замовити продукти з супермаркету, аптеки, подарунки та закуски, навіть може замовити пару взуття або одягу, так само можна замовити фаст-фуд, піцу, гамбургери, все, що подобається [12], [13]. У клієнтів є нескінченна кількість продуктів на вибір. Слід зазначити, що Glovo – це не лише застосунок, де можна замовити товар, а й надіслати його, тобто якщо потрібно надіслати терміновий пакет, це можна зробити через Glovo [12], [13].

У Glovo є 2 категорії для розміщення замовлення та розрахунку оплати, а саме: експрес-доставка або після купівлі, якщо клієнт обирає експрес-доставку, застосунок обчислить відстань між пунктом отримання товару та адресою доставки та покаже загальну вартість доставки; Якщо ж клієнт обирає «після купівлі» загальну вартість послуги клієнт зможе дізнатися після того, як кур'єр купить товар [12], [13].

Звісно платформа має свої переваги та недоліки, оскільки ПЗ розвивається та пропонує загалом різні послуги та типи товарів. Glovo – це більш повний сервіс, оскільки він не тільки надає клієнтам можливість доставки продуктів харчування або товарів з супермаркету, але користувач також може швидко замовити їжу, одяг або взуття, а також надає можливість відправляти товари так само, як за допомогою служби доставки посилок, але з більш швидкою доставкою [12], [13].

Зараз Glovo є всесвітньо відомою компанією, але це не скасовує того факту, що він має свої недоліки, серед яких можна виділити:

- іноді програма не надсилає сповіщення про нові замовлення власникам мереж, що призводить до ще більшої затримки замовлення;
- може бути представлено, що програма несправна. і користувачу доведеться перезавантажити телефон, щоб ПЗ знову запрацювало.

Аналогічним чином, часто трапляється, що застосунок має помилки в геолокації GPS, що означає, що платіж не може бути розрахований належним чином. » Інтерфейси Glovo наведено на рис. 1.2.

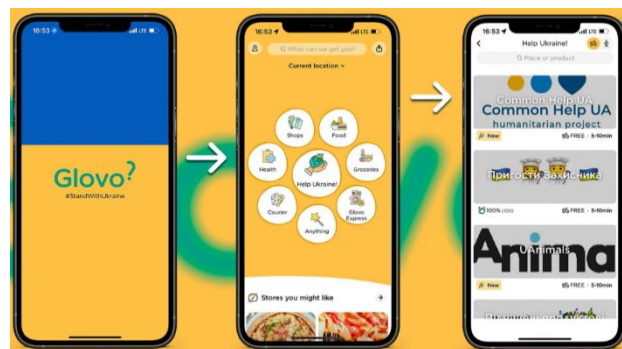


Рисунок 1.2 – Інтерфейси Glovo [12]

### 1.3.3 Сервіс «Bolt Food»

Для того, щоб запустити Bolt Food менш ніж за 6 місяців, розробникам довелося працювати над бекендом і фронтом одночасно [14], [15]. Розробникам необхідно було зрозуміти основний потік і побачити кінцевий користувальницький інтерфейс, щоб правильно побудувати інфраструктуру, створити фундамент і паралельно працювати над інтерфейсом [14], [15].

По суті, це призвело до того, що на розробку кожного з продуктів було відведено всього 2 тижні.

Те, чим ми повинні були бути спроектовані [14], [15]:

- споживчий застосунок;
- застосунок для кур'єра;
- застосунок для ресторану;
- система управління меню.

Розробка продукту, передача якого мала відбутися через 2 тижні, призвела до дуже жорстких компромісів. Було вирішено зосередитись на конкурентному аналізі як на основному дослідницькому методі. Завдяки набору Bolt UI kit, який вже існував, розробники могли негайно поекспериментувати з високоякісним користувальницьким інтерфейсом і не витрачати час на рамки [14], [15]. Було очевидно, що, враховуючи терміни, у них не було часу на належні дослідження та тестування користувачами. Вони планували створити Model-View-Presenter, а потім вивчати, розуміти, збирати зворотний зв'язок [14], [15].

Заглядаючи вперед, це було хорошим рішенням, оскільки в період тестування не було жодних серйозних блокувань, а також з часом ми дізналися, що працює та багато іншого [14], [15].

На сьогоднішній день Bolt Food працює в 19 країнах, більш ніж в 40 містах, з більш ніж 12000 ресторанами [14], [15]. Він став основним продуктом в екосистемі Bolt.

Озираючись на прийняте розробниками рішення, здається, що вони вчинили правильно. Копіювання конкурсу було єдиним розумним варіантом, враховуючи терміни. Але навіть в них було більше часу, вони проводили дослідження і витрачали гроші і час на тестування користувачів, експерименти з різними моделями і дизайнами ніколи не замінили б реальні дані, які вони отримали після запуску сервісу. Створення чогось дешевого і швидкого відкрило можливість для вивчення бізнесу і поступилося місцем створенню інноваційних та унікальних рішень [14], [15].

Наприклад, в режимі активного замовлення, після відбору, вони постаралися приховати багато деталей і дати користувачеві тільки відчуття прогресу. Але з часом вони вирішили спробувати інший підхід-зосередитися на прозорості та надати користувачам повну картину [14], [15]. Вони отримали величезну кількість позитивних відгуків, і тепер їхня мета – показати користувачеві ще більше деталей, використовуючи поточне дизайнерське рішення. Приклади інтерфейсів Bolt Food наведено на рис. 1.3.

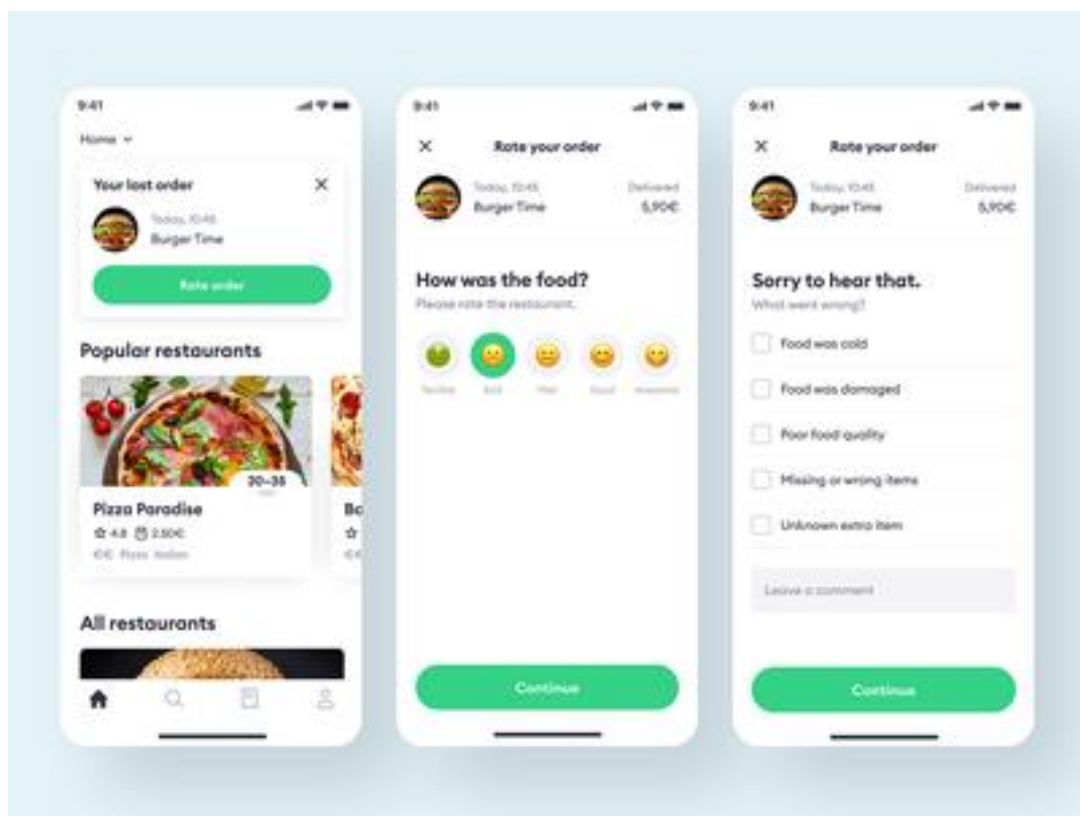


Рисунок 1.3 – Приклади інтерфейсів Bolt Food [14]

### 1.3.4 Порівняння існуючих аналогів

Проведемо порівняння за певними критеріями, хід якого представимо у вигляді таблиці (табл. 1.1). Серед критеріїв виділимо наступні:

- мапа надання послуг – в якій частині України сервіс надає свої послуги;
- обслуговування великих замовлень – можливість обслуговування великих заходів, з великою кількістю гостей;
- власне приготування – сервіс самостійно займається приготуванням їжі доставляє її з ресторанів та кафе;
- послуги з організації оточення – чи надає сервіс послуги із підготовки, облаштування, підтримки та прибирання середовища, де відбувається захід.

Таблиця 1.1 – Програмне забезпечення для підтримки освітньої діяльності

| Критерії порівняння                       | Живіт-кейтеринг  | Glovo          | Bolt Food      | NewCate        |
|-------------------------------------------|------------------|----------------|----------------|----------------|
| Мапа надання послуг                       | Київська область | По всій країні | По всій країні | По всій країні |
| Обслуговування великих замовлень          | Можливо          | Не можливо     | Не можливо     | Можливо        |
| Власне приготування                       | Так              | Ні             | Ні             | Так            |
| Послуги організації оточення <sup>3</sup> | Так              | Ні             | Ні             | Так            |

З результатів порівняння можна зробити висновки, що більшості випадків такі застосунки або достатньо обмежені за територіальною областю, чи то за форматом роботи. Тобто саме кейтерингові служби в звичайному розумінні зосереджують свою роботу виключно на декількох містах, що знаходяться поруч. Звичайні ж служби доставки співпрацюють з ресторанами конкретного міста, але об'єм доставки обмежено кур'єрськими ресурсами

служби. Також оскільки така служба є виключно доставкою, то відповідальність за якість їжі такі служби не несуть. І в разі чого, слід буде звертатися саме до ресторану, що готував замовлення.

Тож актуальною задачею залишається завдання програмної реалізації застосунку для організації роботи кейтерингової служби.

#### **1.4 Постановка завдання**

З метою створення системи здорового харчування співробітників компанія замовляє комплексні обіди підприємству громадського харчування. Це замовлення залежить від кількості співробітників, їх смаків та звичок харчування.

Підприємство громадського харчування пропонує комплексний обід згідно меню, що оновлюється кожного дня, в кількості 4-х страв та напою. Робітники компанії мають змогу зробити замовлення напередодні на один день чи на весь тиждень згідно меню весь комплексний обід або бажані страви окремо.

##### **1.4.1 Межі проєкту**

Система буде складатися з декількох модулів. Деякі з них будуть в готовому вигляді підключені до системи: модуль платіжної системи, модуль роботи з базою даних, а всі інші будуть розроблені групою програмістів системи.

##### **1.4.2 Функціональність системи**

Система повинна:

- аутентифікувати користувача за допомогою логіна та пароля;
- надавати доступ користувачам до персональної сторінки, меню;

- зберігати відомості про всіх користувачів;
- надавати можливість робити замовлення комплексного обіду;
- надавати можливість оновлювати меню;
- надавати можливість обробляти замовлення користувачів;
- надавати можливість сплатити за обіди через електронну платіжну систему;
- надавати можливість виводити звіти замовлень по кожному працівнику та по всій компанії загалом;
- надавати нескладний та лаконічний інтерфейс;
- забезпечувати швидку та стабільну роботу без збоїв.

#### **1.4.3 Вимоги до інтерфейсу**

Інтерфейс системи повинен бути зручним, інтуїтивно зрозумілим та дружнім. Всі страви в меню повинні бути детально описані та візуально підкріплені відповідними зображеннями.

В системі повинно бути передбачено окремі вкладки для різних типів користувачів.

#### **1.4.4 Вимоги до продуктивності**

Система, що розроблюється, повинна бути швидкою у використанні, витрати на обробку замовлення повинні складати декілька секунд.

Система розрахована на користування співробітниками малих та середніх за розміром підприємств, тому вона повинна обробляти замовлення зі швидкістю 2 на секунду та забезпечувати доступ до 200 користувачам на день.

### **1.4.5 Вимоги до безпеки**

В системі буде передбачено проведення фінансових операцій, доступ до яких має бути обмежений. Тому ці фінансові відомості повинні бути захищені спеціальним чином.

### **1.4.6 Експлуатаційні вимоги**

Для користування застосунком, користувач повинен мати ПК з доступом до мережі Інтернет, обладнаний веббраузером. Для створення власного сховища користувач має зареєструватися [13]-[15].

### **1.4.7 Не функціональні вимоги до продукту**

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини мобільного застосунку:

- сервер із розгорнутою серверною операційною системою (ОС);
- підтримка інтерпретатора PHP5;
- MySQL-сервер з розгорнутою базою даних (БД) та налаштованою

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК з IBM-сумісною архітектурою;
- процесор із тактовою частотою від 1,7 GHz (для безперебійної роботи нових версій браузерів);
- обсяг оперативної пам'яті – від 3GB (для безперебійної роботи нових версій браузерів);
- підключення до мережі Internet;
- встановлений браузер.

### **1.5 Висновки за розділом 1**

В першому розділі було проаналізовано проблемну область та досліджено служби кейтерингу та їх місце в сучасному світі. Було досліджено існуючі аналоги та виявлено їх переваги і недоліки. Спираючись на це вдалося сформулювати набір функціональних вимог та вирішити основні проєктні питання.

## 2 МАТЕРІАЛИ І МЕТОДИ

В попередньому розділі було виконано аналіз проблемної області служб кейтерингу та їх роботи. Було досліджено існуючі аналоги та виявлено їх переваги і недоліки. Спираючись на це вдалося сформулювати набір функціональних вимог та вирішити основні проєктні питання.

В другому розділі буде розглянуто низку технічних питань розробки: вибір технології та засобів розробки, принципи реалізації. В цьому розділі буде детально проаналізовано та обрано засоби розробки, вому числі: мову програмування, програмні комплекси та засоби для оформлення ПЗ.

### 2.1 Вибір мови програмування та інших технологій розробки

PHP (гіпертекстовий препроцесор) – це серверна мова сценаріїв, яка широко використовується для веброзробки. Це дуже універсальна мова, яку легко вивчити, має велику спільноту розробників, сумісна з широким спектром платформ та економічна. Незважаючи на появу нових мов програмування та фреймворків, PHP все ще залишається популярним вибором для веброзробки. У цьому есе ми розглянемо деякі причини, чому PHP все ще є гарною базовою ідеєю для розробки вебдодатків [16]-[18].

По-перше, PHP простий у вивченні та має простий синтаксис. Це робить його ідеальним вибором для новачків, які тільки починають займатися веб-розробкою. PHP-код може бути вбудований безпосередньо в HTML, що полегшує створення динамічних вебсторінок, які можуть взаємодіяти з користувачем. Синтаксис PHP подібний до інших мов програмування, таких як C та Java, що полегшує швидке вивчення PHP розробниками, знайомими з цими мовами [16]-[18].

