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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмна платформа для закладів харчування на основі 3D візуалізації»

Виконав:
студент IV курсу, групи КП-92
Сусленко Матвій Олегович _____

Керівник:
Доцент кафедри ПЗКС, д.т.н., доцент,
Сулема Євгенія Станіславівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Старший викладач кафедри ПМА,
Темнікова Олена Леонідівна _____

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

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Сусленку Матвію Олеговичу

1. Тема проєкту «Програмна платформа для закладів харчування на основі 3D візуалізації», керівник проєкту Сулема Євгенія Станіславівна, д.т.н., доцент, затверджені наказом по університету від «31» травня 2023 р. №2107-с.

2. Термін подання студентом проєкту «19» червня 2023 р.

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

4. Зміст пояснювальної записки:
   - аналіз предметної області та існуючих рішень;
   - аналіз мов програмування, ігрових рушіїв та технологій розроблення програмних продуктів, що використовують тривимірну графіку;
   - розроблення програмної платформи для закладів харчування на основі 3D візуалізації;
   - аналіз розроблених алгоритмів та підпрограм;
   - аналіз розробленої програмної платформи.

5. Перелік обов'язкового графічного матеріалу:
   - створення замовлення клієнтом закладу харчування (креслення);
   - надання унікального номера-ідентифікатора сформованому замовленню (креслення);

- структура сторінок клієнтської частини програмної платформи (плакат).
- структура сторінки серверної частини програмної платформи (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «30» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2022	
2.	Розроблення та узгодження технічного завдання	25.11.2022	
3.	Розроблення структури програмного модуля	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2023	
7.	Програмна реалізація програмного модуля	10.03.2023	
8.	Тестування програмного модуля	17.03.2023	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2023	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2023	
11.	Підготовка графічної частини дипломного проєкту	21.04.2023	
12.	Оформлення документації дипломного проєкту	26.05.2023	

Студент Матвій СУСЛЕНКО

Керівник проєкту Євгенія СУЛЕМА

АНОТАЦІЯ

Цей дипломний проєкт присвячений створенню Програмної платформи для закладів харчування на основі 3D візуалізації.

Розробка представляє собою програмний модуль, який може бути інтегрований на вебсайт, використовуватись як додаток на інтерактивних стендах закладів харчування або на мобільних девайсах та додаток-сервер для оброблення створених замовлень. Програмний продукт призначений для автоматизації створення замовлення клієнтом закладу харчування та надання унікального користувацького досвіду за рахунок використання тривимірної графіки; отримання та обробки замовлення працівниками закладу харчування.

Функціональність програмного модуля забезпечує процес формування замовлення клієнтом закладу харчування з можливістю обирати готові страви та напої, а також створювати страви зі списку окремих інгредієнтів. Серверна частина додатку отримує інформацію про сформоване замовлення та надає клієнтові закладу харчування унікальний номер його заявки.

У дипломному проєкті було розроблено: архітектуру програмного модуля; алгоритм створення замовлення користувачем; алгоритм обробки замовлення серверною частиною системи; тривимірні моделі напоїв, страв та харчових продуктів для наочного їх подання; дизайн графічного інтерфейсу клієнтської та серверної частини; систему локалізації програмного модуля з можливістю обрання мови графічного інтерфейсу.

ABSTRACT

This diploma project is devoted to the creation of a software platform for creating an order in a restaurant based on 3D visualization.

The project is a software module that can be integrated into a website, used as an application on interactive stands of restaurants or on mobile devices, and a server application for processing of created orders. This system is designed to automate the creation of an order by a client of a restaurant and provide a unique user experience through the use of three-dimensional graphics; receiving and processing of the order by employees of the restaurant.

The functionality of the software module ensures the process of forming an order by a client with the possibility to choose ready-made dishes and drinks, as well as create dishes from a list of ingredients. The server part of the application receives information about the created order and provides the client of the restaurant with a unique number of his order.

Parts of the diploma project that were developed: software module architecture; algorithm for creating an order by the customer of a restaurant; order processing algorithm by the server part of the system; three-dimensional models of drinks, dishes and food ingredients for their visual presentation; client and server graphic user interface design; localization system of the software module with the ability to select the language of the interface.

ДП.045440-01-90 Програмна платформа для закладів харчування на основі 3D візуалізації. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмна платформа	5	
	для закладів харчування		
	на основі 3D візуалізації.		
	Технічне завдання		
ДП.045440-03-81	Програмна платформа	7	
	для закладів харчування		
	на основі 3D візуалізації.		
	Пояснювальна записка		
ДП.045440-04-51	Програмна платформа	4	
	для закладів харчування		
	на основі 3D візуалізації.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмна платформа	10	
	для закладів харчування		
	на основі 3D візуалізації.		
	Керівництво користувача		
ДП.045440-06-99	Програмна платформа	1	
	для закладів харчування		
	на основі 3D візуалізації.		
	Надання унікального		
	номера-ідентифікатора		
	сформованому замовленню.		
	Схема алгоритму		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Програмна платформа	1	
	для закладів харчування		
	на основі 3D візуалізації.		
	Model View Presenter.		
	Діаграма класів		
ДП.045440-08-98	Програмна платформа	1	
	для закладів харчування		
	на основі 3D візуалізації.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ЗАКЛАДІВ ХАРЧУВАННЯ НА ОСНОВІ 3D ВІЗУАЛІЗАЦІЇ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Матвій СУСЛЕНКО

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмна платформа для закладів харчування на основі 3D візуалізації.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

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

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмний продукт призначений для використання закладами харчування в якості платформи для створення та обробки замовлень.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмна платформа повинна забезпечувати такі основні функції:

1) створення замовлення клієнтом закладу харчування;
2) надання унікального ідентифікатора сформованому замовленню;
3) можливість розширення структури програмного модуля та сервера;

Розробку виконати на платформі Unity.

Додаткові вимоги:

1) наявність динамічного меню;
2) наявність анімованих кнопок;
3) наявність локалізації з можливістю обрати мову інтерфейсу;
4) наявністю можливості додавати готові страви до замовлення;
5) наявністю можливості додавати напої до замовлення;
6) наявністю можливості створювати страви з окремих інгредієнтів;
7) присвоєння сформованому замовленню унікального номера.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Надання унікального номера-ідентифікатора сформованому замовленню. Схема алгоритму»;
- «Model View Presenter. Схема архітектури».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи ……16.11.2022

Розроблення та узгодження технічного завдання ……. 27.11.2022

Розроблення структури програмного модуля …… 15.12.2022

Розроблення дизайну сторінок та графічних елементів …….. 03.02.2023

Програмна реалізація програмного модуля ……... 15.03.2023

Тестування програмного модуля …… 07.04.2023

Підготовка матеріалів текстової частини проєкту …… 28.04.2023

Підготовка матеріалів графічної частини проєкту ……... 12.05.2023

Оформлення технічної документації проєкту ……... 25.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програма та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ЗАКЛАДІВ ХАРЧУВАННЯ НА ОСНОВІ 3D ВІЗУАЛІЗАЦІЇ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Матвій СУСЛЕНКО

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

PC (Personal Computer) – персональний комп’ютер.

Кастомізація (від англ. customize – налаштовувати) – процес адаптації та налаштування об’єкту або продукту під певні потреби.

Мовна локалізація (від лат. locus – місце) – переклад та культурна адаптація продукту до особливостей регіону, країни чи групи населення

Ре́ндеринг (англ. rendering – візуалізація, проявлення, відмальовування, подання) – в комп'ютерній графіці – це процес отримання зображення за моделлю з допомогою комп'ютерної програми.

Конвеєр візуалізації (англ. Render Pipeline) – послідовність кроків та алгоритмів для рендерингу зображення

HDRP (High Defenition Render Pipeline) – конвеєр візуалізації в Unity, який надає вдосконалені технології RT3D-рендеринга, оптимізовані для PC (Windows, Mac, Linux, VR) та консолей (PlayStation 4/5, Xbox One, Xbox One Series X и S).

URP (Universal Render Pipeline) – конвеєр візуалізації в Unity анонсований у версії 2019.3. Легший за HDRP та більш оптимізований.

IDE (англ. Integrated Development Environment) – середовище розроблення програмного забезпечення.

*HTTP (*Hyper Text Transfer Protocol) – протокол передачі гіпертекстових документів.

ПЗ – Програмне Забезпечення.

ВСТУП

Використання наочної візуалізації їжі в меню ресторанів є дуже важливою складовою успішного гастрономічного бізнесу. Це допомагає клієнтам краще зрозуміти, що саме вони замовляють і як це буде виглядати на їх тарілці.

При використанні наочних зображень страв у меню, клієнти можуть більш точно уявити, як їх страва буде виглядати та які будуть її складові. Це може зменшити ймовірність незадоволення клієнтів, які можуть бути розчаровані, коли страва, яку вони замовили, виглядає інакше, ніж вони очікували.

Також наочна візуалізація їжі може допомогти збільшити продажі певних страв у ресторані. Коли страва виглядає дуже апетитно на зображенні, клієнти можуть бути більш схильні замовляти її.