По-друге, PHP має велику та активну спільноту розробників, які сприяють його розробці та надають підтримку користувачам. Ця спільнота розробила величезну кількість бібліотек, фреймворків та інструментів, які

прискорюють та спрощують розробку. PHP має величезну кількість доступних бібліотек з відкритим кодом, таких як популярна бібліотека PHPMailer для надсилання електронних листів та популярні фреймворки Symfony та Laravel для створення вебдодатків [16]-[18].

Symfony framework – це популярний PHP-фреймворк, який використовується для створення масштабованих і високопродуктивних вебдодатків. Він надає набір повторно використовуваних компонентів, які розробники можуть використовувати для швидкого та ефективного створення вебдодатків. Фреймворк Laravel – це ще один популярний фреймворк PHP, який розроблений таким чином, щоб бути простим у використанні та дуже виразним. Він має простий та елегантний синтаксис, який дозволяє розробникам легко писати чистий та підтримуваний код [16]-[18].

По-третє, PHP сумісний з широким спектром операційних систем, вебсерверів і баз даних. Це полегшує розгортання додатків на різних платформах. PHP може працювати практично на будь-якому веб-сервері, включаючи Apache, Nginx та Microsoft IIS. Він також сумісний з широким спектром баз даних, включаючи MySQL, PostgreSQL та Oracle. Така сумісність дозволяє розробникам легко створювати вебдодатки, які можуть працювати на широкому спектрі платформ і взаємодіяти з різними типами джерел даних [16]-[18].

По-четверте, PHP добре працює і є масштабованим. PHP – це серверна мова, що означає, що вона виконується на сервері та надсилає вихідні дані клієнту. Це призводить до швидшого завантаження сторінки та підвищення продуктивності. PHP також має високу масштабованість, що означає, що він може обробляти великі обсяги трафіку без уповільнення роботи. Це робить його ідеальним вибором для створення високопродуктивних вебдодатків, здатних обслуговувати велику кількість користувачів [16]-[18].

Нарешті, PHP є економічно ефективним. PHP-це мова з відкритим кодом, що означає, що вона вільна у використанні та розповсюдженні. Це робить його привабливим варіантом для малого бізнесу та стартапів, які хочуть

зменшити витрати на розробку. РНР можна використовувати для створення складних вебдодатків без високих ліцензійних зборів, пов'язаних з іншими мовами програмування та фреймворками [16]-[18].

На закінчення, РНР все ще є гарною базовою ідеєю для розробки вебдодатків. Його простота, універсальність, велика спільнота, сумісність з різними платформами та базами даних, масштабованість та економічна ефективність роблять його привабливим варіантом для веброзробників. Незважаючи на те, що існують інші доступні мови програмування та фреймворки, РНР залишається популярним вибором для веброзробки завдяки своїй простоті використання, продуктивності та сумісності з широким спектром платформ [16]-[18].

Проведемо порівняння мов програмування для реалізації веб-застосунків, результати порівняння представимо в табличній формі (табл. 2.1). У якості критеріїв будемо використовувати наступні:

- простота входу в розробку – оцінка складності входу розробника в процес розробки;
- навантаженість синтаксису – складність та специфічність синтаксису мови програмування, що може ускладнити розробку;
- підтримка спільноти – активність підтримки спільноти, через форуми, конференції та спілкування;
- складність підтримки, розвитку та масштабованості – оцінка складності подальшої підтримки та розвитку застосунку, та оцінка можливості подальшого масштабування застосунку;
- комплекси для спрощення розробки – спеціалізовані програмні надбудови та комплекси для спрощення процесу розробки;

З таблиці видно, що вже довгий час РНР залишається одним з лідерів веброзробки, що пояснюється значним досвідом спільноти та розвитком множини програмних комплексів для спрощення розробки.

Таблиця 2.1 – Порівняння мов програмування

| Критерій                                             | PHP                  | JavaScript           | C# + .Net                 |
|------------------------------------------------------|----------------------|----------------------|---------------------------|
| Простота входу в розробку                            | Проста               | Проста               | Ускладнена                |
| Навантаженість синтаксису                            | Низька               | Низька               | Середня                   |
| Підтримка спільноти                                  | Потужна<br>спільнота | Потужна<br>спільнота | Спільнота<br>розвивається |
| Складність підтримки, розвитку<br>та масштабованості | Низька               | Низька               | Середня                   |
| Комплекси для спрощення<br>розробки                  | Багато               | Середньо             | Мало                      |

## 2.2 Вибір та обґрунтування фреймворку

Symfony – це популярний PHP-фреймворк, який широко використовується для розробки вебдодатків [19]-[22]. Вперше він був випущений в 2005 році і з тих пір став одним з найбільш широко використовуваних фреймворків PHP. Symfony – це фреймворк з відкритим кодом, заснований на архітектурному шаблоні Model-View-Controller (MVC). У цьому есе ми розглянемо деякі причини, чому Symfony фреймворк є гарною ідеєю для розробки вебдодатків на PHP [19]-[22].

Однією з ключових переваг Symfony є його модульна архітектура. Symfony розроблений з високим ступенем модульності, що означає, що розробники можуть використовувати лише ті компоненти, які вони потребують для свого конкретного застосунка. Це робить Symfony дуже гнучкою платформою, яка може бути адаптована до конкретних потреб кожного проєкту. Symfony надає широкий спектр повторно використовуваних компонентів, які можна використовувати для швидкого та ефективного створення вебдодатків. Ці компоненти включають маршрутизацію, кешування, створення шаблонів, автентифікацію та багато інших [19]-[22].

Symfony також надає ряд інструментів, які полегшують управління залежностями та упаковку залежностей у повторно використовувані

компоненти. Компонент `Symfony Dependency Injection` є одним з таких інструментів, який надає гнучку і просту у використанні платформу для управління залежностями. Компонент `Dependency Injection` дозволяє розробникам вказувати залежності, необхідні їхній програмі, а потім автоматично пов'язувати ці залежності під час виконання. Це дозволяє легко керувати складними залежностями та створювати повторно використовувані компоненти, які можна використовувати в різних проєктах [19]-[22].

Ще однією перевагою `Symfony` є його акцент на тестуванні. `Symfony` надає широкий спектр інструментів тестування, включаючи модульне тестування, функціональне тестування та інтеграційне тестування. Ці інструменти дозволяють розробникам легко протестувати свій код і переконатися, що він працює належним чином. `Symfony` також надає потужну панель інструментів налагодження, яку можна використовувати для налагодження коду безпосередньо в браузері. Це дозволяє легко виявляти та швидко виправляти помилки, що може заощадити багато часу та зусиль у процесі розробки [19]-[22].

`Symfony` також надає широкий спектр функцій для створення `RESTful API`. `Symfony` надає потужну систему маршрутизації, яка дозволяє легко зіставляти шаблони `URL`-адрес з діями контролера. Це полегшує створення `API`, які можуть обробляти запити з різних джерел та відповідати відповідними даними. `Symfony` також надає ряд інструментів серіалізації, які дозволяють легко конвертувати об'єкти в різні формати, такі як `JSON` або `XML`. Це спрощує створення `API`, які можуть використовуватися різними типами клієнтів, такими як веббраузери, мобільні додатки або інші вебслужби [19]-[22].

`Symfony` також приділяє велику увагу безпеці. Платформа надає ряд функцій, пов'язаних із безпекою, включаючи автентифікацію, авторизацію та хешування паролів. `Symfony` також пропонує потужну систему брандмауера, яку можна використовувати для обмеження доступу до різних частин програми. Це полегшує створення безпечних вебдодатків, які захищають конфіденційні дані [19]-[22].

Нарешті, Symfony має велику та активну спільноту розробників. Спільнота Symfony постійно працює над вдосконаленням фреймворку та наданням нових функцій та компонентів. Ця спільнота надає багато ресурсів, включаючи документацію, навчальні посібники та приклади коду. Існує також безліч сторонніх бібліотек та плагінів, які можна використовувати для розширення функціональності Symfony [19]-[22].

На закінчення, Symfony-це потужний і гнучкий PHP-фреймворк, який добре підходить для розробки вебдодатків. Його модульна архітектура, акцент на тестуванні, підтримка RESTful API, акцент на безпеці та велика та активна спільнота роблять його привабливим вибором для веброзробників. Symfony надає ряд повторно використовуваних компонентів, які можна використовувати для швидкого та ефективного створення вебдодатків. Його акцент на тестуванні та налагодженні дозволяє легко переконатися, що код працює належним чином. Його підтримка RESTful API спрощує створення вебсервісів, які можуть використовуватися різними типами клієнтів. Його орієнтація на безпеку дозволяє легко створювати вебдодатки, які захищають конфіденційні дані. Загалом, Symfony-це потужний і гнучкий PHP-фреймворк, який добре підходить для створення складних вебдодатків [19]-[22].

Порівняльна характеристика найпоширеніших CMS для розробки вебзастосунку у таблиці 2.2.

Серед критеріїв порівняння будемо розглядати наступні:

- простота початку розробки – простота входу та початку розробки застосунку із використанням обраного фреймворку;
- генерація моделей – наявність вбудованих механізмів для автоматизації генерації моделей;
- генерація контролерів – наявність вбудованих механізмів для автоматизації генерації контролерів;
- збирачі пакетів – співпраця фреймворку із збирачами пакетів для простоти встановлення та використання сторонніх модулів;

- взаємодія із СКБД – простота та захищеність роботи фреймворку із СКБД;
- спільнота та підтримка – наявність та активність підтримки спільноти.

Таблиця 2.2 – Порівняння фреймворків

| Порівняння фреймворків    | Symfony                            | Laravel!                                                             |
|---------------------------|------------------------------------|----------------------------------------------------------------------|
| Простота початку розробки | Середня                            | Середня                                                              |
| Генерація моделей         | Через вбудовані механізми          | Через вбудовані механізми                                            |
| Генерація контролерів     | Через вбудовані механізми          | Через вбудовані механізми (насправді, не прив'язаний жорстко до MVC) |
| Збирачі пакетів           | Співпрацює                         | Співпрацює                                                           |
| Взаємодія із СКБД         | Проста, багато захисних механізмів | Проста                                                               |
| Спільнота та підтримка    | Широка                             | Широка                                                               |

З таблиці можна зробити висновок, що Symfony залишається лідером, через поважну історію та активність професійної спільноти з розробки та розвитку фреймворку.

### 2.3 Структурна схема розробки

На рисунку 2.1 наведено структурну схему ПЗ, що розробляється.

Система складається з двох частин:

- клієнтська;
- серверна.

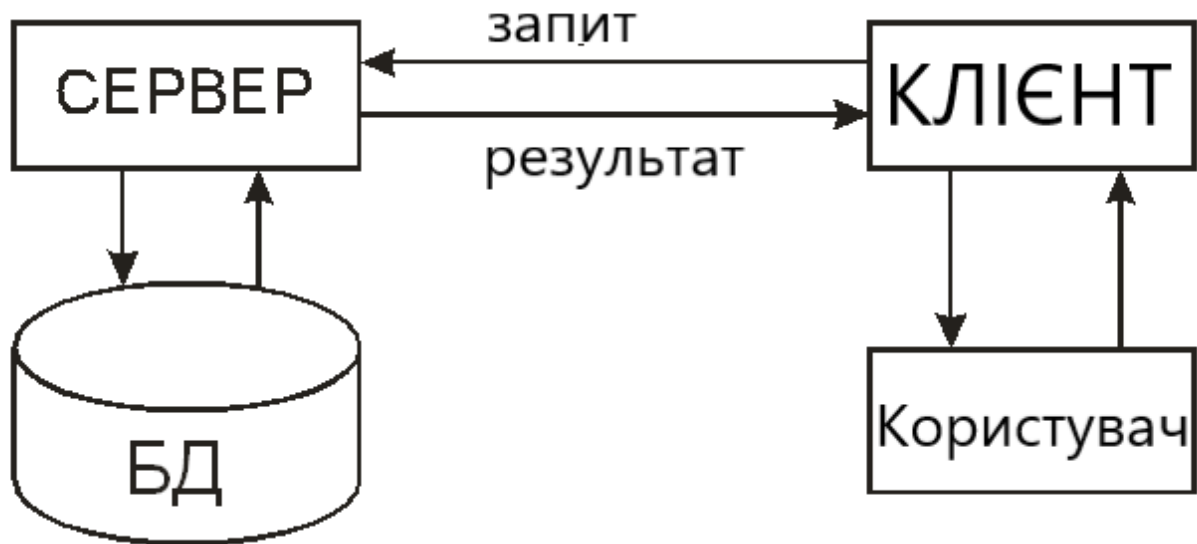


Рисунок 2.1 – Загальна схема системи за обраною темою

В якості клієнта виступають користувачі різних категорій, в якості сервера – СКБД.

## 2.4 Висновки за розділом 2

Під час роботи над другим розділом було виконано дослідження та аналіз засобів розробки, серед яких: мова програмування, фреймворк та інші. Так, після огляду та вибору мов програмування для розробки було обрано PHP, для пришвидчення розробки та впорядкування бізнес-логіки застосунку було обрано фреймворк Symfony.

## **3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ**

### **3.1 Проєктування моделей системи**

Уніфікована мова моделювання (UML) – це стандартизована графічна мова, яка використовується для моделювання програмних систем. UML використовується розробниками програмного забезпечення для створення візуальних уявлень програмних систем та їх компонентів. UML важливий для проєктування розробки додатків, оскільки він забезпечує стандартизований спосіб моделювання та документування програмних систем. Це дозволяє розробникам ефективніше взаємодіяти між собою та зацікавленими сторонами протягом усього процесу розробки [23], [24].

UML надає набір графічних позначень для представлення програмних систем, включаючи класи, об'єкти, інтерфейси та компоненти. Ці позначення можуть бути використані для створення візуальних діаграм, що представляють структуру та поведінку програмних систем. Діаграми UML можна використовувати для моделювання різних аспектів програмних систем, включаючи випадки використання, класи, послідовності, дії та розгортання [23], [24].

Однією з ключових переваг UML є те, що він дозволяє розробникам створювати візуальне представлення програмної системи до її створення. Це може допомогти виявити потенційні проблеми та недоліки дизайну на ранніх стадіях процесу розробки, коли вносити зміни простіше та дешевше. Діаграми UML також можуть використовуватися для інформування зацікавлених сторін, таких як клієнти або керівники проєктів, про дизайн програмної системи. Це може допомогти гарантувати, що всі сторони, які беруть участь у процесі розробки, мають чітке уявлення про створювану систему [23], [24].

UML також може допомогти покращити якість програмних систем, надаючи стандартизований спосіб документування архітектури та дизайну системи. Діаграми UML можна використовувати для створення детальної схеми

системи, включаючи її структуру, поведінку та взаємодію з іншими системами. Це може допомогти гарантувати, що програмна система розроблена та впроваджена таким чином, щоб відповідати вимогам та потребам користувачів [23], [24].