На сьогодні спостерігається висока конкуренція у сфері харчування. Створюються нові концепції ресторанів, процеси автоматизуються. Деякі з кафе та ресторанів пропонують самому створити страву зі списку інгредієнтів. Це можуть бути канапки, бургери, салати тощо. Така форма замовлення набирає популярність, як у мережах ресторанів, так і в одиничних закладах.

Отже, створення програмного забезпечення, що автоматизує створення замовлення в закладі харчування з використанням наочної візуалізації їжі та можливістю створювати страви зі списку інгредієнтів, є актуальною задачею.

Даний дипломний проєкт присвячено розробленню Програмної платформи для закладів харчування на основі 3D візуалізації. Проєкт призначений для надання можливості створювати та обробляти замовлення у закладі, які складаються, як з готових страв та напоїв, так і зі страв, що зібрані клієнтом кафе або ресторану з окремих інгредієнтів.

1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Аналіз проблем

Проблемною областю, що вимагає покращення та аналізу є потреби користувачів. Відсутність наочної візуалізації страв у меню закладу харчування може створювати проблеми для клієнтів, оскільки в такому випадку вони не мають чіткого уявлення про те, як виглядатиме та з чого складатиметься їхнє замовлення. Це може призвести до незадоволення клієнтів, які отримали не те, що очікували, та втратили довіру до ресторану або кафе.

Крім того, відсутність можливості створювати страву з окремих інгредієнтів при замовленні їжі з ресторану також може призвести до незручностей та отримання небажаного користувацького досвіду. Клієнти з певними дієтами або харчовими обмеженнями можуть наражати себе на небезпеку, замовляючи страви з певними складовими, які їм не підходять або шкідливі для їхнього організму. Це може призвести до того, що люди не будуть звертатись до ресторану, в якому немає зручної можливості кастомізувати страву.

Також неможливість інтегрувати програмну платформу для закладів харчування на вебсайт, у мобільний додаток, використовувати на інтерактивних стендах у кафе, ресторанах тощо, є проблемою, яка потребує вирішення.

1.2. Опис рішень

Під час розроблення даного дипломного проєкту були проаналізовані:

1) алгоритми створення та обробки замовлень у закладах харчування з використанням різних програмних платформ та засобів;

2) процеси, необхідні для зручного функціонування та масштабування роботи закладів харчування;
3) існуючі програмні рішення, їх функціонал, характеристика, переваги та недоліки.

Одне з найкращих рішень у цьому випадку – проаналізувати потреби відвідувачів закладів харчування, задовольнити якомога більше з них для покращення враження про обслуговування в кафе чи ресторані та вдосконалити гастрономічний бізнес у цілому.

Якщо проаналізуємо загальні потреби користувача під час замовлення їжі в закладі харчування, то побачимо наступне:

1) додавати готові страви до замовлення;
2) додавати напої до замовлення;
3) бачити візуально кожну з позицій меню;
4) переглядати замовлення повністю;
5) видаляти додану страву або напій з замовлення;
6) замовляти дистанційно, без необхідності звертатись до офіціанта або бармена;
7) змінювати мову користувацького графічного інтерфейсу.

Виходячи з цього списку потреб, можна впевнено сказати, що вибір програмної платформи для закладів харчування має значення, тому необхідно враховувати кожен з факторів, які впливатимуть на користувацький досвід клієнтів закладу та на обсяг продажів ресторану чи кафе.

Таким чином, задовольняючи якомога більше потреб, надаючи можливість виконувати необхидні задачи із тих, що перераховані вище, можна підвищити зручність та якість сервісу обслуговування.

1.3. Огляд існуючого програмного забезпечення

В дипломному проєкті переглянуті та досліджені можливості найбільш розповсюджених існуючих програмних засобів. Метою

проведеного аналізу є порівняння Програмних платформ для закладів харчування з розміщенням меню для клієнтів.

Виконано порівняння з існуючими аналогами. Був проведений аналіз особливостей кожного з продуктів.

Таблиця 1

Порівняння існуючих аналогів

Програмна платформа	Expirenza by Mono	Resto Market
Популярність за версією google.com	8	6
Візуалізація страв	Лише фото	Лише фото
Дистанційне замовлення	Відсутнє, необхідно кликати офіціанта	Присутнє, але лише у залі закладу харчування
Можливість створювати страви з окремих інгредієнтів	–	–
Мовна локалізація	–	–
Можливість використовувати програму на інтерактивних стендах для замовлення їжі	–	–

1.3.1. *Програмний продукт Expirenza by Mono*

Переваги:

1. Зручний функціонал по роботі з відгуками.
2. Перехід до процесу замовлення за QR-кодом.
3. Можливість розрахуватись за замовлення.
4. Прив’язаність замовлення до столика, за яким перебуває клієнт закладу харчування.
5. Приємний дизайн інтерфейсу користувача.
6. Можливість залишити відгук про обслуговування.

7. Можливість оцінити користувацький досвід чисельно, обираючи оцінку в межах від одного до п’яти.

Недоліки:

1. Відсутність можливості замовити їжу дистанційно. Замовлення приймаються за участю офіціанта або бармена.
2. Відсутність мовної локалізації.
3. Відсутність можливості створити страву з окремих інгредієнтів та/або вилучити інгредієнти з готової страви

1.3.2. *Програмний продукт Resto Market*

Переваги:

1. Можливість створити замовлення без участі офіціанта або бармена.
2. Сучасний дизайн інтерфейсу користувача.

Недоліки:

1. Відсутність мовної локалізації.
2. Відсутність можливості оплатити замовлення.
3. Відсутність можливості створити страву з окремих інгредієнтів та/або вилучити інгредієнти з готової страви.

1.4. Актуальність розробки

В процеси розробки дипломного проєкту були досліджені дві різні програмні платформи для закладів харчування з розміщенням меню для клієнтів:

1. Expirenza by Mono.

2. Resto Market.

Як видно з порівняльного аналізу (табл. 1), найбільше якісних переваг у додатку Expirenza by Mono. Аналіз різноманітних ресурсів мережі Інтернет: вебсторінок, порівняльних таблиць, статей, відкритих баз даних, рекламних платформ, які містять інформацію про програмні

рішення для закладів харчування показав переваги та недоліки двох програмних платформ, Expirenza by Mono та Resto Market, які широко використовуються в Україні та мають велику кількість позитивних відгуків, як від представників, так і безпосередньо від відвідувачів ресторанів, барів та кафе.

Як наслідок проведеного дослідження функціоналу основних програмних рішень за вказаними вище параметрами доходимо до висновку, що жодна з наявних на ринку програмних платформ повністю не задовольняє вимоги та потреби власників закладів харчування та клієнтів ресторанів, барів та кафе. Відсутність мовної локалізації, неможливість використовувати програму на інтерактивних стендах для замовлення їжі, відсутність можливості інтегрувати програмне рішення на власний вебресурс, відсутність опції замовляти страви дистанційно без участі офіціантів та відсутність можливості формувати страви з окремих інгредієнтів створюють помітні незручності для автоматизації гастрономічного бізнесу.

Отже, проблема є актуальною та запропоноване рішення має можливість отримати свою стійку позицію на ринку.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Графічні бібліотеки або ігровий рушій

Для створення додатку на основі 3D візуалізації можна використовувати лише графічні бібліотеки або більш функціональні ігрові рушії.

Графічні бібліотеки, такі як OpenGL та DirectX, зазвичай використовуються для створення додатків, таких як програми для обробки фото чи відео, візуалізації даних тощо. OpenGL (Open Graphics Library – відкрита графічна бібліотека) – специфікація, що визначає незалежно від мови програмування кросплатформний програмний інтерфейс для написання додатків, що використовують двовимірну і тривимірну комп'ютерну графіку [1].

Графічні бібліотеки дозволяють створювати складну графіку з широким спектром ефектів та можливостей. Однак, вони не мають вбудованого фізичного рушія чи іншої логіки, тому для створення інтерактивних додатків вони часто використовуються в поєднанні з іншими інструментами.

Ігрові рушії, такі як Unity та Unreal Engine, спеціально розроблені для створення ігор та додатків. Вони мають вбудовані фізичні рушії, системи колізій, штучного інтелекту та інші функції, що дозволяють створювати складні інтерактивні додатки з великою кількістю об'єктів та ефектів. Крім того, ігрові рушії часто мають вбудовані засоби для розроблення на різних платформах, що полегшує розроблення та розгортання додатків. Саме тому обираємо використання ігрового рушія для легкої та швидкого створення програмного продукту.

2.2. Вибір ігрового рушія

У цьому підрозділі буде розглянуто найпопулярніші ігрові рушії для розроблення програмного забезпечення з використанням тривимірної графіки.

2.2.1. *Огляд ігрового рушія Godot Engine*

Godot Engine – це багатофункціональний кросплатформенний ігровий рушій для створення 2D і 3D ігор та додатків за допомогою єдиного інтерфейсу [2]. Рушій надає повний набір загальних інструментів, які дозволяють зосередитися на створенні продукту без потреби реалізовувати базові алгоритми, компонентну систему, витрачати багато часу на налаштування процеси візуалізації.