UML також важливий для розробки програмного забезпечення, оскільки він сприяє стандартизованому підходу до моделювання та документування. UML є широко визнаним стандартом моделювання програмного забезпечення, що означає, що розробники можуть ефективніше взаємодіяти між собою та зацікавленими сторонами під час використання UML. UML також просуває стандартизований підхід до документації, який може допомогти забезпечити, щоб програмні системи були добре задокументовані та підтримувалися з часом [23], [24].

Нарешті, UML важливий для розробки програмного забезпечення, оскільки він підтримує шаблони дизайну програмного забезпечення. Шаблони проєктування програмного забезпечення-це перевірені рішення поширених проблем проєктування програмного забезпечення. UML надає набір графічних позначень, які можна використовувати для представлення шаблонів дизайну програмного забезпечення, полегшуючи розробникам застосування цих шаблонів до своїх програмних систем [23], [24].

На закінчення, UML є важливим інструментом для проєктування розробки додатків, оскільки він забезпечує стандартизований спосіб моделювання та документування програмних систем. UML дозволяє розробникам створювати візуальні уявлення програмних систем до їх побудови, що може допомогти виявити потенційні проблеми та недоліки дизайну на ранніх стадіях процесу розробки. UML просуває стандартизований підхід до моделювання та документування, який може допомогти покращити якість програмних систем та підтримувати шаблони дизайну програмного забезпечення [23], [24].

Діаграма класів UML – це тип діаграми UML, яка показує статичну структуру системи шляхом моделювання її класів, атрибутів, методів та

взаємозв'язків. Діаграми класів є широко використовуваним типом діаграм UML і можуть використовуватися для моделювання широкого спектру програмних систем, від невеликих додатків до великих корпоративних систем. Ось кілька кроків щодо використання діаграм класів UML [23], [24].

**Визначення класів:** першим кроком у створенні діаграми класів UML є визначення класів, які є частиною системи, яку ви моделюєте. Класи представляють об'єкти або поняття в системі, і кожен клас повинен мати ім'я, яке точно відображає його призначення. Наприклад, якщо ви моделюєте банківську систему, у вас можуть бути такі класи, як рахунок, клієнт, транзакція та банк [23], [24].

**Визначення атрибутів:** після того, як ви визначили класи, наступним кроком є визначення атрибутів, що належать кожному класу. Атрибути-це властивості класу, і вони можуть бути як загальнодоступними, так і приватними. Загальнодоступні атрибути доступні для ІНШИХ КЛАСІВ, тоді як до приватних атрибутів можна отримати доступ лише всередині класу. Наприклад, клас рахунку може мати такі атрибути, як номер рахунку, баланс та тип рахунку [23], [24].

**Визначення методів:** методи – це операції, які можна виконати над класом. Методи використовуються для зміни стану класу (тобто значень його атрибутів) або для повернення інформації про клас. Наприклад, клас Account може містити такі методи, як deposit, withdraw та getBalance [23], [24].

**Додавання відносин:** відносини між класами використовуються, щоб показати, як вони пов'язані один з одним. В UML існує кілька типів відносин, включаючи асоціацію, агрегацію та успадкування. Асоціація - це зв'язок між двома класами, агрегація - це зв'язок, при якому один клас містить інший клас, А успадкування-зв'язок, при якому один клас успадковує атрибути та методи від іншого класу [23], [24].

**Визначення множинностей:** множинність визначає кількість екземплярів одного класу, які можуть бути пов'язані з екземплярами іншого

класу. Наприклад, у банківській системі Клієнт може мати кілька облікових записів, але обліковий запис може належати лише одному клієнту [23], [24].

Додавання анотацій: анотації можуть бути додані до діаграми класів UML для надання додаткової інформації або роз'яснень. Наприклад, анотації можуть бути використані для пояснення призначення класу, атрибута чи методу або для надання приміток до самої діаграми [23], [24].

Доопрацювання діаграми: після створення оригінальної діаграми класів UML може знадобитися її доопрацювання, щоб зробити її більш повною або точною. Це може включати додавання або видалення класів, атрибутів, методів або зв'язків, а також налаштування множинності або анотацій [23], [24].

Діаграми класів UML можна створювати за допомогою різних програмних засобів, включаючи програмне забезпечення для моделювання UML, текстові редактори або навіть інструменти малювання, такі як Microsoft Visio або Lucidchart. Важливо створити чітке та достовірне представлення модельованої системи, щоб її можна було легко зрозуміти та використати розробниками, зацікавленими сторонами та іншими членами команди розвитку [23], [24].

Діаграма класів для застосунку, що розробляється наведено на рис. 3.1.

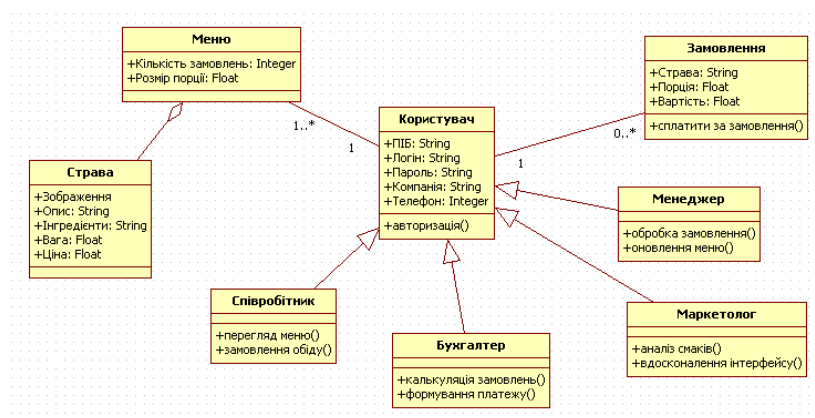


Рисунок 3.1 – Діаграма класів

Діаграма пакетів UML – це тип діаграми UML, яка показує організацію та залежності пакетів у системі. Пакети використовуються для організації

пов'язаних елементів (таких як класи, інтерфейси та інші пакети) в єдину групу, а діаграми пакетів використовуються, щоб показати, як ці пакети організовані та пов'язані між собою. Ось кілька кроків щодо використання діаграми пакетів UML [23], [24].

Ідентифікація пакетів: першим кроком у створенні діаграми пакетів UML є ідентифікація пакетів, які є частиною системи, яку ви моделюєте. Пакети повинні мати назви, що відображають їх зміст і призначення, і вони повинні бути організовані таким чином, щоб це мало сенс для модельованої системи [23], [24].

Додавання пакетів на діаграму: після того, як ви визначили пакети, додайте їх на діаграму пакетів, використовуючи прямокутні форми. Назва кожної упаковки повинна бути написана у верхньому лівому куті прямокутника, а зміст упаковки повинен бути вказаний всередині прямокутника [23], [24].

Додавання залежностей між пакетами: залежності між пакетами Показані стрілками. Залежність між двома пакетами вказує на те, що один пакет якимось чином залежить від іншого. Наприклад, пакет, що містить класи, які використовують класи з іншого пакета, матиме залежність від цього іншого пакета [23], [24].

Додавання узагальнень між пакетами: Узагальнення між пакетами Показані суцільною лінією з порожнистою стрілкою, що вказує від дочірнього пакета до батьківського. Узагальнення вказує на те, що дочірній пакет успадковує зміст батьківського пакета. Це схоже на успадкування між класами, але стосується і пакетів [23], [24].

Додати асоціації між пакетами: Зв'язки між пакетами Показані пунктирною лінією. Асоціація вказує на те, що два пакети якимось чином пов'язані, але між ними немає відносин успадкування або залежності. Наприклад, два пакети можуть мати асоціацію, якщо вони містять пов'язані класи або інтерфейси [23], [24].

Додавання обмежень до пакетів: обмеження можуть бути додані до пакетів, щоб вказати вимоги або обмеження до вмісту пакета. Обмеження записуються як текст і укладаються у фігурні дужки поруч із назвою пакета [23], [24].

Доопрацювання діаграми: після створення оригінальної діаграми пакета UML може знадобитися доопрацювати її, щоб зробити її більш повною або точною. Це може включати додавання або видалення пакетів, залежностей, узагальнень, асоціацій або обмежень, а також коригування макета діаграми, щоб зробити її більш читабельною [23], [24].

Діаграми пакетів UML можна створювати за допомогою різних програмних засобів, включаючи програмне забезпечення для моделювання UML, текстові редактори або навіть інструменти малювання, такі як Microsoft Visio або Lucidchart. Важливо створити чітке та достовірне уявлення про організацію та залежності пакетів у модельованій системі, щоб її можна було легко зрозуміти та використати розробниками, зацікавленими сторонами та іншими членами команди розробників.

Діаграма пакетів для застосунку, що розробляється наведено на рис. 3.2.

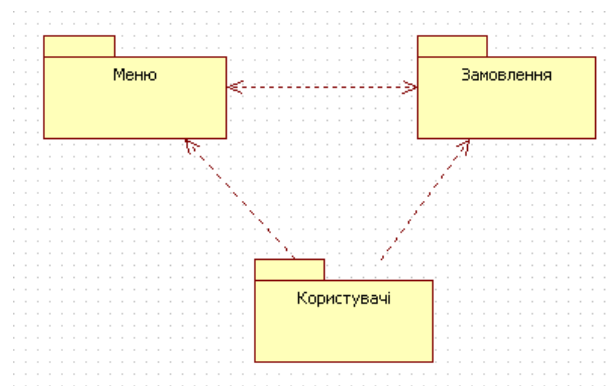


Рисунок 3.2 – Діаграма пакетів

Діаграма компонентів UML – це тип діаграми UML, яка показує залежності та взаємозв'язки між програмними компонентами в системі. Компоненти використовуються для представлення модулів, підсистем та інших функціональних одиниць системи, а діаграми компонентів використовуються

для демонстрації того, як ці компоненти взаємодіють між собою. Ось кілька кроків щодо використання діаграми компонентів UML [23], [24].

Визначте компоненти: першим кроком у створенні діаграми компонентів UML є визначення компонентів, які є частиною системи, яку ви моделюєте. Компоненти повинні бути названі таким чином, щоб вони відображали їх функціональність і призначення, і вони повинні бути організовані таким чином, щоб це мало сенс для модельованої системи [23], [24].

Додавання компонентів до діаграми: після того, як ви визначили компоненти, додайте їх до діаграми компонентів, використовуючи прямокутні форми. Назва кожного компонента повинна бути написана у верхньому лівому куті прямокутника, а інтерфейси компонента повинні бути перераховані всередині прямокутника [23], [24].

Додавання залежностей між компонентами: залежності між компонентами Показані стрілками. Залежність між двома компонентами вказує на те, що один компонент певним чином залежить від іншого. Наприклад, компонент, що надає послугу, може залежати від іншого компонента, що надає дані [23], [24].

Додайте необхідні та надані інтерфейси: необхідні інтерфейси представляють послуги, які компонент вимагає від інших компонентів, тоді як надані інтерфейси представляють послуги, які компонент надає іншим компонентам. Необхідні інтерфейси відображаються як півкола на лівій стороні прямокутника компонента, тоді як надані інтерфейси відображаються як півкола на правій стороні прямокутника компонента [23], [24].

Додавання з'єднувачів між компонентами: З'єднувачі використовуються для представлення взаємодій між компонентами, таких як виклики методів або передача повідомлень. З'єднувачі відображаються у вигляді ліній між інтерфейсами компонентів і можуть бути позначені іменами переданих методів або повідомлень [23], [24].

Додавання портів до компонентів: порти використовуються для забезпечення точки з'єднання між компонентом та зовнішнім світом, таким як інтерфейс користувача або мережа. Порти показані у вигляді маленьких квадратів з боків прямокутника компонента і можуть бути підключені до інших компонентів або до зовнішніх систем.

Додавання обмежень до компонентів: обмеження можуть бути додані до компонентів, щоб вказати вимоги або обмеження до функціональності компонента. Обмеження записуються у вигляді тексту і укладаються в фігурні дужки поруч з ім'ям компонента [23], [24].

Доопрацювання діаграми: після створення оригінальної діаграми компонентів UML може знадобитися її доопрацювання, щоб зробити її більш повною або точною. Це може включати додавання або видалення компонентів, залежностей, інтерфейсів, з'єднувачів, портів або обмежень, а також коригування макета діаграми, щоб зробити її більш читабельною [23], [24].

Діаграми компонентів UML можна створювати за допомогою різних програмних засобів, включаючи програмне забезпечення для моделювання UML, текстові редактори або навіть інструменти малювання, такі як Microsoft Visio або Lucidchart. Важливо створити чітке та достовірне представлення взаємодій та залежностей між компонентами модельованої системи, щоб її можна було легко зрозуміти та використати розробниками, зацікавленими сторонами та іншими членами команди розвитку [23], [24].

Діаграма компонентів наведена на рис. 3.3.

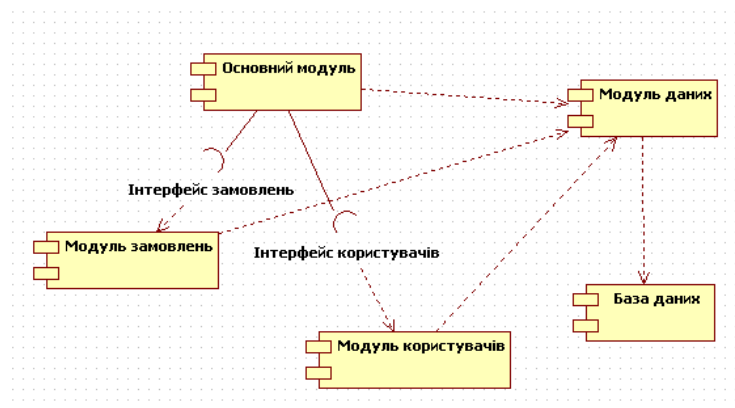


Рисунок 3.3 – Діаграма компонентів

Діаграма прецедентів UML – це графічне зображення функціональності системи та учасників, які взаємодіють з нею. Діаграми випадків використання використовуються для моделювання системи з точки зору користувача, показуючи різні способи взаємодії користувачів із системою для досягнення своїх цілей. Ось кілька кроків щодо використання діаграми випадків використання UML [23], [24].

Визначення учасників: першим кроком у створенні діаграми випадків використання є визначення учасників, які будуть взаємодіяти з системою. Дійовими особами можуть бути користувачі-люди, інші системи або зовнішні пристрої. Дійові особи представлені на схемі фігурками з паличок [23], [24].

Визначення випадків використання: після того, як ви визначили акторів, визначте випадки використання. Варіант використання представляє конкретну мету або завдання, яке користувач хоче досягти за допомогою системи. Варіанти використання представлені овалами на діаграмі [23], [24].

Підключення учасників до випадків використання: після того, як ви визначили учасників та випадки використання, з'єднайте їх лініями, щоб показати, як учасники взаємодіють із системою через випадки використання. Кожен рядок представляє зв'язок між суб'єктом і варіантом використання і позначена ім'ям випадку використання [23], [24].

Доопрацювання діаграми: після створення початкової схеми випадків використання може знадобитися її доопрацювання, щоб зробити більш повною або точною. Це може включати додавання або видалення акторів або випадків використання, а також коригування макета діаграми, щоб зробити її більш читабельною [23], [24].

Додавання додаткових елементів: діаграми випадків використання також можуть включати додаткові елементи, такі як відносини включення, розширення відносин та узагальнення відносин. Відносини включення представляють той факт, що один варіант використання включає функціональність іншого випадку використання, тоді як відносини розширення представляють необов'язкову функціональність, яка може бути додана до

випадку використання. Відносини Узагальнення представляють той факт, що один варіант використання є більш загальною формою іншого випадку використання [23], [24].

Створення опису для кожного випадку використання: для кожного випадку використання на діаграмі створіть опис, який описує конкретні кроки, необхідні для досягнення мети чи завдання. Це може бути використано в якості орієнтира для розробників, які реалізують варіанти використання [23], [24].

Діаграми прецедентів UML можна створювати за допомогою різних програмних засобів, включаючи програмне забезпечення для моделювання UML, текстові редактори або навіть інструменти малювання, такі як Microsoft Visio або Lucidchart. Важливо створити чітке і достовірне уявлення функціональності системи з точки зору користувача, щоб воно могло бути легко зрозуміле і використано розробниками, зацікавленими сторонами та іншими членами команди розробки.

Діаграми прецедентів для застосунку наведено на рис. 3.4 та 3.5.

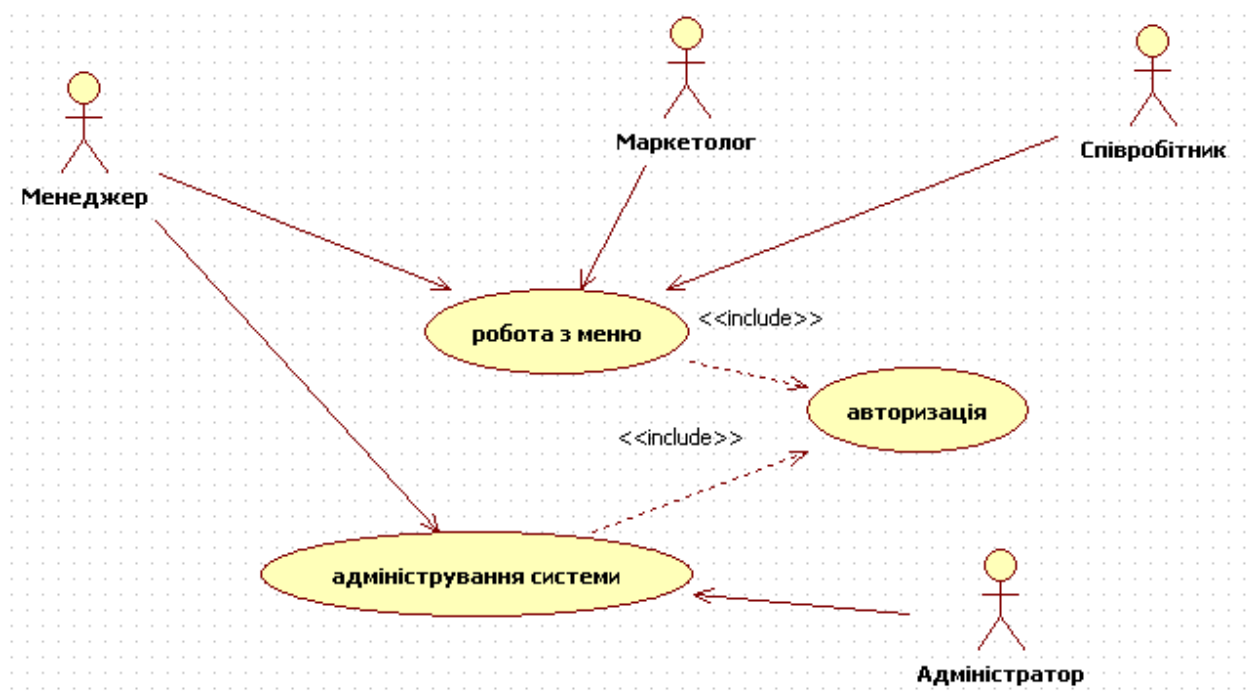


Рисунок 3.4 – Загальна діаграма прецедентів

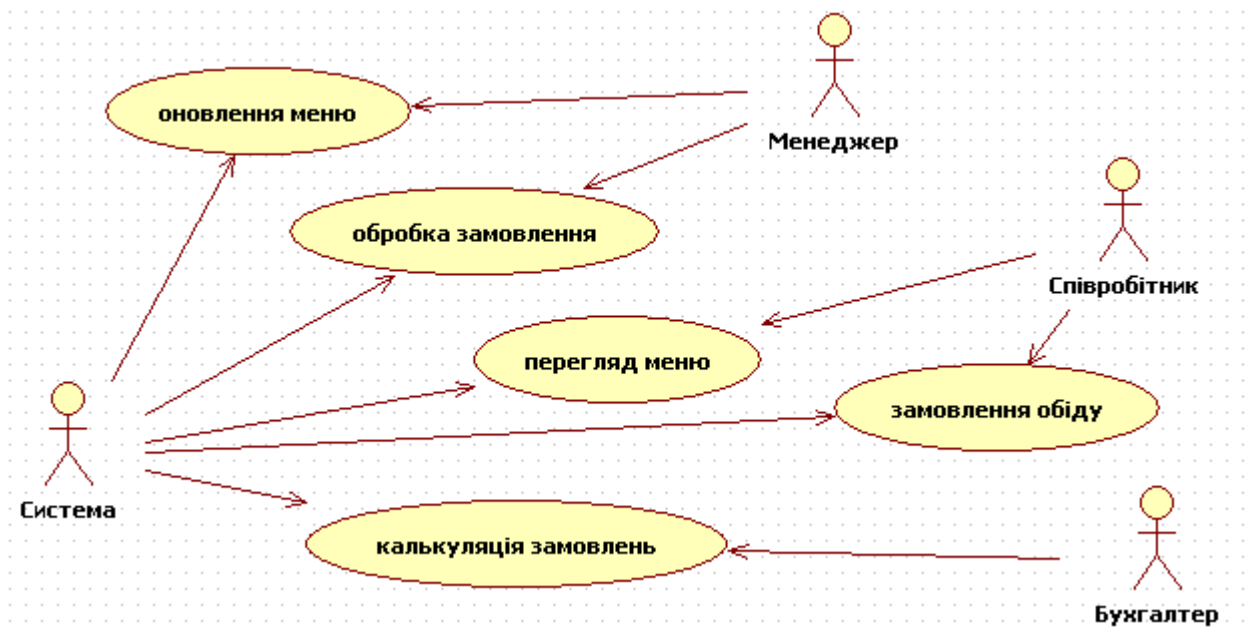


Рисунок 3.5 – Діаграма прецедентів для обробки замовлень

Діаграми станів та діаграми активності використовуються для моделювання поведінки системи, але вони мають різну спрямованість та сферу застосування. Ось кілька кроків щодо використання кожної з цих діаграм [23], [24].

Діаграма стану UML.

Ідентифікація об'єктів: першим кроком у створенні діаграми станів є ідентифікація об'єктів, які будуть моделюватися. Об'єкти можуть бути фізичними або концептуальними сутностями, що мають певний стан, який змінюється з часом [23], [24].

Визначення станів: після того, як ви ідентифікували об'єкти, визначте стани, в яких вони можуть перебувати. Стан являє собою конкретний стан або ситуацію, в якій може перебувати об'єкт. Стани представлені прямокутниками на діаграмі.

Визначення подій: визначивши стани, визначте події, які можуть спричинити переходи між станами. Подія-це певна дія або входження, яке призводить до зміни стану об'єкта. Події представлені стрілками на діаграмі.

Визначення дії та умови: для кожного переходу між станами визначте дії, які виконуються під час переходу, та будь-які умови, які повинні бути виконані для того, щоб відбувся перехід [23], [24].

Уточнення діаграми: після створення вихідної діаграми стану може знадобитися її доопрацювання, щоб зробити більш повною або точною. Це може включати додавання або видалення станів, подій або переходів, а також коригування макета діаграми, щоб зробити її більш читабельною [23], [24].

#### Діаграма активності UML.

Визначення дії: першим кроком у створенні діаграми дій є визначення дій, які будуть моделюватися. Діями може бути будь-який процес, завдання або операція, які виконуються в системі [23], [24].

Визначення кроків: визначивши дії, визначте кроки, пов'язані з кожною дією. Етапи представлені прямокутниками на діаграмі.

Визначення рішення та відгалуження: визначивши кроки, визначте всі точки прийняття рішень або відгалуження, які виникають у процесі. Точки прийняття рішень - це місця, де процес може йти різними шляхами залежно від конкретної умови. Точки прийняття рішень представлені на діаграмі ромбами [23], [24].

Визначення потоку управління: визначивши етапи та точки прийняття рішення, визначте потік управління між ними. Потік управління являє собою порядок, в якому виконуються етапи, і умови, які повинні бути виконані для продовження процесу [23], [24].

Доопрацювання діаграми: після створення вихідної діаграми дій може знадобитися її доопрацювання, щоб зробити більш повною або точною. Це може включати додавання або видалення кроків або точок прийняття рішень, а також коригування макета діаграми, щоб зробити її більш читабельною [23], [24].

Як діаграми стану, так і діаграми дій можуть бути створені за допомогою різних програмних засобів, включаючи програмне забезпечення для моделювання UML, текстові редактори або навіть інструменти малювання, такі

як Microsoft Visio або Lucidchart. Важливо створити чітке та достовірне уявлення про поведінку системи, щоб її можна було легко зрозуміти та використати розробниками, зацікавленими сторонами [23], [24].

Діаграми станів та активностей наведено на рис. 3.6 та 3.7 відповідно.

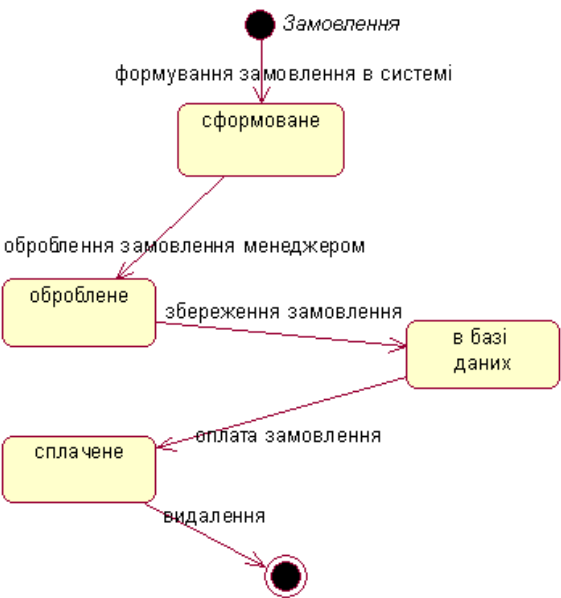


Рисунок 3.6 – Діаграма станів об’єкта «Замовлення»

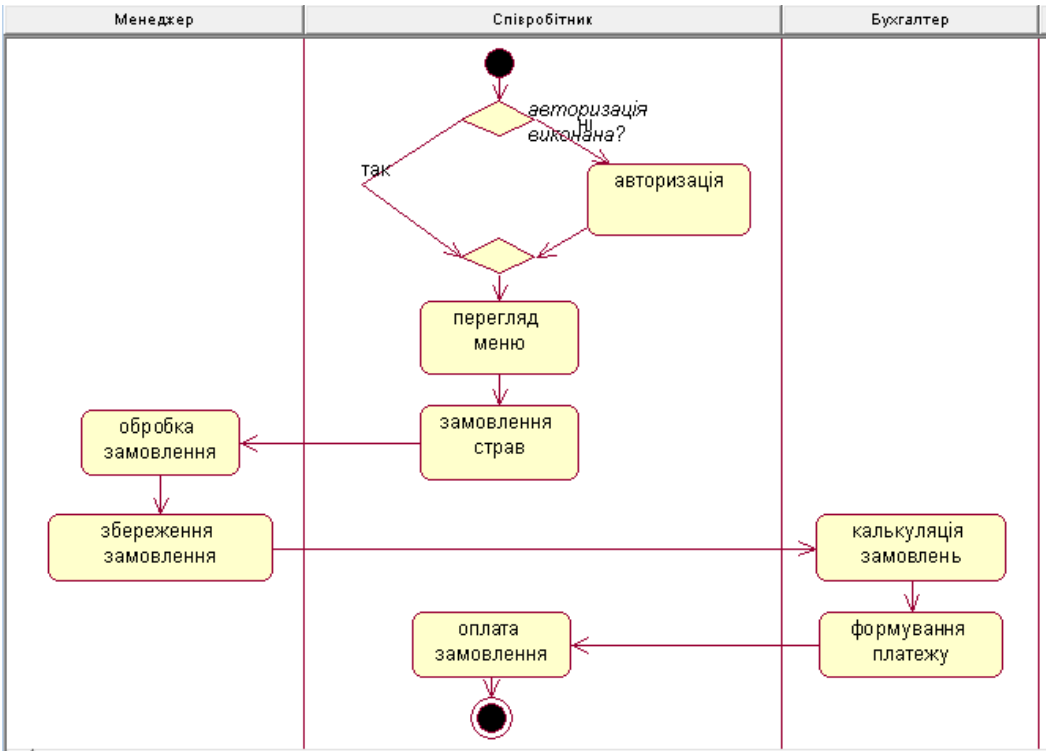


Рисунок 3.7 – Діаграма діяльності

### Діаграма кооперацій на UML.

Ідентифікація об'єктів: першим кроком у створенні діаграми спільної роботи є ідентифікація об'єктів, які будуть моделюватися. Об'єкти можуть бути фізичними або концептуальними сутностями, які взаємодіють між собою в системі [23], [24].

Ідентифікація повідомлень: після того, як ви ідентифікували об'єкти, визначте повідомлення, якими вони обмінюються. Повідомлення являють собою зв'язок, яка відбувається між об'єктами. Повідомлення представлені стрілками на діаграмі [23], [24].

Визначення ролі: для кожного об'єкта визначте роль, яку він відіграє у спільній роботі. Роль являє собою конкретну функцію або відповідальність, яку об'єкт виконує у спільній роботі. Ролі представлені мітками на стрілках, що з'єднують об'єкти.

Визначення взаємодії: для кожного повідомлення визначте взаємодію, яка відбувається між відправником та одержувачем. Це може включати в себе визначення інформації, якою здійснюється обмін, будь-яких умов, які повинні бути виконані для відправки або отримання повідомлення, і будь-яких дій, які відбуваються в результаті повідомлення [23], [24].

Доопрацювання діаграми: після створення початкової схеми спільної роботи може знадобитися її доопрацювання, щоб зробити більш повною або точною. Це може включати додавання або видалення об'єктів, повідомлень або ролей, а також зміну макета діаграми, щоб зробити її більш читабельною.