Ігровий ігровий Godot розроблений нещодавно, порівнючи з іншими, наведеними. Його перший вихід у світ був у лютому 2014 року. На момент релізу руший мав мінімальний необхідний набор функціоналу без перевантаженої системи вкладок і налаштувань. За цю простоту він швидко став основним інструментом для розробки ігор та інтерактивних додатків багатьма молодими розробниками. На сьогоднішній день Godot Engine поступово переростає з формату аматорського ігрового рушія до формату повноцінного потужного рушія для розробки як 2Д, так і 3Д проектів.

Сам рушій написаний з використанням мов програмування C і C++.

Розглянемо переваги ігрового рушія Godot Engine:

1. Кросплатформеність. Додатки, створені за допомогою Godot Engine, можна легко експортувати на низку платформ:
   a. ПК: Godot engine підтримує розроблення ігор для різних операційних систем, таких як Windows, macOS та Linux.
   b. Консолі: Godot engine підтримує гральні консолі, такі як PlayStation, Xbox та Nintendo Switch.

c. Мобільні пристрої: Godot engine має підтримку для мобільних платформ, таких як Android та iOS. Godot engine має підтримку для мобільних процесорів та графічних карт, що дозволяє створювати більш складні ігри та додатки.
d. VR та AR: Godot engine має підтримку для віртуальної та доповненої реальності, включаючи платформи, такі як Oculus та Google Cardboard.
e. Web: Godot engine можна використовувати для розроблення веб-ігор та додатків з використанням технологій, таких як HTML5 та WebAssembly.
f. Інші платформи: Godot engine також має підтримку для інших платформ, таких як FreeBSD, OpenBSD, Haiku та Nintendo 3DS.

2. Доступність. Godot є повністю безкоштовним і з відкритим вихідним кодом під ліцензією MIT. Дозвільна ліцензія MIT надає розробникам можливість використовувати Godot Engine для будь-яких цілей. Godot не вимагає платежів за використання. Створений додаток юридично повністю належить розробнику. Ігровий рушій підтримується некомерційною організацією Software Freedom Conservancy.
3. Сценічний підхід. Однією із корисних особливостей Godot є те, що він використовує сцени, як елемент структури додатку. Це означає, що все у вашій грі або додатку організовано в сцени, і кожну сцену можна завантажувати та вивантажувати за потреби. Це допомагає підтримувати низький рівень використання пам’яті гри, що, у свою чергу, сприяє її плавній роботі.
4. Однак те, де цей ігровий движок не відповідає ставкам у продуктивності Godot vs Unity, — це його можливості рендерингу. Механізм візуалізації Unity просто потужніший і може працювати зі складнішою графікою. Отже, якщо ви хочете створити візуально приголомшливу гру, Unity — кращий вибір.

5. Легкість розробки для новачків. Інтерфейс користувача на основі вузлів полегшує новачкам програмування, що робить Godot Engine чудовим інструментом для створення простих ігор та додатків. Однак якщо є необхідність створювати складніші проєкти, програмування за допомогою графічних вузлів викликатиме суттєві обмеження, такі як неможливість легкого масштабування, створення комплексних алгоритмів.

Розглянемо недоліки ігрового рушія Godot Engine:

1. Низька якість графіки за стандартних налаштувань. Відображення графіки у Godot поступається Unity та деяким іншим рушіям та технологіям, а доступні інструменти затінення не такі просунуті. Звичайно, це не означає, що не можливо створювати реалістичні сцени в Godot. Це означає, що на це доводиться докладати значно більше зусиль.

2.2.2. *Огляд ігрового рушія Unreal Engine*

Unreal Engine – ігровий рушій, що розробляється та підтримується компанією Epic Games [3], відкрита розвинена платформа для створення 3D контенту у реальному часі. Постійно вдосконалючись як один з найсучасніших ігрових рушіїв, він дає творцям у різних галузях свободу та контроль для надання передового вмісту, інтерактивного досвіду та захоплюючих віртуальних світів.

Для спрощення портування рушій використовує модульну систему залежних компонентів; підтримує різні системи рендерингу (Direct3D, OpenGL, Pixomatic; у ранніх версіях: Glide, S3, PowerVR), відтворення звуку (EAX, OpenAL, DirectSound3D; раніше: A3D), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримки різних пристроїв введення.

Розглянемо переваги ігрового рушія Unreal Engine:

1. Кросплатформність. Unreal Engine дозволяє розробникам створювати ігри для різних платформ, включаючи ПК, консолі, мобільні та віртуальні реальності. Повний перелік доступних платформ:
    a. ПК: Unreal Engine може бути використаний для створення ігор на різних операційних системах, таких як Windows, macOS та Linux.
    b. Консолі: Unreal Engine підтримує такі гральні консолі, як PlayStation, Xbox та Nintendo Switch.
    c. Мобільні пристрої: Unreal Engine підтримує розроблення ігор для мобільних платформ, таких як iOS та Android. Unreal Engine має підтримку для мобільних процесорів та графічних карт, що дозволяє створювати більш складні ігри та додатки.
    d. VR та AR: Unreal Engine є популярним рушієм для розроблення віртуальної та доповненої реальності. Unreal Engine має підтримку для різних гарнітур віртуальної реальності та AR-платформ.
    e. Web: Unreal Engine можна використовувати для розроблення веб-ігор та додатків з використанням технологій, таких як WebAssembly та WebGL.
    f. Інші платформи: Unreal Engine також має підтримку для інших платформ, таких як TVOS та Linux for Tegra.
2. Велика та активна спільнота: Unreal Engine має розвинену спільноту розробників, яка допомагає новачкам та надає підтримку досвідченим розробникам.
3. Графіка. Unreal Engine володіє потужними інструментами для створення детальної графіки, включаючи високоякісні ефекти освітлення, текстури, воду, тіні, та багато іншого.
4. Багатофункціональність. Рушій дозволяє створювати різні типи ігор - від шутерів до головоломок та інших жанрів; має вбудовані

інструменти для розроблення штучного інтелекту, фізики, анімації та інших функцій.

5. Мультиплеєр. Unreal Engine має вбудовані інструменти для розроблення мультиплеєрних ігор та підтримує різні мережеві технології.
6. Відкритість коду. Unreal Engine є відкритим джерелом, що означає, що розробники можуть взяти код движка та змінити його за потребою.
7. Розглянемо недоліки ігрового рушія Unreal Engine:
8. Високі вимоги до системи. Unreal Engine потребує потужного комп'ютера для ефективної роботи, що може бути проблемою для новачків або розробників з обмеженим бюджетом.
9. Складність. Unreal Engine є потужним ігровим движком, тому він може бути складним у використанні, особливо для новачків. Він має велику кількість інструментів та функцій, що може призвести до того, що розробник витрачає багато часу на навчання і розуміння різних аспектів движка.
10. Вартість. Unreal Engine є комерційним продуктом і має високу вартість для комерційних проєктів, що може стати проблемою для невеликих студій або незалежних розробників.
11. Проблеми з оптимізацією. Оскільки Unreal Engine має велику кількість функцій та можливостей, деякі розробники можуть стикатися з проблемами оптимізації, особливо на мобільних пристроях.
12. Великий розмір файлів. Готові проєкти, створені з використанням Unreal Engine, можуть мати великий розмір файлів, що може призвести до проблем зі збереженням, передачею та завантаженням проєктів.

2.2.3. *Огляд ігрового рушія Unity*

Unity (у перекладі з англ. «єдність», вимовляється як «юніті») – кросплатформове середовище розроблення комп'ютерних ігор та інтерактивних додатків, розроблене американською компанією Unity Technologies [4]. Unity почав своє життя як ігровий рушій, але перетворився на творчий інструмент, який використовується у багатьох різних галузях.

В основі Unity лежить компонентно-орієнтована концепція. Будь-яка гра або додаток, створений з використанням ігрового рушія Unity, складається з великої кількості об'єктів із додаванням окремих компонентів, що реалізовують логіку. Для прикладу, під час створення гри-платформера, створюється об’єкт GameObject, до нього додатково прикріплюється графічна складову, що відповідає за відображення персонажа, і керуючий компонент, який забезпечує управління персонажем за рахунок миші, клавіатури, джойстика або тачскрин. Ігровий руший не накладає обмеження на кількість таких модулів. До GameObject можна додати стільки компонентів, скільки цього потребує розробка. Вся робота на рушії будується на тому самому створенні GameObject і застосування до них відповідних компонентів.

Розглянемо переваги ігрового рушія Unity:

1. Легкість використання. Unity має інтуїтивно зрозумілий інтерфейс, що дозволяє новачкам швидко розпочати розроблення. Він має широкий спектр інструментів та бібліотек, що полегшує створення ігор.
2. Мультиплатформеність. Unity підтримує різні платформи.Повний перелік доступних платформ:
    a. ПК: Unity підтримує розроблення ігор для різних операційних систем, таких як Windows, macOS та Linux.
    b. Консолі: Unity підтримує гральні консолі, такі як PlayStation, Xbox та Nintendo Switch.

c. Мобільні пристрої: Unity є популярним рушієм для розроблення ігор та додатків для мобільних платформ, таких як iOS та Android. Unity має підтримку для мобільних процесорів та графічних карт, що дозволяє створювати більш складні ігри та додатки.
d. VR та AR: Unity є популярним рушієм для розроблення віртуальної та доповненої реальності. Unity має підтримку для різних гарнітур віртуальної реальності та AR-платформ.
e. Web: Unity можна використовувати для створення веб-ігор та додатків з використанням технологій, таких як WebGL.
f. Інші платформи: Unity також має підтримку для інших платформ, таких як TVOS.