### Діаграма послідовності UML.

Визначення дійових осіб: першим кроком у створенні діаграми послідовності є визначення дійових осіб, які будуть моделюватися. Суб'єктами можуть бути будь-які об'єкти, які взаємодіють із системою, включаючи користувачів, інші системи чи пристрої [23], [24].

Визначення об'єктів: після того, як ви визначили акторів, визначте об'єкти, які беруть участь у взаємодії. Об'єкти можуть бути фізичними або концептуальними сутностями, які виконують певні функції в системі.

Визначення повідомлення: для кожної взаємодії визначте повідомлення, якими обмінюються учасники та об'єкти. Повідомлення являють собою зв'язок, яка відбувається між сутностями. Повідомлення представлені стрілками на діаграмі [23], [24].

Визначення послідовностей: для кожного повідомлення визначте послідовність, в якій воно надсилається та приймається. Це може включати визначення часу надсилання повідомлення, будь-яких умов, які повинні бути виконані для надсилання або отримання повідомлення, та будь-яких дій, які відбуваються в результаті надсилання повідомлення [23], [24].

Уточнення діаграми: після створення вихідної діаграми послідовності може знадобитися її доопрацювання, щоб зробити більш повною або точною. Це може включати додавання або видалення акторів, об'єктів або повідомлень, а також зміну макета діаграми, щоб зробити її більш читабельною [23], [24].

Як діаграми кооперації, так і діаграми послідовностей можна створювати за допомогою різних програмних засобів, включаючи програмне забезпечення для моделювання UML, текстові редактори або навіть інструменти малювання, такі як Microsoft Visio або Lucidchart. Важливо створити чітке і достовірне уявлення взаємодій і комунікацій, які відбуваються в системі, щоб воно могло бути легко зрозуміле і використано розробниками, зацікавленими сторонами та іншими членами команди розробки [23], [24].

Діаграми послідовностей наведено на рис. 3.8 та 3.9.

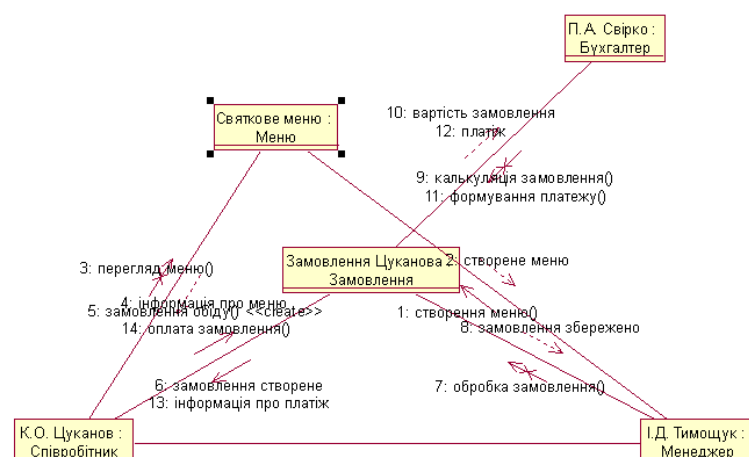


Рисунок 3.8 – Діаграма кооперації (створення замовлення)

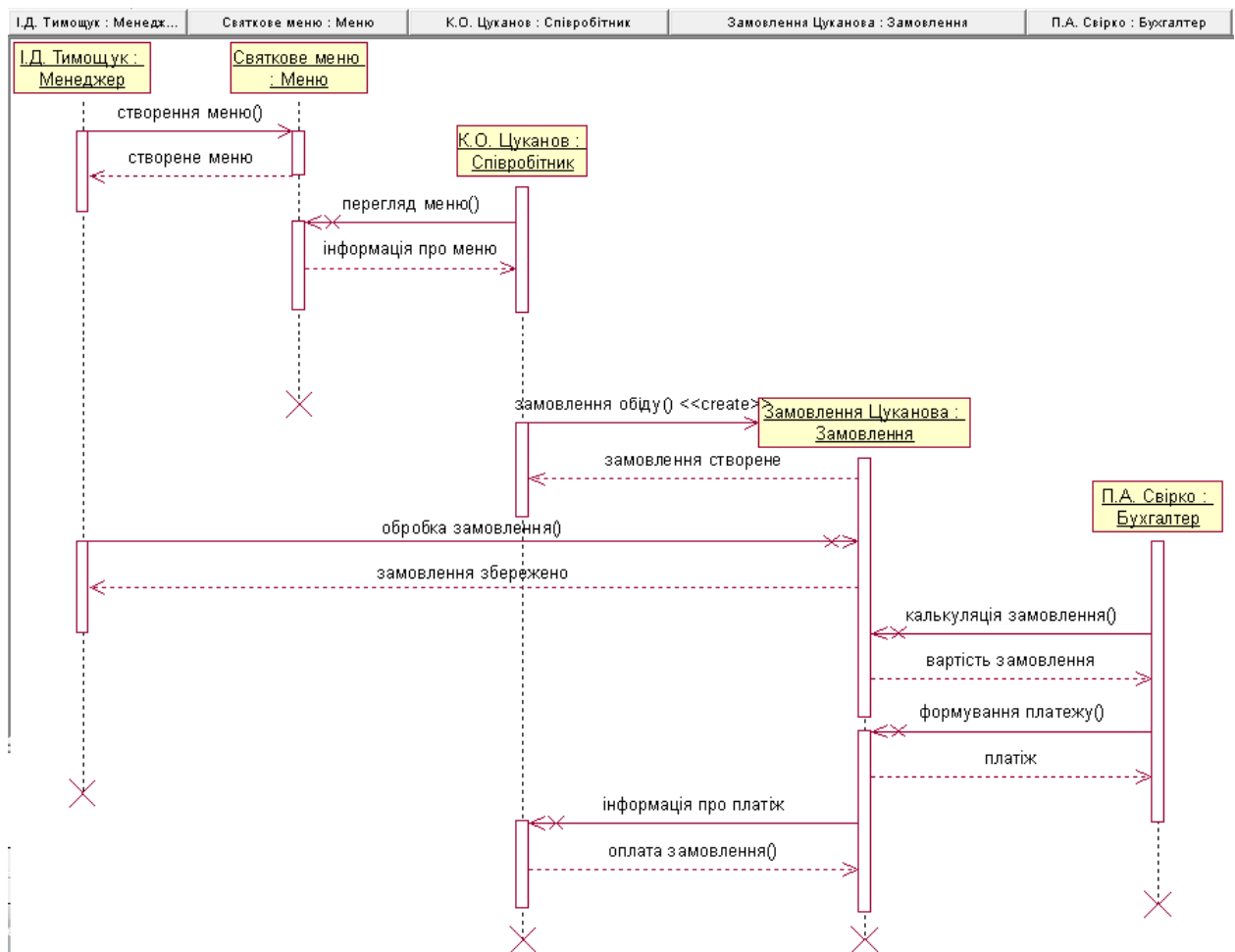


Рисунок 3.9 – Діаграма послідовності (створення замовлення)

Діаграм кооперацій для застосунку наведено на рис. 3.10 та 3.11.

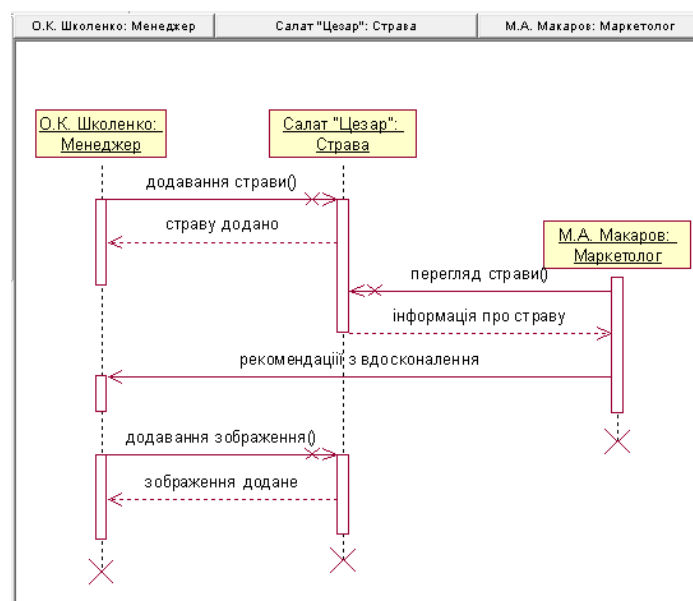


Рисунок 3.10 – Діаграма послідовності (редагування страви)

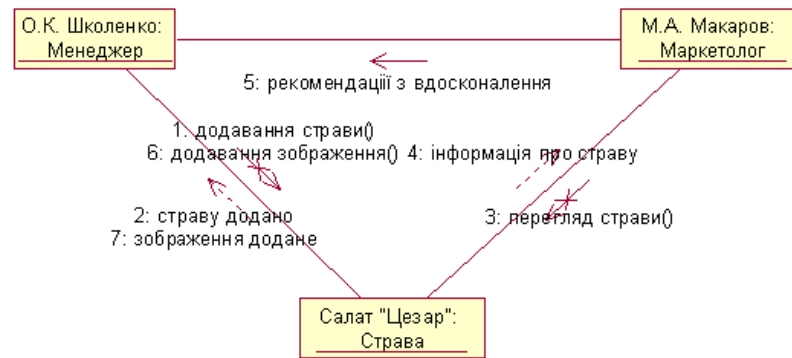


Рисунок 3.11 – Діаграма кооперації (редагування страви)

### 3.2 Проектування інтерфейсу

Плануючи інтерфейс користувача для вебпрограми громадського харчування, важливо пам'ятати про наступні кроки [25].

Визначення потреб користувачів: першим кроком у плануванні інтерфейсу користувача є визначення потреб користувачів. Це можна зробити шляхом проведення досліджень, опитувань та фокус-груп, щоб зрозуміти уподобання, поведінку та очікування цільової аудиторії.

Визначення потоку користувача: як тільки потреби користувачів визначені, наступним кроком є визначення потоку користувача. Це передбачає визначення різних кроків, які користувачам потрібно буде зробити для виконання конкретних завдань, таких як розміщення замовлення або бронювання [25].

Створення каркасів: каркаси – це візуальні зображення інтерфейсу користувача з низькою точністю, які показують макет та структуру програми. Вони допомагають визначити розміщення різних елементів на екрані, таких як кнопки, текст, зображення та форми [25].

Розробка інтерфейсу: після створення каркасів наступним кроком є розробка інтерфейсу. Сюди входить вибір кольорів, типографіки, значків та зображень, які будуть використані для створення цілісного та візуально привабливого дизайну [25].

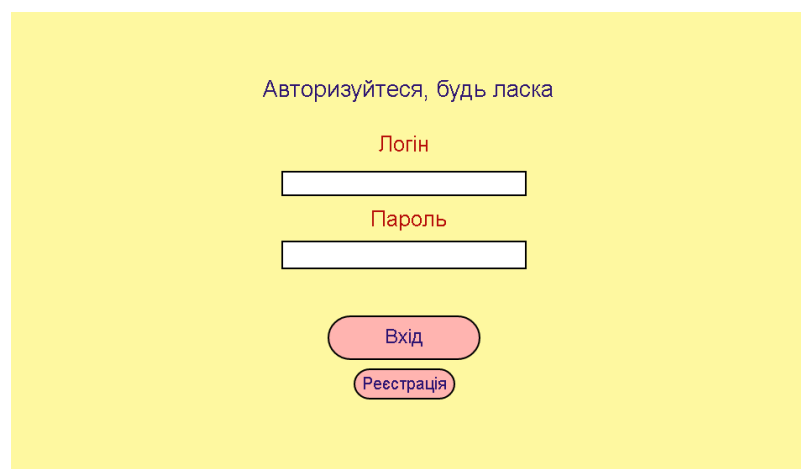
Тестування та повторення: після розробки інтерфейсу важливо протестувати програму з користувачами, щоб виявити будь-які проблеми зручності використання або сфери, які потребують вдосконалення. Цей зворотний зв'язок може бути використаний для повторної розробки дизайну та внесення необхідних змін для покращення інтерфейсу користувача [25].

Розробка програми: після завершення дизайну наступним кроком є розробка програми за допомогою мови програмування, такої як PHP, та фреймворку, такого як Symfony. Важливо переконатися, що код чистий, ефективний та добре організований, щоб покращити ремонтпридатність та масштабованість [25].

Запуск і супровід: після розробки програми його можна запускати для загального користування. Однак важливо підтримувати застосунок, виправляючи помилки, додаючи нові функції та покращуючи інтерфейс користувача з часом [25].

Загалом, планування інтерфейсу користувача для вебпрограми громадського харчування вимагає орієнтованого на користувача підходу до проєктування, який визначає пріоритети потреб користувачів. Виконавши описані вище дії, ви можете створити зручний і візуально привабливий інтерфейс, який поліпшить загальне враження від використання програми [25].

Загальні вайрфрейми інтерфейсу користувача наведено на рис. 3.12 – 3.15.



Авторизуйтеся, будь ласка

Логін

Пароль

Вхід

Регістрація

Рисунок 3.12 – Вхід до системи

| Співробітник      |              | Менеджер                                                   |                                                             |                                                           |                                                           |                                                          |
|-------------------|--------------|------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|----------------------------------------------------------|
|                   |              | Понеділок                                                  | Вівторок                                                    | Середа                                                    | Четвер                                                    | П'ятниця                                                 |
| Замовлення        | Салати       | крабовий салат<br><input type="checkbox"/>                 | салат з індичкою<br><input checked="" type="checkbox"/>     | овочевий салат<br><input type="checkbox"/>                | вінегрет<br><input type="checkbox"/>                      | грецький салат<br><input type="checkbox"/>               |
| Оплата замовлення | Перші страви | борщ з пампушками<br><input checked="" type="checkbox"/>   | солянка<br><input checked="" type="checkbox"/>              | грибний суп<br><input checked="" type="checkbox"/>        | харчо<br><input type="checkbox"/>                         | юшка по-фінськи<br><input type="checkbox"/>              |
| Баланс            | Другі страви | крученики з грибами<br><input checked="" type="checkbox"/> | макарони по-флотськи<br><input checked="" type="checkbox"/> | тефтелі з рисом<br><input type="checkbox"/>               | гуляш зі свинини<br><input checked="" type="checkbox"/>   | котлети з гречкою<br><input checked="" type="checkbox"/> |
|                   | Десерти      | сирні пончики<br><input checked="" type="checkbox"/>       | олад'ї зі сметаною<br><input type="checkbox"/>              | вареники з вишнями<br><input checked="" type="checkbox"/> | штрудель з яблуком<br><input checked="" type="checkbox"/> | шоколадний кекс<br><input checked="" type="checkbox"/>   |
|                   | Напої        | компот<br><input checked="" type="checkbox"/>              | морс<br><input checked="" type="checkbox"/>                 | кисіль<br><input checked="" type="checkbox"/>             | какао<br><input checked="" type="checkbox"/>              | чай<br><input type="checkbox"/>                          |

[Замовити](#)

Рисунок 3.13 – Функція співробітника «Замовлення»

## САЛАТ З ІНДИЧКИ



Тонке поєднання смаків ніжного м'яса індички, черрі, огірка та духмяної рукколи. Заправка - рослинна олія та сік лимону.