3. Велика спільнота. Unity має велику спільноту розробників, що дозволяє отримати допомогу та підтримку в розвитку проєкту. У спільноті також є багато безкоштовних ресурсів, які можна використовувати для навчання та підвищення кваліфікації.
4. Розширюваність. Unity дозволяє додавати різні розширення та плагіни для збільшення функціональності движка.
5. Розвинений функціонал для візуалізції. Unity має потужні можливості візуалізації, що дозволяє створювати реалістичні та деталізовані графічні ефекти.

Розглянемо недоліки ігрового рушія Unity:

1. Обмежена швидкість. Unity не так швидкий, як деякі інші рушії, хоча це може стати проблемою лише для проєктів з великою кількістю об'єктів та ефектів.
2. Обмежена контрольованість. Unity має менше контролю над рівнем оптимізації та розміру вихідного коду порівняно з іншими рішеннями.

Таблиця 2

Порівняння ігрових рушіїв

Ігровий рушій	Godot Engine	Unreal Engine	Unity
Розробник	Співавторство Godot Engine Community	Epic Games	Unity Technologies
Доступні плафторми розміщення (портування) додатків	Windows, macOS, Linux, PlayStation, Xbox, Nintendo Switch, Android, iOS, VR та AR, Web (HTML5 та WebAssembly), FreeBSD, OpenBSD, Haiku та Nintendo 3DS.	Windows, macOS, Linux, PlayStation, Xbox, Nintendo Switch, iOS, Android. , VR та AR, Web (WebAssembly та WebGL) TVOS.	Windows, macOS, Linux, PlayStation, Xbox, Nintendo Switch, iOS, Android. , VR та AR, Web (WebAssembly та WebGL) TVOS.
Підтримка 3D графіки	Так	Так	Так
Мова програмування ігрових скриптів	GDScript, C#, та C, C++ з використанням GDExtension technology	C++	C#
Мова написання шейдерів	Аналогічна до GLSL ES 3.0	HLSL	HLSL
Програмування вузлами	Доступне	Доступне	Доступне (Unity Visual Scripting)
Графіка за замовчуванням	На невисокому рівні	Дуже реалістична	Реалістична

2.3. Вибір конвеєра візуалізації

У цьому підрозділі буде розглянуто два можливих конвеєра візуалізації, які надає розробникам рушій Unity. High Definition Render Pipeline (HDRP) та Universal Render Pipeline (URP) – це два різних

рендерингових рішення від Unity, кожне з яких має свої особливості та переваги.

2.3.1. *Огляд конвеєра візуалізації High Definition Render Pipeline*

HDRP – це потужна система рендерингу, яка забезпечує високу якість візуалізації з деталізацією високої якості, реалістичною обробкою світла та високою продуктивністю [5]. HDRP має підтримку різних платформ та API, таких як DirectX 11 та 12, Vulkan, Metal, OpenGL Core та інші. HDRP дозволяє створювати складні різноманітні ефекти, такі як реалістична вода, туман, дим, світлові ефекти, що роблять гру чи додаток більш живим і реалістичним.

2.3.2. *Огляд конвеєра візуалізації Universal Render Pipeline*

Універсальний конвеєр візуалізації (URP) – це конвеєр візуалізації, створений Unity. URP забезпечує робочі процеси, зручні для художників, які дозволяють швидко та легко створювати оптимізовану графіку на різних платформах, від мобільних до консолей високого класу та ПК [6]. URP – це легша версія HDRP, яка забезпечує високу продуктивність при створенні ігор та додатків з більш простими графічними ефектами. URP підтримує ті ж платформи та API, що й HDRP, але з меншою кількістю можливостей та функцій. URP є оптимізованим для мобільних пристроїв та планшетів з низькою потужністю та може знижувати витрати на ресурси збільшуючи продуктивність в багатьох ситуаціях.

2.4. Вибір середовища розроблення програмного забезпечення

Розглянемо Visual Studio та Rider, два популярні інтегровані середовища розроблення (IDE), які можна використовувати для створення ігор та додатків на Unity.

Visual Studio є найбільш популярним IDE для розроблення на Unity, що надає зручний та простий інтерфейс, підтримку автодоповнення та

відладку, першокласний досвід налагодження ігрового рушія Unity [7]. Visual Studio також має деякі можливості підключення додаткових інструментів, таких як ReSharper, що дозволяє покращити продуктивність розроблення.

Rider – це інша популярна IDE від компанії JetBrains, яка має підтримку для розроблення на Unity. Rider надає більше функцій, що дозволяють покращити продуктивність розроблення, такі як автодоповнення, швидкий пошук та відлагодження, значну кількість корисних модулів для тестування та рефакторингу коду. Також Rider пропонує інтеграцію з додатковими інструментами, такими як ReSharper та dotTrace, що робить його відмінним вибором для роботи з Unity. Rider підтримує .NET Framework, нову платформу .NET Core та проекти на основі Mono. IDE дозволяє розробляти десктопні програми, .NET-сервіси та бібліотеки, ігри на движку Unity, мобільні програми Xamarin, веб-додатки ASP.NET та ASP.NET Core. Rider надає понад 2200 інспекцій коду, сотні контекстних дій та рефакторингів, запозичених із ReSharper, у поєднанні з просунутою функціональністю середовищ розробки на основі платформи IntelliJ. Незважаючи на великий набір функцій, Rider – надзвичайно швидке середовище розроблення програмного забезпечення.

2.5. Обрані інструменти

У даному розділі розглянуто перелік інструментів, які можна використовувати для створення Програмної платформи для закладів харчування на основі 3D візуалізації.

Для розроблення клієнтської частини програмного забезпечення використаємо ігровий рушій Unity, який є легким для розроблення, має зручний інтерфейс, надає реалістичну картинку навіть із параметрами за замовчуванням, підтримує велику кількість платформ, має потужну спільноту розробників, розгорнуті інструкції та довідники, багату базу допоміжної літератури.

З конвеєрів візуалізації Unity зупиняємось на Universal Render Pipeline, який є оптимальним рішенням для швидкої роботи додатку на багатьох платформах, у тому числі на мобільних.

Сервер створимо, використовуючи мову C#, яка є основною мовою розроблення на Unity.

За середовище розроблення програмного забезпечення було обрано Rider, як найбільш зручне та сучасне інтегроване середовище розроблення у поєднанні з ігровим рушієм Unity.

3. РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА ПРОГРАМНИХ МОДУЛІВ

3.1. Загальний опис архітектури

Використаємо клієнт-серверну архітектуру.

Клієнт-серверна архітектура – це підхід розподілення додатків. Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером [8]. Клієнт-серверна архітектура складається з наступних компонентів:

- клієнт, який користується сервісами, що надаються сервером;
- сервер, який обслуговує клієнтів, надає їм інформацію;
- мережа – через неї проходить взаємодія сервера та клієнта.

Клієнт та сервер відокремлені, незалежні одна від одної та розміщені на окремому апаратному забезпеченні, проте для функціонування вони мають залишаться в рамках однієї системи. На стороні сервера може працювати одна чи більше програм, які надають свої ресурси клієнтам, що до них звертаються. Клієнт не надає свої ресурси, але він робить запити на сервер для того, щоб отримати необхідну для нього інформацію. Клієнт є стороною-ініціатором спілкування з сервером, а сервер очікує на вхідні запити. Клієнт може бути будь-яким пристроєм, під’єднаним до мережі інтернет. На рис. 1 наведена клієнт-серверна архітектура.

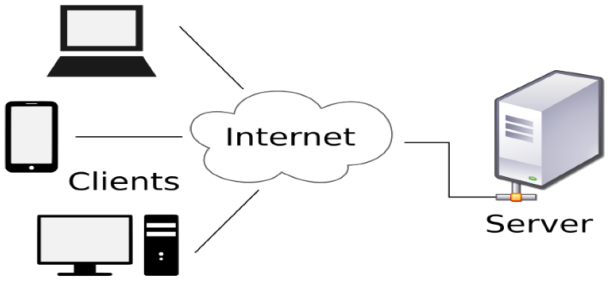


Рис. 1. Клієнт-серверна архітектура

На стороні сервера будемо використовувати клас HttpListener, використаємо чергу для запитів, щоб почергово їх обробляти (рис. 2).

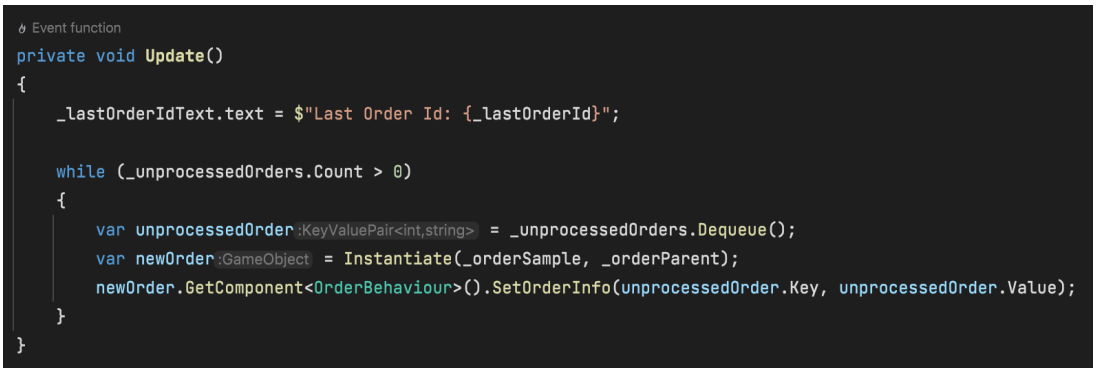


Рис. 3. Обробка запитів-замовлень з черги на стороні сервера.

На стороні клієнта взаємодію з сервером будуємо на основі класу HttpClient з використання протоколу HTTP (рис. 3).

Завдання, яке традиційно вирішується за допомогою протоколу HTTP – обмін даними між додатком користувача, що здійснює доступ до веб-ресурсів і веб-сервером.

Використовуватимемо метод POST з можливістю передавати тіло в запиті. Метод є послідовністю з будь-яких символів, крім керуючих і роздільників, і визначає операцію, яку потрібно здійснити із зазначеним ресурсом. Специфікація HTTP 1.1 не обмежує кількість різних методів, які можуть бути використані, проте з метою відповідності загальним стандартам та збереження сумісності з максимально широким спектром програмного забезпечення зазвичай використовуються лише деякі, найбільш стандартні методи, зміст яких однозначно розкритий у специфікації протоколу.

В тілі запиту будемо передавати текст спеціального формату із вмістом замовлення: обраними напоями, додатковими стравами, сформованими власноруч клієнтом закладу харчування стравами з наданого переліку інгредієнтів. Також при обранни формату тіла запіта будемо враховувати потенційне розширення функіоналу додатку. У відповідь від сервера будемо отримувати унікальний ідентифікатор замовлення.

```
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri(Address + Endpoint),
    Content = new StringContent(body, Encoding.UTF8, mediaType: "application/json"),
};

try
{
    var response = await _httpClient.SendAsync(request).ConfigureAwait(true);
    var responseBody :string = await response.Content.ReadAsStringAsync().ConfigureAwait(true);
    NewOrderReceived?.Invoke( obj: int.Parse(responseBody));
}
catch (Exception e)
{
    Debug.LogWarning(e.Message);
}
```

Рис. 3. Формування та оброблення запитів на стороні клієнта.

Також в процесі написання коду будемо базуватись на особливостях життєвого циклу об’єкта MonoBehaviour як основного батьківського класу для інших похідних класів у додатку.

Під час використання та написання шейдерів, під час роботи з текстурами, матеріалами та іншими графічними об’єктами, будемо враховувати загальний принцип роботи шейдерів. Також в проєкті наведена діаграма варіантів використання.

3.2. Опис модулів програмного забезпечення

Структура програмного забезпечення складається з наступних модулів:

- модуль обробки та зберігання інформації про замовлення;
- модуль створення канапки з інгредієнтів;
- модуль доступних напоїв;
- модуль доступних додаткових готових страв;
- модуль клієнт-серверної комунікації;
- модуль мовної локалізації.

3.2.1. *Модуль обробки та зберігання інформації про замовлення*

Розглянемо основні сутності та процеси, які використовуються під час роботи із замовленнями у закладі харчування.

Сутність інформації про страву (рис. 4) має наступні атрибути:

- ingredients – масив інгредієнтів;
- description – детальний опис страви всіма доступними мовами;
- name – назва страви всіма доступними мовами;
- calories – кількість калорій;
- price – ціна в гривнях.

```
public interface IDishInfo
{
    1 usage  2 implementations
    string[] Ingredients { get; }
    2 implementations
    TranslatableName Description { get; }
    5 usages  2 implementations
    TranslatableName Name { get; }
    2 implementations
    float Calories { get; }
    3 usages  2 implementations
    int Price { get; }
}
```

Рис. 4. Сутність інформації про страву

Сутність інформації про страву є полем в сутності Страви, яка окрім того прив’язана до тривимірного об'єкту (ресурсу), який візуалізує страву.

Для страв, зібраних клієнтом закладу харчування власноруч, вводимо сутність Інгредієнту (рис. 5), що складається з полів:

- ingreditent asset – ресурс (тривимірний об’єкт);
- ingredient sprite – зображення інгредієнту для меню інгредієнтів;
- name – назва інгредієнту усіма доступними мовами;

- calories – кількість калорій;
- weight – вага в грамах;
- price – ціна в гривнях.

```
[Serializable]
1 usage  1 inheritor
public class Ingredient
{
    1 usage
    public GameObject IngredientAsset => _ingredientAsset;
    1 usage
    public Sprite IngredientSprite => _ingredientSprite;
    4 usages
    public TranslatableName Name => _name;
    1 usage
    public float Calories => _calories;
    1 usage
    public float Weight => _weight;
    1 usage
    public int Price => _price;

    [SerializeField] private GameObject _ingredientAsset;  Serializable
    [SerializeField] private Sprite _ingredientSprite;  Serializable
    [SerializeField] private TranslatableName _name;  Serializable
    [SerializeField] private float _calories = 5;  Serializable
    [SerializeField] private float _weight = 20;  Serializable
    [SerializeField] private int _price = 10;  Serializable
}
```

Рис. 5. Сутність інгредієнту

На початку використання програмного модуля клієнт закладу харчування має можливість створити нове замовлення. У такому випадку локальна модель замовлення очищується. Далі користувач потрапляє у головне меню, в якому може скасувати замовлення і повернутись в початковий стан програми; підтвердити замовлення при наявності хоча б однієї страви; додати страву до замовлення.

3.2.2. *Модуль створення канапки з інгредієнтів*

Розширимо сутність інгредієнту для випадку з інгредієнтами канапки, утворивши Сутність інгредієнту канапки (рис. 6), яка матиме додаткове поле: height – висота (товщина) інгредієнту, яка використовується для вертикального зміщення наступних інгредієнтів, що потрібно для тривимірної візуалізації канапки

```
[Serializable]
18 usages
public class SandwichIngredient : Ingredient
{
    2 usages
    public float Height => _height;

    [SerializeField] private float _height = 0.01f;
}
```

Рис. 6. Сутність інгредієнту канапки

3.2.3. *Модуль клієнт-серверної комунікації*

Клієнт-серверна комунікація відбуватиметься з використанням протоколу передачі гіпертекстових документів (Hyper Text Transfer Protocol) . Клієнт надсилатиме запит зі сформованим замовленням та отримуватиме унікальне число-ідентифікатор замовлення, яке буде показуватись на екрані відвідувачу/користувачу закладу харчування. Детальніше особливості реалізації клієнт-серверної комунікації наведені вище в описі архітектури проєкту.

3.2.4. *Модуль мовної локалізації*

Щоб адаптувати продукт для інших країн якісно, необхідний переклад та локалізація – створення мовних версій з урахуванням

культурних особливостей іншої країни. На вересень 2017 року кількість додатків у Google Play становила близько 3 млн. Широкий спектр локалізованих версій давав можливість купувати програми користувачам майже з 150 країн. Мовна локалізація є невід'ємною частиною сучасного програмного забезпечення з графічним інтерфейсом користувача.

Зміна мови інтерфейсу відбувається динамічно на клієнтській стороні. Для цього з-поміж інших використовується Сутність Translatable Name (назва, що перекладається, рис. 7), яка має наступні атрибути:

- ukrainian name – назва українською мовою;
- english name – назва англійською мовою.

```
[System.Serializable]
14 usages  11 exposing APIs
public class TranslatableName
{
    2 usages
    public TranslatableName(string ukrainianName, string englishName)
    {
        _ukrainianName = ukrainianName;
        _englishName = englishName;
    }

    public string _ukrainianName;  Serializable
    public string _englishName;  Serializable
}
```

Рис. 7. Сутність Translatable Name (назва, що перекладається)

Також створимо Сутність Locale Label (рис. 8), яка дозволятиме вказувати у вікнах редактора Unity переклад для текстових елементів графічного інтерфейсу користувача (рис. 9), що дасть змогу легко і швидко додавати переклад для створених елементів меню та програмного модуля загалом. Сутність реагуватиме на обрання мови інтерфейсу та автоматично буде змінювати значення відповідного текстового поля.

```
[RequireComponent(typeof(Text))]
21 asset usages
public class LocaleLabel : MonoBehaviour
{
    [SerializeField] private string _ukrainianVariant;  Unchanged
    [SerializeField] private string _englishVariant;  Changed in 1 asset
    private Text _textComponent;

    Event function
    private void Start(){...}

    3 usages
    private void OnLanguageSelected(Language language)
    {
        switch (language)
        {
            case Language.Ukrainian :
                _textComponent.text = _ukrainianVariant;
                break;
           case Language.English :
               _textComponent.text = _englishVariant;
               break;
        }
    }

    Event function
    private void OnDestroy(){...}
}
```