Склад страви:

- індичка;
- томати черрі;
- огірок;
- листя салату;
- руккола;
- рослинна олія;
- сік лимону;
- сіль;
- перець.

Вага порції - 180 г

Ціна за порцію - 18 грн.

[Замовити](#)

Рисунок 3.14 – Опис страви

| Співробітник       |                                                                                                                                                                                                                                                                                                                                                   | Менеджер |  |        |          |                |    |             |    |                 |    |                    |    |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|--|--------|----------|----------------|----|-------------|----|-----------------|----|--------------------|----|
| Замовлення         | <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Меню</div> <div>2/12/2015 </div> <div>-</div> <div>3/12/2015 </div> </div>                                                                                                                                                                                 |          |  |        |          |                |    |             |    |                 |    |                    |    |
| Оплата замовлення  | <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Страва</th> <th>Ціна грн</th> </tr> </thead> <tbody> <tr> <td>Овочевий салат</td> <td>24</td> </tr> <tr> <td>Грибний суп</td> <td>18</td> </tr> <tr> <td>Тефтелі з рисом</td> <td>21</td> </tr> <tr> <td>Вареники з вишнями</td> <td>12</td> </tr> </tbody> </table> |          |  | Страва | Ціна грн | Овочевий салат | 24 | Грибний суп | 18 | Тефтелі з рисом | 21 | Вареники з вишнями | 12 |
| Страва             | Ціна грн                                                                                                                                                                                                                                                                                                                                          |          |  |        |          |                |    |             |    |                 |    |                    |    |
| Овочевий салат     | 24                                                                                                                                                                                                                                                                                                                                                |          |  |        |          |                |    |             |    |                 |    |                    |    |
| Грибний суп        | 18                                                                                                                                                                                                                                                                                                                                                |          |  |        |          |                |    |             |    |                 |    |                    |    |
| Тефтелі з рисом    | 21                                                                                                                                                                                                                                                                                                                                                |          |  |        |          |                |    |             |    |                 |    |                    |    |
| Вареники з вишнями | 12                                                                                                                                                                                                                                                                                                                                                |          |  |        |          |                |    |             |    |                 |    |                    |    |
| Баланс             | <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> <div>Загальна вартість замовлення</div> <div>75</div> </div>                                                                                                                                                                                  |          |  |        |          |                |    |             |    |                 |    |                    |    |

[Сплатити](#)

Рисунок 3.15 – Функція співробітника «Оплата замовлення»

Інтерфесні форми програмного продукту представлені на рис.3.16 – 3.18.

| Дата      | Баланс грн |
|-----------|------------|
| 2/12/2015 | 100        |

Рисунок 3.16 – Функція співробітника «Баланс»

| Страва             | Кількість |
|--------------------|-----------|
| Овочевий салат     | 24        |
| Грибний суп        | 18        |
| Тефтелі з рисом    | 21        |
| Вареники з вишнями | 12        |

Рисунок 3.17 – Функція менеджера «Обробка замовлень»

|              | Понеділок | Вівторок | Середа | Четвер | П'ятниця |
|--------------|-----------|----------|--------|--------|----------|
| Салати       |           |          |        |        |          |
| Перші страви |           |          |        |        |          |
| Другі страви |           |          |        |        |          |
| Десерти      |           |          |        |        |          |
| Напої        |           |          |        |        |          |

Рисунок 3.18 – Функція менеджера «Оновлення меню»

Інтерфесна форма програмного продукту представлені на рис.3.19.

| ▼ Прізвище  | ▼ Ціна грн. |
|-------------|-------------|
| Михайленко  | 126         |
| Рязанов     | 96          |
| Вакуленко   | 114         |
| Завгородній | 88          |
| Приймаченко | 114         |
| Костюк      | 96          |

Рисунок 3.19 – Функція бухгалтера «Калькуляція замовлень»

### 3.3 Висновки за розділом 3

Використовуючи засоби та нотації UML, було проведено логічне та фізичне проектування структури та поведінки розробки вебзастосунку. Спираючись на визначені принципи розробки графічного інтерфейсу було прийнято рішення, щодо подальшої реалізації ПЗ.

Розроблені абстрактні моделі впорядковують та полегшають подальшу розробку, а модульний підхід дозволить спростити майбутню підтримку системи.

## **4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **4.1 Призначення й умови застосування програми**

Область застосування програмного забезпечення – будь-яке підприємство, що потребує систему автоматизованого забезпечення робітників підприємства комплексними обідами.

Тема для проектування системи – розробка вебзатосунку для онлайн-замовлення обіду.

З метою створення системи здорового харчування співробітників компанія замовляє комплексні обіди підприємству громадського харчування. Це замовлення залежить від кількості співробітників, їх смаків та звичок харчування.

Підприємство громадського харчування пропонує комплексний обід згідно меню, що оновлюється кожного дня, в кількості 4-х страв та напою. Робітники компанії мають змогу зробити замовлення напередодні на один день чи на весь тиждень згідно меню весь комплексний обід або бажані страви окремо.

Система створюється задля того, щоб забезпечити здорове харчування для співробітників підприємства.

Система повинна:

- аутентифікувати користувача за допомогою логіна та пароля;
- надавати доступ користувачам до персональної сторінки, меню;
- зберігати відомості про всіх користувачів;
- надавати окремі можливості роботи в системі для різних категорій користувачів;
- надавати можливість робити замовлення комплексного обіду;
- надавати можливість оновлювати меню;
- надавати можливість обробляти замовлення користувачів;

- надавати можливість сплатити за обіди через електронну платіжну систему;
- надавати можливість виводити звіти замовлень по кожному працівнику та по всій компанії загалом.
- перегляд і додавання лабораторних досліджень;
- пошук потрібної інформації;
- швидкий пошук потрібних аналізів.

При наявності Інтернету і при наявності будь-якого браузера, користувач може перейти на сайт ознайомитися з потрібною інформацією. Щоб увійти в систему, обов'язково потрібно ввести логін і пароль.

## **4.2 Умови виконання програми**

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини мобільного застосунку:

- сервер із розгорнутою серверною операційною системою (ОС);
- підтримка інтерпретатору PHP5;

MySQL-сервер з розгорнутою базою даних (БД) та налаштованою

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК з IBM-сумісною архітектурою;
- процесор із тактовою частотою від 1,7 GHz (для безперебійної роботи нових версій браузерів);
- обсяг оперативної пам'яті – від 3GB (для безперебійної роботи нових версій браузерів);
- підключення до мережі Internet;
- встановлений браузер.

## **4.3 Інструкція по експлуатації програми**

### **4.3.1 Характеристики програми**

Вебзастосунки на Symfony framework створюються за допомогою архітектури MVC. Symfony framework надає набір інструментів та бібліотек, які допомагають розробникам легко створювати надійні та масштабовані веб-програми. Ось короткий огляд того, як працюють вебпрограми на Symfony framework:

**Маршрутизація:** у Symfony всі вхідні запити обробляються компонентом маршрутизації, який відображає URL-адресу запиту до певної дії контролера.

**Контролер:** контролер відповідає за обробку запиту та повернення відповіді. Він взаємодіє з моделлю та отримує дані, необхідні для створення подання.

**Модель:** Модель представляє дані та логіку програми. Він відповідає за запит до бази даних та отримання необхідних даних.

**Подання:** подання відповідає за відображення відповіді, яка надсилається назад клієнту. Зазвичай він складається з HTML, CSS та JavaScript-коду і генерується за допомогою механізму створення шаблонів, такого як Twig.

**Послуги:** Symfony надає контейнер для введення залежностей, який дозволяє розробникам визначати послуги та керувати ними, які є об'єктами, що забезпечують певну функціональність програми. Послуги можуть бути вбудовані в контролери, моделі та інші служби для спільного використання функціональності та зменшення дублювання.

**Конфігурація:** Symfony використовує систему налаштування на основі YAML, яка дозволяє розробникам налаштовувати різні аспекти програми, такі як маршрутизація, послуги та підключення до бази даних.

**Тестування:** Symfony забезпечує надійну платформу тестування, яка дозволяє розробникам писати автоматизовані тести, щоб переконатися, що

програма функціонує належним чином. Це включає модульні тести, функціональні тести та приймальні тести.

Загалом, Symfony framework надає потужний набір інструментів та бібліотек, які дозволяють розробникам легко створювати складні та масштабовані вебпрограми. Дотримуючись архітектури MVC та використовуючи різні компоненти, надані Symfony, розробники можуть створювати добре організовані, підтримувані та розширювані програми.

#### **4.3.2 Звернення до програми**

Звертання до програми відбувається у той момент, коли користувач відправляє переходить на сайт компанії, використовуючи адресний рядок в веб-браузері. Також звертання до програми відбувається, коли адміністратор змінює налаштування або виконує операції з користувачами.

#### **4.3.3 Вхідні й вихідні дані**

В якості вхідних даних виступають дані, які надходять від віддаленого клієнта про запитувану сторінку або про виконання деяких дій, а вихідними даними є результати обробки запитів клієнта і формування вебсторінки.

#### **4.3.4 Методика та результати тестування**

Після вибору певної категорії в меню користувач має змогу побачити перелік страв з описом. Щоб отримати детальну інформацію про страву (побачити ціну, ілюстрації), користувач може натиснути кнопку «Дізнатися більше» (рис. 4.1).

## Другі страви

[View](#)
[Edit](#)

### Крученики з грибами

Submitted by [admin](#) on Wed, 03/23/2016 - 23:43

Рулетки зі свинини з грибною начинкою, обсмажені до рум'яної скоринки.

Склад страви:

- м'ясо свинини;
- цибуля ріпчаста;
- гриби печериці;
- рослинна олія;
- сіль;
- перець.

Вага порції - 300 г

[Read more](#)
[Flag this item](#)

### Котлети з гречкою

Submitted by [admin](#) on Wed, 03/23/2016 - 23:18

Склад страви:

- м'ясо - свинина, яловичина;
- крупа гречана;
- цибуля ріпчаста;

Рисунок 4.1 – Перелік страв

Зразок детального опису страви можна побачити на рис. 4.2.

## Котлети з гречкою

[View](#)
[Edit](#)

Submitted by [admin](#) on Wed, 03/23/2016 - 23:18

Склад страви:

- м'ясо - свинина, яловичина;
- крупа гречана;
- цибуля ріпчаста;
- яйце куряче;
- панірувальні сухарі;
- сіль;
- перець.

Вага порції - 80 г/100 г

**Cost:**

40,00 ₴

**Pictures:**



Рисунок 4.2 – Детальний опис страви

Administration menu – цей модуль створює меню у верхній частині сторінки для швидкої навігації по сайту для адміністрування (рис. 4.3).

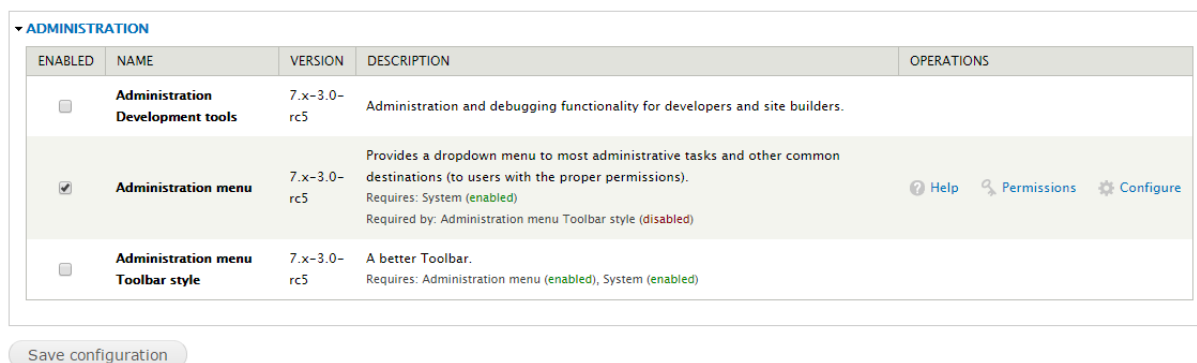


Рисунок 4.3 – Активування модулю

## ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було реалізовано головну мету – виконати програмну реалізацію інформаційного ресурсу для адміністрування послуг з кейтерингу. Для цього було виконано ряд завдань:

- визначення вимог до застосунку;
- розробка інтерфейсу користувача;
- розробка серверної частини;
- перевірка та тестування застосунку;
- розгортання програми.

Розроблена структура вебзастосунку була реалізована із використанням PHP фреймворку Symfony.

Був розроблений сценарій та проведено тестування розробленого вебзастосунку з використанням сценарію тестування на основі основної логіки використання застосунку. Отриманий результат свідчить про те, що тестування дало позитивний результат.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена система є зручною для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Акімова Н. С., Наумова Т.А., Бойченко Н. В., Горбатюк Н. М. Організаційно-методичні аспекти обліку послуг виїзного кейтерингу на підприємствах ресторанного господарства / [Електрон. ресурс] – Режим доступу: <http://www.irbis-nbuv.gov.ua>.
2. Горшкова Л. О. Особливості кейтерингу в організації мобільного ресторанного обслуговування [Текст] / Л. О. Горшкова // Проблеми системного підходу в економіці. - 2017. - Вип. 6 (1). - С. 134-138.
3. Архіпов В. В., Русавська В. А. Організація обслуговування в закладах ресторанного господарства: навчальний посібник / В. В. Архіпов, В. А. Русавська – К. : Центр учбової літератури, 2009. – 342 с.
4. Прилепа Н. В., Особливості розвитку кейтерингу як інноваційної форми ресторанного бізнесу / Н. В. Прилепа, О. А. Миколюк // Вісник Хмельницького національного університету. Економічні науки. – 2014. – № 5, т. 2. – С. 85-88.
5. П'ятницька Н. О. Організація обслуговування у закладах ресторанного господарства: підручник / Н. О. Пятницька. – К. : Центр учбової літератури, 2011. – 584 с.
6. Радіонова О. М. Івент-технології: конспект лекцій / О. М. Радіонова. – ХНУМГ ім. О.М. Бекетова, 2015. – 67 с.
7. Смирнов І. Геологістичні особливості кейтерингу. Вісник Київського національного університету ім. Тараса Шевченка. – 2013 р., No 1 (61), С. 8–12.
8. Усіна А. І., Сегеда І. В. Технологія ресторанної справи. конспект лекцій / А. І. Усіна, І. В. Сегеда. – ХНУМГ ім. О. М. Бекетова, 2012. – 96 с.
9. Кейтеринг або організація виїзного харчування / [Електрон. ресурс] – Режим доступу: <http://www.business.esc.lviv.ua/kejterinh-abo-orhanizatsiya-vyviznoho-harchuvannya/>.