Рис. 8. Сутність Locale Label

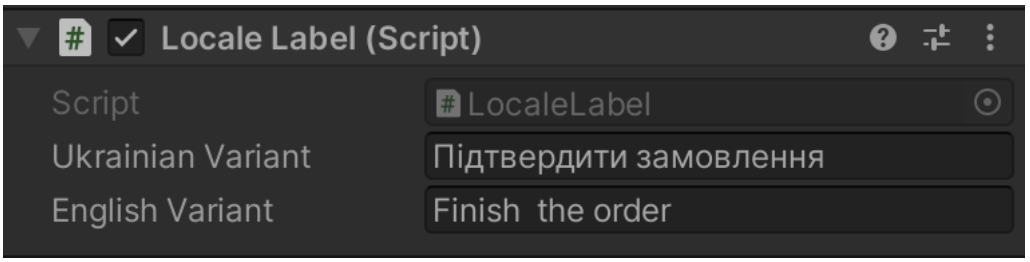


Рис. 9. Зазначення перекладу у вікнах редактору Unity

3.3. Шаблон розробки інтерфейсу користувача

Розроблення клієнтської частини програмного забезпечення спиратиметься на використання шаблону проєктування MVP.

MVP (Model-View-Presenter) – патерн розробки інтерфейсу користувача. Шаблон MVP є похідним від MVC, але має дещо інший підхід. Основна відмінність – presenter менш обмежений моделлю (model).

Шаблон MVP обрано в результаті його зручного практичного використання та порівняння зручності розробки додатків з його використанням у порівнянні з іншими шаблонами, наведеними нижче.

MVC (Model-View-Controller) – схема поділу даних програми та керуючої логіки на три окремі компоненти: модель, уявлення та контролер. Таким чином, модифікація кожного компонента може здійснюватися незалежно.

Разом з попередніми існує шаблон проєктування MVVM. Паттерн MVVM, що розшифровується як Model-View-ViewModel, також дозволяє відокремити логіку програми від візуальної частини (подання). Цей патерн був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в різних технологіях, у тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну. MVVM складається з трьох компонентів: моделі (Model), моделі уявлення (ViewModel) та уявлення (View).

3.4. Ін'єкція залежностей

Головна проблема великих додатків у тому, що зі зростанням обсягу коду стає набагато складніше в ньому розібратися, легше припуститися помилки і, як наслідок, розробка сповільнюється. Для боротьби з цим використовується поділ відповідальності («розділяй і володарюй») – ми розбиваємо складне завдання (наприклад, завдання реєстрації користувачів) на невеликі, незалежні частини: виведення форми реєстрації, прийом даних із форми, перевірка інформації, збереження її в базу даних,

авторизація користувача. Кожна з цих підзадач відповідає окремий метод чи клас.

Принцип єдиного обов'язку у тому, кожен клас займається лише своєю справою. Наприклад, один клас відповідає лише за взаємодію з БД, інший за перевірку правильності введених даних, третій за виставлення та перевірку кук для авторизованих користувачів.

Тепер, якщо ми хочемо зрозуміти, як відбувається, наприклад, перевірка даних при реєстрації, або щось у ній поміняти, нам треба вивчити лише один клас, який цим займається. Також ми можемо протестувати процес перевірки окремо від решти коду (більше того, ми можемо його тестувати навіть якщо інші методи поки не написані).

Якщо ми розбили код на класи, то може виявитися, що одному з них потрібен інший.

Наприклад, якщо у нас є клас авторизації, а в ньому метод, що перевіряє логін та пароль користувача на правильність, які зберігаються в базі даних, йому знадобиться клас, який дістає їх звідти.

У таких випадках говорять, що клас А залежить від класу В або В є залежністю (dependency) класу А.

Залежно бувають обов'язкові (без яких клас працювати не може) і необов'язкові. Приклад необов'язкової залежності – це клас-логгер. Він може бути потрібний лише при налагодженні коду, а на бойовому сервері не потрібно.

Тепер обговоримо, в чому шкода від глобальних змінних та зловживання статичними методами, щоб у тебе не виникло навіть думки використовувати їх для зв'язку класів між собою. А потім дізнаємося про правильний метод застосування залежностей.

Глобальна змінна – це змінна, яка завжди існує в одному екземплярі та доступна з будь-якого місця коду. Статичні поля класів теж існують в одному екземплярі, тому їх можна розглядати як різновид глобальної змінної.

На перший погляд вони добре підходять для зберігання таких речей, як: налаштування (наприклад ім'я та пароль до бази даних), або мова повідомлень на багатомовному сайті, але якщо придивитися, то це майже завжди погане рішення.

Те, що глобальна змінна доступна з будь-якого місця коду – призведе для зловживання цією можливістю, погіршення загального ривня якості розробленої архітектури проєкту. Це означає, що при зміні коду, пов'язаного з цією змінною нам доведеться робити пошук по всьому коду і вивчати кожен випадок використання. Наприклад, ми хочемо зрозуміти, звідки береться значення в глобальній змінній, і виявили, що воно задається в різних файлах.

Як зрозуміти, чому ж ця змінна дорівнює нашому випадку? Швидше за все, доведеться розбирати код у цих файлах, дивитися звідки він викликається загалом витрачати багато часу.

Чим менше область, де доступна змінна, тим простіше поміняти код, що працює з нею. У випадку з налаштуваннями бази даних достатньо передати їх тільки в клас, який відповідає за з'єднання з нею.

Також, доступна глобально змінна (наприклад, налаштування бази даних) провокує програмістів писати код роботи з базою даних у будь-якому місці програми, замість сконцентрувати його в одному класі.

Нижче наведемо недоліки класів, в яких всі реалізовані методи статичні (порівняно з класами із звичайних методів, де треба спочатку створити об'єкт і лише потім викликати у нього методи):

1. Так як властивості у таких класів теж статичні, то не можна створити другий об'єкт з дещо іншими значеннями властивостей. Наприклад, не можна створити 2 класи-валідатора, які видають повідомлення про помилку різними мовами, не можна створити об'єкт, що працює з іншими налаштуваннями бази даних. Не можна створити тимчасовий об'єкт, користуватися та викинути його, не вплинувши на решту коду.

2. Статичні методи не можуть звертатися до this, і можуть зберігати дані тільки в статичних полях, які по суті аналогічні глобальним змінним, і їх зміна впливає на весь інший код.
3. Статичний метод можна викликати з будь-якого місця коду. А ось у разі використання звичайних методів їх можна викликати тільки там, куди передано об'єкт. Це дозволяє обмежити зону його використання та натякнути, що працювати з об'єктом потрібно лише тут.
4. Зв'язки між класами жорстко прописані у коді. Коли один клас А викликає статичний метод іншого класу В, ми можемо якось вказати класу А використовувати клас С замість В.
5. Зв'язки між класами не очевидні. Щоб зрозуміти, що клас А використовує клас В, треба вивчити його код повністю. Ми не можемо визначити це тільки за конструктором або заголовками методів.

Щоб створити гнучку, зручну для розробки архітектуру для встановлення та надання залежностей між об’єктами використаємо ін’єкцію залежностей.

Ін’єкція залежностей дає нам максимальну гнучкість використання і дозволяє зробити класи слабо пов'язаними один з одним, так що зміна в одному не вимагатиме переробки іншого. Ін’єкція залежностей – це стиль налаштування об'єкта, коли поля об'єкта задаються зовнішньою сутністю. Інакше кажучи, об'єкти налаштовуються зовнішніми об'єктами. DI – це альтернатива самоналаштування об'єктів.

Скористаємось Zenject, контейнер ін’єкція залежностей з відкритим вихідним кодом, орієнтований на використання з ігровим рушієм Unity3D, що забезпечує роботу на більшості платформ, підтримуваних Unity3D.

Контейнер DI – це програмна бібліотека, яка може автоматизувати багато завдань, які виконуються під час компонування об'єктів та керування їх життєвим циклом.

Коли розробник конфігурує DI-контейнер, вин визначає, екземпляри яких компонентів контейнер може бути здатний створити, і які залежності впровадити в кожен компонент. Програміст також може налаштувати режим створення екземпляра для кожного компонента. Наприклад, чи має новий екземпляр створюватися щоразу? Або той самий екземпляр компонента повинен бути перевикористаний (синглтон) скрізь, куди він впроваджується?

Якщо деякі компоненти налаштовані як синглтони, деякі контейнери мають можливість викликати методи синглтону тоді, коли контейнер вимикається. Таким чином, синглтон може звільнити будь-які ресурси, які він використовує, такі як підключення до БД або мережеве з'єднання. Це зазвичай називають "управлінням життєвим циклом об'єкта". Це означає, що контейнер здатний керувати компонентом різних стадіях життєвого циклу компонента. Наприклад, створення, конфігурування та видалення.

Управління життєвим циклом – це один з обов'язків, який виконують контейнери ін’єкції залежностей. Той факт, що контейнер іноді зберігає посилання на компоненти після створення екземпляра, і є причиною, через яку він називається «контейнером», а не фабрикою.

DI-контейнери зазвичай зберігають посилання на об'єкти, чиїм життєвим циклом вони мають керувати або які будуть перевикористані для майбутніх впроваджень, такі як синглтон або пристосуванець.

Коли контейнер налаштований на створення нових екземплярів компонентів під час кожного виклику, контейнер зазвичай «забуває» про створені об'єкти. В іншому випадку у збирач сміття (Garbage Collector) буде небажано сильно навантаженим, коли настане час збирати всі ці об'єкти.

Збирання сміття (GC) — це функція відновлення пам’яті, вбудована в такі мови програмування, як C# і Java. Мова програмування з підтримкою GC включає один або кілька збирачів сміття (моделі GC), які автоматично звільняють простір пам’яті, який було виділено для об’єктів, які більше не

потрібні програмі. Звільнений простір пам’яті потім можна використовувати для майбутніх розподілів об’єктів у цій програмі.

Збирання сміття гарантує, що програма не перевищить свою квоту пам’яті або не досягне точки, коли вона більше не зможе працювати. Це також звільняє розробників від необхідності вручну керувати пам’яттю програми, що, у свою чергу, зменшує ймовірність помилок, пов’язаних із пам’яттю.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості тестування програмного забезпечення

У цьому розділі проаналізовано методи та підходи тестування програмного продукту.

4.1.1. Методи тестування програмного забезпечення

Тестування програмного забезпечення – це дослідження ПЗ з метою отримання інформації про якість продукту [9], процес звірення відповідності заявленого та дійсного функціоналу програми методом спостереження за процесами під час штучно утворених ситуацій на чітко визначеній групі тестів.

Під час тестування оцінюється:

- перевірка очікуваних функцій;
- очікувана відповідь для можливих вхідних наборів даних;
- витрата часу на досліджувані процеси;
- практичність;
- сумісність із ПЗ та заявленими ОС;
- перевірка графічного інтерфейсу користувача;

4.1.2. Рівні тестування

Під час тестування важливо зважати на рівні тестування. Наведемо деякі з існуючих:

- компонентне тестування (Component\module\unit testing);
- інтеграційне тестування (Integration testing);
- системне тестування (System testing);
- приймальне тестування (Acceptance testing);
- приймальне тестування користувача (User acceptance testing);
- експлуатаційне приймальне тестування (Operational acceptance testing);

- альфа-тестування (Alpha testing);
- бета-тестування (Beta testing).

4.1.3. Види тестування

Поняття виду тестування переслідує одну з наведених цілей:

- оцінити функціональні характеристики якості ПЗ: повнота, правильність, доцільність;
- оцінити нефункціональні характеристики якості ПЗ: надійність, продуктивність, безпека, сумісність, зручність тощо;
- оцінити структуру та архітектуру ПЗ;
- оцінити наслідки змін.

4.1.4. Види функціонального тестування

Наведемо види функціонального тестування:

- тестування функціональної коректності (Functional correctness testing);
- тестування функціональної придатності\доцільності (Functional appropriateness testing);
- тестування функціональної повноти (Function completeness testing).

4.1.5. Види нефункціонального тестування

Наведемо види нефункціонального тестування:

- тестування захищеності (Security testing);
- тестування безпеки (Safety testing);
- тестування продуктивності (Performance testing);
- навантажувальне тестування (Load testing);
- стресове тестування (Stress testing);
- тестування графічного інтерфейсу (GUI testing);
- тестування зручності використання (Usability testing);

- тестування доступності (Accessibility testing);
- тестування сумісності (Compatibility testing);
- тестування кросбраузерності (Cross browser testing);
- тестування кросплатформенності (Cross platform testing);
- тестування інсталяції (Installation testing);
- тестування інтернаціоналізації (Internationalization testing);
- тестування локалізації (Localization testing).

4.1.6. Димове тестування

Димове тестування ПЗ означає мінімальній набір тестів на явні помилки. Це швидкі тести, які перевіряють функціональність програми. Програму, яка не справилась з димовим тестуванням, немає сенсу тестувати далі [10].

Повертаючись до програмного забезпечення, димове тестування можна визначити як деякий короткий цикл тестів, що здійснюються після виходу чергового білда. Виконується це для перевірки роботи основного функціоналу програмної системи, що розробляється. Після успішного проходження димового тестування система, що розробляється, відправляється на наступні цикли більш серйозних видів тестів. Якусь подібність мають приймальне тестування та тестування складання, суть яких полягає у невеликих тестах, що визначають долю проекту. Також димне тестування ефективно піддається автоматизації, що й зумовило популярність автоматизації даного виду тестування.

Під час цього етапу тестування потрібно перевірити усі заявлені функції модулів. З цією метою виконаємо описані нижче кроки:

1. Починаємо роботу з додатком. Спостерігаємо Початковий екран українською (рис Б1).
2. Змінюємо мову на англійську. Спостерігаємо Початковий екран англійською (рис Б2).
3. Перевіряємо правильність тексту для обох мов.

4. Натискаємо кнопку «Створити замовлення».
5. Перевіряємо, що потрапили на Екран основного меню (рис. 1.3).
6. Перевіряємо, що загальна сума замовлення дорівнює нулю.
7. Перевіряємо, що немає обраних страв та/або напоїв.
8. Перевіряємо, що кнопка «Підтвердити замовлення» не активна та натискання на неї нічого не змінює .
9. Змінюємо мову на українську.
10. Перевіряємо правильність тексту (рис. 1.4).
11. Перевіряємо, що після натискання на «Скасувати замовлення», потрапляємо на Початковий екран.
12. Повертаємось до Основного меню.
13. Натискаємо на «Додати позицію до замовлення.
14. Пересвідчуємось, що відкрилось Меню вибору страв (рис. 1.5).
15. Змінюємо мову на англійську (рис. 1.6).
16. Перевіряємо правильність тексту обома мовами.
17. Перевіряємо, що після натискання на «Не додавати нову страву», потрапляємо у Основне меню.
18. Повертаємось до Меню вибору страв.
19. Обираємо опцію «Напій».
20. Перевіряємо, що потрапили до Меню вибору напоїв (рис. 1.7).
21. Змінюємо мову на українську (рис. 1.8).
22. Перевіряємо правильність тексту обома мовами.
23. Перевіряємо коректну роботу «Каруселі» обрання напоїв.
24. Перевіряємо крайні положення «Каруселі».
25. Обираємо напій.
26. Перевіряємо, що потрапили до Основного меню (рис. 1.9).
27. Перевіряємо, що напій додався до замовлення.
28. Перевіряємо «Карусель» при одній позиції в замовленні.
29. Звіряємо вартість замовлення з очікуваною.

30. Перевіряємо, що кнопка «Підтвердити замовлення» стала активною.
31. Видаляємо напій із замовлення, натискаючи на кнопку в правому верхньому куті прямокутника з напоєм.
32. Перевіряємо, що напій зник із замовлення.
33. Перевіряємо, що кнопка «Підтвердити замовлення» не активна.
34. Звіряємо вартість замовлення з очікуваною.
35. Переходимо до Меню вибору страв.
36. Обираємо опцію «Додатково».
37. Перевіряємо, що ми знаходимось в додатку у стані Меню вибору додаткових страв (рис. 1.10).
38. Перевіряємо коректну роботу «Каруселі» додаткових страв.
39. Перевіряємо крайні положення «Каруселі», гортаючі її праворуч та ліворуч.
40. Перевіряємо текст інтерфейсу обома мовами.
41. Обираємо Шоколадне печиво.
42. Перевіряємо, що потрапили до Основного меню.
43. Перевіряємо, що обране печиво коректно додано до замовлення та відображається правильно (рис. 1.11).
44. Переходимо до Меню вибору страв.
45. Обираємо опцію «Канапка».
46. Перевіряємо, що потрапили до Конструктору канапок (рис. 1.12).
47. Перевіряємо текст інтерфейсу користувача англійською та українською мовами.
48. Перевіряємо, що кнопка «Додати страву» не активна.
49. Додаємо інгредієнти до канапки.
50. Перевіряємо вертикальне зміщення інгредієнтів.
51. Прибираємо інгредієнт з канапки.
52. Додаємо інгредієнт знову.
53. Перевіряємо вертикальне зміщення інгредієнтів.