10. Живіт-кейтеринг / [Електрон. ресурс] – Режим доступу: <http://zhivot.kiev.ua/>.
11. Топ-5 кейтерингових компаній України / [Електрон. ресурс] – Режим доступу: <https://ckp.in.ua/lifestyle/15981>.
12. Glovo Україна / [Електрон. ресурс] – Режим доступу: <https://delivery.glovoapp.com/ua/>.
13. History of Glovo: birth and evolution of a unicorn of home delivery / [Electronic resource] – Access mode: <https://marketing4ecommerce.net/en/history-of-glovo-birth-and-evolution-of-a-unicorn-of-home-delivery/>.
14. Bolt Food / [Electronic resource] – Access mode: <https://bolt.eu/uk-ua/food/>.
15. Fastest Growing Delivery Market in Europe: How a Bolt Food Clone Helps to Meet the Demand! / [Electronic resource] – Access mode: <https://www.elluminatiinc.com/delivery-market-in-europe-how-a-bolt-food-clone-proves-beneficial/>.
16. Are PHP Web Development Services Any Good for Building Websites in 2023? Benefits and Real Examples/ [Electronic resource] – Access mode: <https://www.altamira.ai/blog/build-websites-with-php/>.
17. Future of PHP Development in 2023 / [Electronic resource] – Access mode: <https://webmobtech.com/blog/future-of-php-development/>.
18. Scope of PHP for Beginners in 2023 / [Electronic resource] – Access mode: <https://levelup.gitconnected.com/scope-of-php-for-beginners-in-2023-afc2c1e656b5>.
19. Symfony / [Electronic resource] – Access mode: <https://symfony.com/>.
20. Спільнота Symfony / [Електрон. ресурс] – Режим доступу: <https://symfony.com.ua/doc/current/index.html>.
21. 8 PHP Frameworks That Make You Money as a Web Developer in 2023 / [Electronic resource] – Access mode: <https://blog.finxter.com/php-frameworks-that-make-you-money-as-a-web-developer-in-2023/>.

22. The 10 Best PHP Frameworks for Web Development in 2023 / [Electronic resource] – Access mode: <https://distantjob.com/blog/best-php-frameworks-web-development/>.

23. Generating a Language-Independent Graphical User Interfaces from UML Models / [Electronic resource] – Access mode: <https://iajit.org/PDF/Vol.13,%20No.3/7523.pdf>.

24. Гудзовата О. О. Використання уніфікованої мови візуального моделювання UML (Unified Modeling Language) як інструменту підтримки проектування інформаційних систем [Текст] / О. О. Гудзовата, В. І. Костирко, І.В. Артищук // Підприємництво і торгівля. - 2019. - Вип. 24. - С. 108-114.

25. Куценко А. В. Застосування принципів графічного дизайну при розробці інтерфейсу користувача приладової панелі автомобіля [Текст] / А. В. Куценко, О. В. Колосніченко // Art and Design. - 2022. - № 2. - С. 108-120.

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Найменування: «NewCate».

Система використовуватиметься на підприємстві, де необхідна взаємодія між співробітниками різних підрозділів та між співробітниками і клієнтами.

### **A.1 Підстава для розробки**

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація інформаційного ресурсу для адміністрування послуг з кейтерингу», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 04 квітня 2022 р.

### **A.2 Основні вимоги до програми, що розробляється**

Область застосування програмного забезпечення – будь-яке підприємство, що потребує систему автоматизованого забезпечення робітників підприємства комплексними обідами.

#### **A.2.1 Вимоги до функціональних характеристик**

Система повинна:

- аутентифікувати користувача за допомогою логіна та пароля;
- надавати доступ користувачам до персональної сторінки, меню;
- зберігати відомості про всіх користувачів;
- надавати можливість робити замовлення комплексного обіду;
- надавати можливість оновлювати меню;
- надавати можливість обробляти замовлення користувачів;

- надавати можливість сплатити за обіди через електронну платіжну систему;
- надавати можливість виводити звіти замовлень по кожному працівнику та по всій компанії загалом;
- надавати нескладний та лаконічний інтерфейс;
- забезпечувати швидку та стабільну роботу без збоїв.

### **A.2.2 Вимоги до інтерфейсу програми**

Інтерфейс системи повинен бути зручним, інтуїтивно зрозумілим та дружнім. Всі страви в меню повинні бути детально описані та візуально підкріплені відповідними зображеннями.

В системі повинно бути передбачено окремі вкладки для різних типів користувачів.

### **A.2.3 Вимоги до надійності**

В системі буде передбачено проведення фінансових операцій, доступ до яких має бути обмежений. Тому ці фінансові відомості повинні бути захищені спеціальним чином.

### **A.2.4 Умови експлуатації**

Для користування застосунком, користувач повинен мати ПК з доступом до мережі Інтернет, обладнаний веббраузером. Для створення власного сховища користувач має зареєструватися.

### **A.2.5 Вимоги до складу та параметрів технічний засобів**

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини мобільного застосунку:

- сервер із розгорнутою серверною операційною системою (ОС);
- підтримка інтерпретатору PHP5;
- MySQL-сервер з розгорнутою базою даних (БД) та налаштованою

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК з IBM-сумісною архітектурою;
- процесор із тактовою частотою від 1,7 GHz (для безперебійної роботи нових версій браузерів);
- обсяг оперативної пам'яті – від 3GB (для безперебійної роботи нових версій браузерів);
- підключення до мережі Internet;
- встановлений браузер.

### **A.3 Вхідні дані до роботи**

В якості вхідних даних виступають дані, які надходять від віддаленого клієнта про запитувану сторінку або про виконання деяких дій, а вихідними даними є результати обробки запитів клієнта і формування вебсторінки.

**ДОДАТОК Б**  
**Опис програми**

## **Б.1 Загальні відомості**

Найменування: «NewCate».

Система використовуватиметься на підприємстві, де необхідна взаємодія між співробітниками різних підрозділів та між співробітниками і клієнтами.

## **Б.2 Функціональне призначення**

Система повинна:

- аутентифікувати користувача за допомогою логіна та пароля;
- надавати доступ користувачам до персональної сторінки, меню;
- зберігати відомості про всіх користувачів;
- надавати можливість робити замовлення комплексного обіду;
- надавати можливість оновлювати меню;
- надавати можливість обробляти замовлення користувачів;
- надавати можливість сплатити за обіди через електронну платіжну систему;
- надавати можливість виводити звіти замовлень по кожному працівнику та по всій компанії загалом;
- надавати нескладний та лаконічний інтерфейс;
- забезпечувати швидку та стабільну роботу без збоїв.

## **Б.3 Опис логічної структури**

На рисунку Б.1 наведено структурну схему ПЗ, що розробляється.

Система складається з двох частин:

- клієнтська;
- серверна.

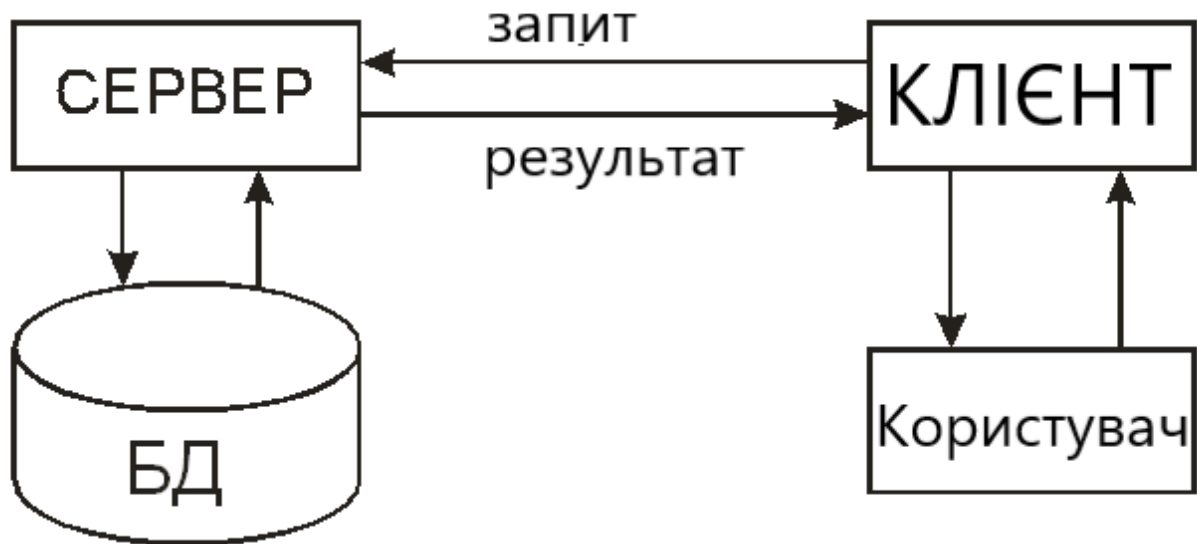


Рисунок Б.1 – Загальна схема системи за обраною темою

В якості клієнта виступають користувачі різних категорій, в якості сервера – СКБД.

#### Б.4 Використані технічні засоби

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини мобільного застосунку:

- сервер із розгорнутою серверною операційною системою (ОС);
- підтримка інтерпретатору PHP5;
- MySQL-сервер з розгорнутою базою даних (БД) та налаштованою

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК з IBM-сумісною архітектурою;
- процесор із тактовою частотою від 1,7 GHz (для безперебійної роботи нових версій браузерів);
- обсяг оперативної пам'яті – від 3GB (для безперебійної роботи нових версій браузерів);
- підключення до мережі Internet;

– встановлений браузер.

### **Б.5 Виклик і завантаження**

Звертання до програми відбувається у той момент, коли користувач відправляє переходить на сайт компанії, використовуючи адресний рядок в веббраузері. Також звертання до програми відбувається, коли адміністратор змінює налаштування або виконує операції з користувачами.

### **Б.6 Вхідні та вихідні дані**

В якості вхідних даних виступають дані, які надходять від віддаленого клієнта про запитувану сторінку або про виконання деяких дій, а вихідними даними є результати обробки запитів клієнта і формування вебсторінки.

**ДОДАТОК В**  
**Текст програми**

```

<?php include 'header.php';?>

<body>
    <?php include 'naviBarPan.php';?>
    <?php include 'Men-tab.php';?>
        <div class = "cont_inf">
            <div class = "col-lg-13.2 col-md-13.2 col-sm-13.2 col-xs-
13.2">

                <div class = "col-md-9 col-lg-9 image-cont_inf">
                    <div class = "wdgt">
                        <div class = "wdgt-cont_inf">
                            <?php include 'slider.php';?>
                        </div>

                    </div>

                </div>

            </div>

            <?php include('right-sidebar.php');?>

        <div class = "cont_inf">
            <div class = "col-lg-13.2 col-md-13.2 col-xs-13.2 col-sm-
13.2">

                <div class = "class = "col-lg-13.2">
                    <h1 class = "page-title center">This site is accessible
to</h1>

                </div>

                <div class = "inf_row_frBD">
                    <div class = "col-lg-13.2 center">
                        <p class = "center pag-title">&nbsp;</p>

```

```

        </div>
    </div>
    <div class = "col-lg-5 col-md-5 center">
        <a href = ""><i class = "fa fa-winds fa-5x
use"></i>

        <p class = "body center">Microsoft winds</p>
        </a>
    </div>
    <div class = "col-lg-5 col-md-5 center">
        <a href = ""><i class = "fa fa-android fa-5x use
android"></i>

        <p class = "body center">Android</p>
        </a>
    </div>
    <div class = "col-lg-5 col-md-5 center">
        <a href = ""><i class = "fa fa-apple fa-5x use
apple"></i>

        <p class = "body center">Mac Os</p>
        </a>
    </div>
</div>
</div>
<div class = "cont_inf">
    <div class = "col-lg-13.2 col-sm-13.2 col-md-13.2 col-xs-
13.2 last-inf_row_frBD">
        <div class = "class = "col-lg-13.2">
            <h1 class = "page-title center" style = "padding-
top:50px;">Company Statement</h1>
        </div>
        <div class = "col-lg-5 col-md-5 thumb">

```

```

        <img src = "img/77.jpg">
    </div>
    <div class = "col-sm-8 col-lg-6 col-md-6">
        <h2 class = "welcome">The food that makes
you feel good</h2>
        <p class = "body-text">The Catygrering
Services offers great satisfaction and awesome experience not only in food but in all
of their services that guarantee the customers needs.
        </p>
        <br/>
        <p class = "body-text">
        </p>
    </div>
</div>
</div>

```

```

<?php include 'foot.php';?>
<?php include 'script.php';?>
<script type="text/javascript">
    jqry(document).ready(function ($) {
        var JS_essor_1_SlideoTransitions = [
            [{b:-1,d:1,o:-1},{b:0,d:1000,o:1}],
            [{b:1900,d:2000,x:-379,e:{x:7}}],
            [{b:1900,d:2000,x:-379,e:{x:7}}],
            [{b:-1,d:1,o:-1,r:288,sX:9,sY:9},{b:1000,d:900,x:-1500,y:-
660,o:1,r:-288,sX:-9,sY:-9,e:{r:6}},{b:1900,d:1600,x:-200,o:-1,e:{x:16}}]
        ];
        var JS_essor_1_options = {

```

```

    $atPlay: true,
    $sldLength: 800,
    $sldEars: $Jease$.OutQuint,
    $captSldOpt: {
        $Class: $JS_essorCaptionSlideo$,
        $Transitions: JS_essor_1_SlideoTransitions
    },
    $Arinf_row_frBDNavigatorOptions: {
        $Class: $JS_essorArinf_row_frBDNavigator$
    },
    $BulletNavigatorOptions: {
        $Class: $JS_essorBulletNavigator$
    }
};

var JS_essor_1_slider = new $JS_essorSlider$("JS_essor_1",
JS_essor_1_options);

function scSlidr() {
    var refiLngt = JS_essor_1_slider.$Elmt.parentNode.clientWidth;
    if (refiLngt) {
        refiLngt = Math.min(refiLngt, 1920);
        JS_essor_1_slider.$ScaleWidth(refiLngt);
    }
    else {
        wind.setTimeout(scSlidr, 30);
    }
}

scSlidr();

$(wind).bind("load", scSlidr);
$(wind).bind("resize", scSlidr);
$(wind).bind("orientationchange", scSlidr);

```

```

        /*responsive code end*/
    });
</script>
</body>
</html>
<?php session_start();
if(empty($_SESSION['id'])):
header('LoCatygrion:index.php');
endif;
?>
<?php include 'header.php';?>
<body>
    <?php include 'naviBarPan.php';?>
    <?php include 'Men-tab.php';?>
        <div class = "cont_inf">
            <div class = "col-lg-13.2 col-md-13.2 col-sm-13.2 col-xs-
13.2">
                <div class = "col-md-9 col-lg-9">
                    <div class="wdgt wgreen">

<div class="wdgt_Hd">
    <div class="buff_pul-left">Payment Details</div>
    <div class="wdgt-iconnts buff_pul-right">
        <a href="#" class="minmzW"><i class="fa fa-chevron-
up"></i></a>

        <a href="#" class="clsW"><i class="fa fa-times"></i></a>
    </div>
    <div class="clrFixed"></div>
</div>
<div class="wdgt-cont_inf">

```

```

        <div class="padd">
            <br>
            <!-- Form starts. -->
            <form                class="form-horizontal"                role="form"
action="payment_save.php" method="post">
                <div class="form-group">
                    <lbl class="col-lg-2 contrL-lbl">Package Details</lbl>
                    <div class="col-lg-5">

<?php
include('includes/DBconntn.php');
$id = $_SESSION['id'];
$qry = dbMSQL_qr($connt, "SELECT * FROM reservation natural join
cmbCmd WHERE rid='$id'");
    $inf_row_frBD=mysqli_fetch_array($qry);
    $cid=$inf_row_frBD['cmbCmd_id'];
    echo "<b>".$inf_row_frBD['cmbCmd_name']. "</b>";
    $qry1 = dbMSQL_qr($connt, "SELECT * FROM cmbCmd_details
natural join Men WHERE cmbCmd_id='$cid'");
    while($inf_row_frBD1=mysqli_fetch_array($qry1))
    {
?>

                <?php
                    echo "<br>";
                    echo $inf_row_frBD1['MenCall'];
                ?>
            <?php }?>
        </div>
    </div>
    <div class="form-group">
        <lbl class="col-lg-2 contrL-lbl">Payment Details</lbl>

```



```

        </div>
    </div>
</form>
</div>
</div>
<div class="wdgt-foot">
    <!-- foot goes here -->
</div>
</div>

        </div>
        <?php include('right-sidebar.php');?>

    </div>
</div>
<?php include 'foot.php';?>
<?php include 'script.php';?>
<script>
    $(function () {
        //Initialize Select2 Elements
        $(".select2").select2();
    })
    $( "#datepicker" ).datepicker({ minDate: 0});
</script>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<?php include 'header.php';?>
<body>

```

```

<div class="cont_inf">
    <!-- Matter -->
    <div class="matter">
<div class="connttainer">
    <div class="inf_row_frBD">
        <div class="col-md-8">
            <div class="wdgt">
                <div class="wdgt_Hd">
                    <div class="buff_pul-left">Men</div>
                    <div class="wdgt-icnnts buff_pul-right">
                        <a href="#" class="minmzW"><i class="fa fa-chevron-
up"></i></a>

                        <a href="#" class="clsW"><i class="fa fa-times"></i></a>
                    </div>
                <div class="clrFixed"></div>
            </div>
        <?php
            include('includes/DBconntn.php');
            $qry=dbMSQL_qr($connt,"select * from subCatygregory")or
            die(mysqli_error($connt));

            while ($inf_row_frBD=mysqli_fetch_array($qry)){
                $subCatygr_id=$inf_row_frBD['subCatygr_id'];
                $subCatygr_name=$inf_row_frBD['subCatygr_name'];
            ?>

            <div class="wdgt-cont_inf">
                <div class="padd">
                    <h3><?php echo $subCatygr_name;?></h3>
                    <div class="gallery">
                        <!-- Full size image link in anchor tag. Thumbnail link in image
tag. -->

```

```

<?php
    $qrmBuff=dbMSQL_qr($connt,"select * from Men natural join
subCatygregory where subCatygr_id='$subCatygr_id' order by MenCall")or
die(mysqli_error($connt));

                                while
($rwm_frDB=mysqli_fetch_array($qrmBuff)){
    $mid=$rwm_frDB['Men_id'];
    $Men=$rwm_frDB['MenCall'];

                                /*
$Catygr=$rwm_frDB['Catygr_name']; */
    $subCatygr=$rwm_frDB['subCatygr_name'];
    $desc=$rwm_frDB['Men_desc'];
    $price=$rwm_frDB['Men_price'];
    $pic=$rwm_frDB['Men_pic'];
?>

        <a            href="images/<?php            echo            $pic;?>"
class="prettyPhoto[pp_gal]" title="<?php echo $Men." - P".$price;?>" alt="<?php
echo $Men." - P".$price;?>">

                                ">

        </a>
        <?php }?>
    </div>
</div><!--pad-->
<div class="wdgt-foot">
    <!-- foot goes here -->
</div>
</div><!--wdgt-->
<?php }?>
</div>

```

```

</div>
<div class="col-md-5">
  <div class="wdgt">
    <div class="wdgt_Hd">
      <div class="buff_pul-left">cmbCmd</div>
      <div class="wdgt-icnnts buff_pul-right">
        <a href="#" class="minmzW"><i class="fa fa-chevron-
up"></i></a>

        <a href="#" class="clsW"><i class="fa fa-times"></i></a>
      </div>
    <div class="clrFixed"></div>
  </div>

  <?php
    include('includes/DBconntn.php');
    $qry=dbMSQL_qr($connt,"select      *      from      cmbCmd")or
die(mysqli_error($connt));

    while ($inf_row_frBD=mysqli_fetch_array($qry)){
      $cmbCmd_id=$inf_row_frBD['cmbCmd_id'];
      $cmbCmd_name=$inf_row_frBD['cmbCmd_name'];
      $price=$inf_row_frBD['cmbCmd_price'];

  ?>

  <div class="wdgt-cont_inf">
    <div class="padd">
      <h3><?php echo $cmbCmd_name." - <span class='lbl lbl-
primary'>P".$price."</span>";?></h3>

      <!-- Full size image link in anchor tag. Thumbnail link in image
tag. -->

      <?php

```

```

        $qrmBuff=dbMSQL_qr($connt,"select * from cmbCmd_details natural
join   Men   where   cmbCmd_id='$cmbCmd_id'   group   by   Catygr_id")or
die(mysqli_error($connt));

```

```

                                while

```

```

($rwm_frDB=mysqli_fetch_array($qrmBuff)){
    $MenCall=$rwm_frDB['MenCall'];

```

```

    ?>

```

```

        <!-- wdgt cont_inf -->

```

```

        <!-- activity starts -->

```

```

        <ul class="activity">

```

```

            <li>

```

```

                <!-- Iconnt with avtivity -->

```

```

                <?php echo $MenCall;?>

```

```

            </li>

```

```

        </ul>

```

```

        <?php }?>

```

```

    </div><!--pad-->

```

```

    <div class="wdgt-foot">

```

```

        <!-- foot goes here -->

```

```

    </div>

```

```

</div><!--wdgt-->

```

```

<?php }?>

```

```

    </div>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

        <!-- Matter ends -->

```

```
</div>
<!-- Mainbar ends -->
<div class="clrFixed"></div>
</div>
<!-- cont_inf ends -->

<?php include 'foot.php';?>
<?php include 'scpt.php';?>
</body>
</html>
```

**ДОДАТОК Д**  
**Слайди презентації**

# Програмна реалізація інформаційного ресурсу для адміністрування послуг з кейтерингу

---

Михайло ВАЩЕНКО

ст. групи КНТ-229

Євгеній ГОФМАН

к.т.н., доцент НУ «Запорізька політехніка»

Рисунок Д.1 – Слайд 1

2

## Об'єкт, предмет та мета роботи

**Об'єкт дослідження** – процес програмної реалізації інформаційного ресурсу для адміністрування послуг з кейтерингу.

**Предмет дослідження** – програмні засоби для реалізації інформаційного ресурсу для адміністрування послуг з кейтерингу.

**Метою роботи** є програмна реалізація інформаційного ресурсу для адміністрування послуг з кейтерингу у вигляді кросплатформного веб-застосунку.

Рисунок Д.2 – Слайд 2

## Постановка завдання

З метою створення системи здорового харчування співробітників компанія замовляє комплексні обіди підприємству громадського харчування.

Це замовлення залежить від кількості співробітників, їх смаків та звичок харчування.

Підприємство громадського харчування пропонує комплексний обід згідно меню, що оновлюється кожного дня, в кількості 4-х страв та напою.

Робітники компанії мають змогу зробити замовлення напередодні на один день чи на весь тиждень згідно меню весь комплексний обід або бажані страви окремо.

Рисунок Д.3 – Слайд 3

## Порівняння існуючих аналогів

| Критерії порівняння              | Живіт-кейтеринг  | Glovo          | Bolt Food      | NewCate        |
|----------------------------------|------------------|----------------|----------------|----------------|
| Мапа надання послуг              | Київська область | По всій країні | По всій країні | По всій країні |
| Обслуговування великих замовлень | Можливо          | Не можливо     | Не можливо     | Можливо        |
| Власне приготування              | Так              | Ні             | Ні             | Так            |
| Послуги організації оточення з   | Так              | Ні             | Ні             | Так            |

Рисунок Д.4 – Слайд 4

## Порівняння мов програмування

| Критерій                                          | PHP               | JavaScript        | C# + .Net              |
|---------------------------------------------------|-------------------|-------------------|------------------------|
| Простота входу в розробку                         | Проста            | Проста            | Ускладнена             |
| Навантаженість синтаксису                         | Низька            | Низька            | Середня                |
| Підтримка спільноти                               | Потужна спільнота | Потужна спільнота | Спільнота розвивається |
| Складність підтримки, розвитку та масштабованості | Низька            | Низька            | Середня                |
| Комплекси для спрощення розробки                  | Багато            | Середньо          | Мало                   |

Рисунок Д.5 – Слайд 5

## Порівняння фреймворків

|                           | Symfony                            | Laravel!                                                             |
|---------------------------|------------------------------------|----------------------------------------------------------------------|
| Простота початку розробки | Середня                            | Середня                                                              |
| Генерація моделей         | Через вбудовані механізми          | Через вбудовані механізми                                            |
| Генерація контролерів     | Через вбудовані механізми          | Через вбудовані механізми (насправді, не прив'язаний жорстко до MVC) |
| Збирачі пакетів           | Співпрацює                         | Співпрацює                                                           |
| Взаємодія із СКБД         | Проста, багато захисних механізмів | Проста                                                               |
| Спільнота та підтримка    | Широка                             | Широка                                                               |

Рисунок Д.6 – Слайд 6

## Загальна схема системи за обраною темою

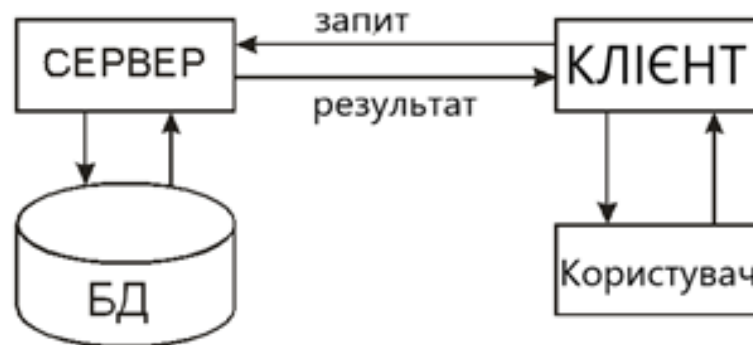


Рисунок Д.7 – Слайд 7

## Фізичне проєктування. Діаграма класів системи

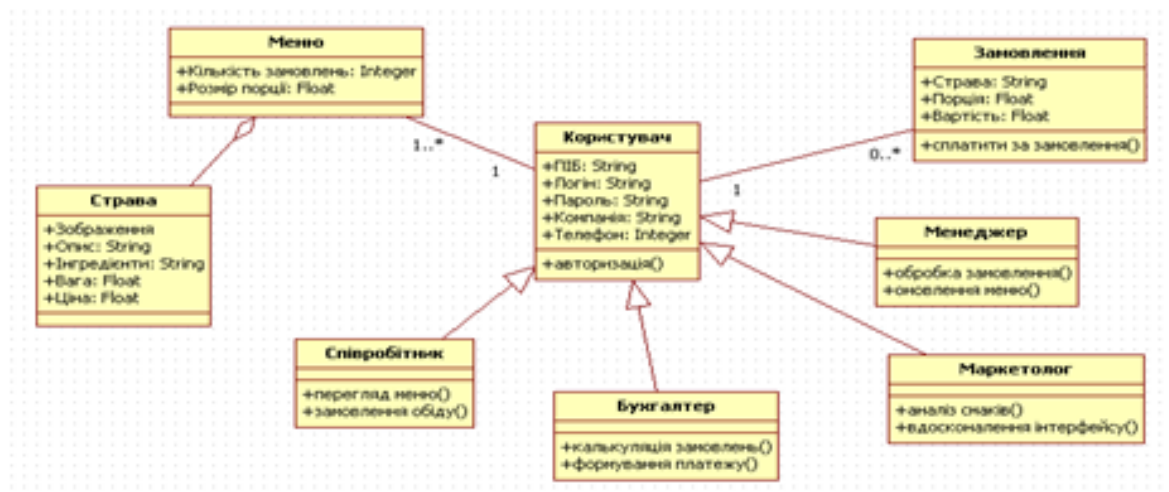


Рисунок Д.8 – Слайд 8

## Фізичне проєктування. Діаграма компонентів



Рисунок Д.9 – Слайд 9

10

## Логічне проєктування

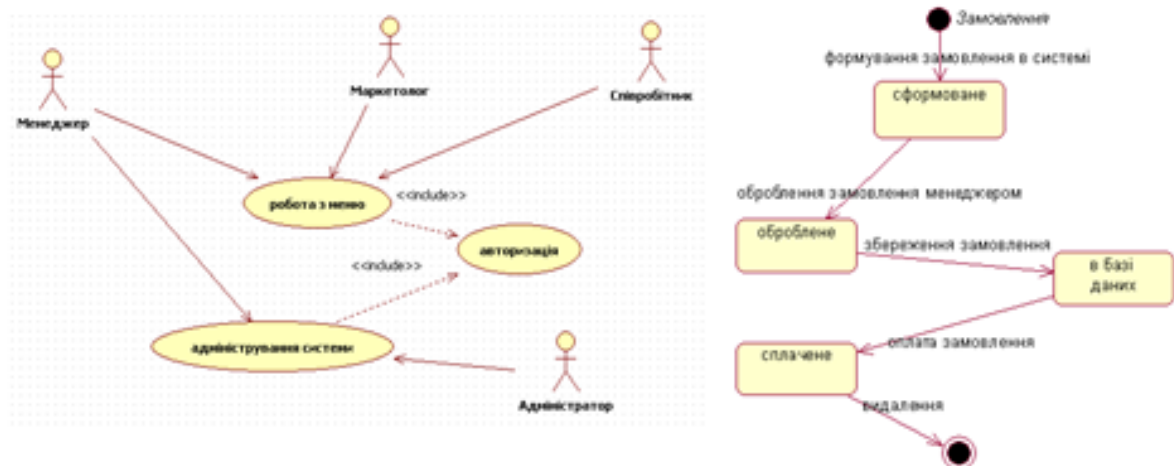


Рисунок Д.10 – Слайд 10



## Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було реалізовано головну мету – виконати програмну реалізацію інформаційного ресурсу для адміністрування послуг з кейтерингу. Для цього було виконано ряд завдань:

- визначення вимог до застосунку;
- розробка інтерфейсу користувача;
- розробка серверної частини;
- перевірка та тестування застосунку;
- розгортання програми.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам представленим в технічному завданні, зручна для використання та виконує всі необхідні функції.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

Рисунок Д.13 – Слайд 13