54. Перевіряємо текстові елементи обома мовами.
55. Перевіряємо ціноутворення.
56. Перевіряємо утворення загальної ваги канапки.
57. Перевіряємо утворення загальної калорійності канапки.
58. Повторюємо створення канапки для перевірки очищення конструктора канапок та його коректної роботи.
59. Обираємо опцію «Канапка».
60. Перевіряємо, що потрапили до Конструктору канапок (рис. 1.12).
61. Перевіряємо текст інтерфейсу англійською та українською мовами.
62. Перевіряємо, що кнопка «Додати страву» не активна.
63. Додаємо інгредієнти до канапки.
64. Перевіряємо вертикальне зміщення інгредієнтів.
65. Прибираємо інгредієнт з канапки.
66. Додаємо інгредієнт знову.
67. Перевіряємо вертикальне зміщення інгредієнтів.
68. Перевіряємо текстові елементи обома мовами.
69. Перевіряємо ціноутворення.
70. Перевіряємо утворення загальної ваги канапки.
71. Перевіряємо утворення загальної калорійності канапки.
72. Підтверджуємо замовлення.
73. Перевіряємо перехід на Завершальний екран (рис. 1.18, рис. 1.19).
74. Перевіряємо наявність отриманого із сервера унікального ідентифікатора замовлення.
75. Перевіряємо сформоване замовлення на Сервері (рис. 1.20).
76. Формуємо та підтверджуємо ще одне замовлення на клієнті.
77. Перевіряємо два замовлення на сервері (рис. 1.21).
78. Звіряємо ідентифікатори замовлень.
79. Натискаємо на кнопку в лівому верхньому куті замовлення.
80. Перевіряємо, що замовлення зникло.

4.1.3. Тестування якості коду

Скористаємось платформою SonarQube для визначення якості написаного коду.

SonarQube – це платформа з відкритим вихідним кодом, розроблена компанією SonarSource для постійного контролю якості коду для виконання автоматичних перевірок зі статичним аналізом коду для виявлення помилок і запахів коду на 29 мовах програмування. SonarQube пропонує звіти про дубльований код, стандарти кодування, модульні тести, покриття коду, складність коду, коментарі, помилки та рекомендації щодо безпеки.

Також SonarQube надає інструменті та публічний програмний інтерфейс для того, щоб записувати історію числових та інших показників роботи ПЗ та надавати графіки їх подальшого прогнозованого розвитку. SonarQube забезпечує повністю автоматизований аналіз та інтеграцію з іншими технологіями, такими як Maven, Ant, Gradle, MSBuild та інструментами постійної інтеграції (Atlassian Bamboo, Jenkins, Hudson тощо).

На сьогоднішній день це один, або ж найвідоміший спосіб автоматичного аналізу коду та його ревью. Популярністю він зобов'язаний тому, що цей сервіс безкоштовний і доступний, а також для встановлення не потрібно багато зусиль. Інтерфейс виглядає сучасно та зрозуміло.

У процесі розробки SonarQube використовувався для контролю якості програмного забезпечення. Також після повного завершення розроблення програмної платформи для закладів харчування на основі 3D візуалізації був проведений найбільш повний можливий аналіз ПЗ. Проведений аналіз не виявив недоліків (рис. 8).

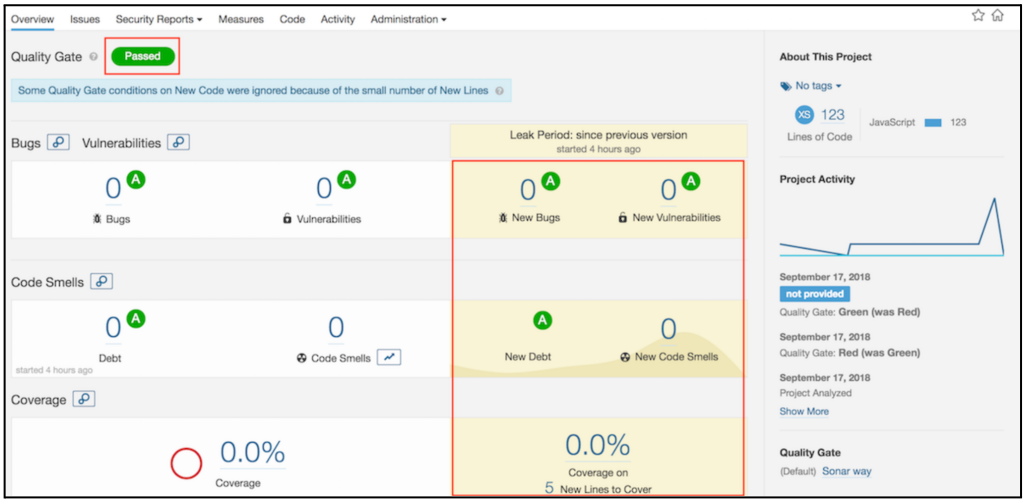


Рис. 8. Результат аналізу коду з використанням SonarQube

4.1.4. Використання сервісу контролю життєздатності ПЗ

Скористаємось сервіосм для контролю життєздатності та для накопичення, аналізу роботи створеної програмної платформи закладів харчування на основі 3D візуалізації. Оберемо New Relic, який повністю задоволняє потребу в виконанні вищезгаданих задач.

New Relic – це програмне забезпечення, яке відстежує різні цифрові середовища. На практиці він сканує середовище для діагностики проблем з продуктивністю, полегшує життя як програмістам, так і менеджерам.

Наприклад, це ефективний додаток для виправлення помилок програмного забезпечення сторінки, в якому спостеригається повільна робота та регулярни затримки. Замість того, щоб розробникам витрачати години робочого часу в пошуках джерела проблеми, з використанням програмних засобів платформі New Relic можна просканувати проєкт та повернути інформацію, необхідну для вирішення проблеми.

New Relic надає багато корисної інформації, яка може бути дуже корисною для компаній-розробників, наприклад, час завантаження кожної сторінки та середній час доступу до кожного клієнта. Сервис дозволяє отримувати реальний і детальний стан вашої мережі, інфраструктури, додатків, взаємодії з кінцевим користувачем, моделей машинного навчання тощо.

Також можливий комплексний пошук, агрегація та аналіз надісланих даних на вебсторінці NewRelic, яка має зручний графічний інтерфейс користувача (рис 9).

Перелік переваг та можливостей, які надає NewRelic:

1. Моніторинг продуктивності додатків (APM): New Relic надає потужні можливості APM, що дозволяє отримати глибоке уявлення про продуктивність ваших додатків. Він відстежує час відповіді додатків, трасування транзакцій і продуктивність бази даних, допомагаючи вам виявляти й усувати вузькі місця, оптимізувати код і покращувати загальну продуктивність додатків.
2. Моніторинг інфраструктури: за допомогою New Relic ви можете контролювати інфраструктуру, що підтримує ваші програми, включаючи сервери, віртуальні машини, контейнери та хмарні служби. Він дає змогу відстежувати ЦП, пам’ять, використання диска, продуктивність мережі тощо, даючи змогу бачити стан і продуктивність базової інфраструктури.
3. Моніторинг і сповіщення в режимі реального часу: New Relic забезпечує моніторинг і сповіщення в реальному часі, дозволяючи вам завчасно виявляти проблеми та реагувати на них, перш ніж вони вплинуть на ваших користувачів. Він пропонує настроювані правила попередження на основі порогових значень продуктивності, тож ви можете негайно отримувати сповіщення про аномалії чи помилки, забезпечуючи швидше усунення несправностей і мінімізуючи час простою.
4. Моніторинг кінцевих користувачів. Важливо розуміти, як ваша програма працює для кінцевих користувачів. New Relic пропонує можливості моніторингу для кінцевих користувачів, які дають змогу зрозуміти взаємодію з користувачем і продуктивність на різних пристроях, браузерах і географічних місцях. Ці дані

допомагають визначити проблеми з продуктивністю, характерні для певних сегментів користувачів, і відповідно оптимізувати свою програму.

5. Розподілене відстеження: New Relic підтримує розподілене відстеження, що дозволяє відстежувати запити під час їх проходження різними компонентами та службами вашої програми. Він надає візуальне представлення всього потоку запитів, допомагаючи вам зрозуміти залежності, виявити проблеми із затримкою та оптимізувати продуктивність у розподілених архітектурах.
6. Масштабованість і хмарна готовність: New Relic розроблено для масштабування з вашими програмами та інфраструктурою. Незалежно від того, чи використовуєте ви свої програми в традиційних центрах обробки даних чи в хмарі, New Relic може адаптуватися до ваших потреб. Він підтримує моніторинг хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP), надаючи інформацію про продуктивність ваших хмарних служб.
7. Аналітика та звітність: New Relic пропонує потужні можливості аналітики та звітності, що дає змогу аналізувати тенденції ефективності, виявляти закономірності та отримувати корисну інформацію. Він надає настроювані інформаційні панелі та звіти, допомагаючи відстежувати ключові показники, візуалізувати дані та приймати керовані даними рішення для оптимізації продуктивності програми.
8. Інтеграція та екосистема: New Relic інтегрується з широким спектром популярних технологій, фреймворків та інструментів. Він підтримує різні мови програмування, бази даних, веб-сервери та хмарні платформи, дозволяючи вам відстежувати та аналізувати продуктивність усього стеку ваших технологій. Крім

того, New Relic має розгалужену екосистему плагінів і розширень, які ще більше розширюють його можливості.

Загалом New Relic допомагає організаціям підвищити продуктивність додатків, оптимізувати використання ресурсів, швидше вирішувати проблеми та забезпечувати виняткову взаємодію з користувачами. Отримавши глибоке уявлення про продуктивність і поведінку ваших програм та інфраструктури, ви можете приймати обґрунтовані рішення для підвищення ефективності, надійності та задоволеності клієнтів.

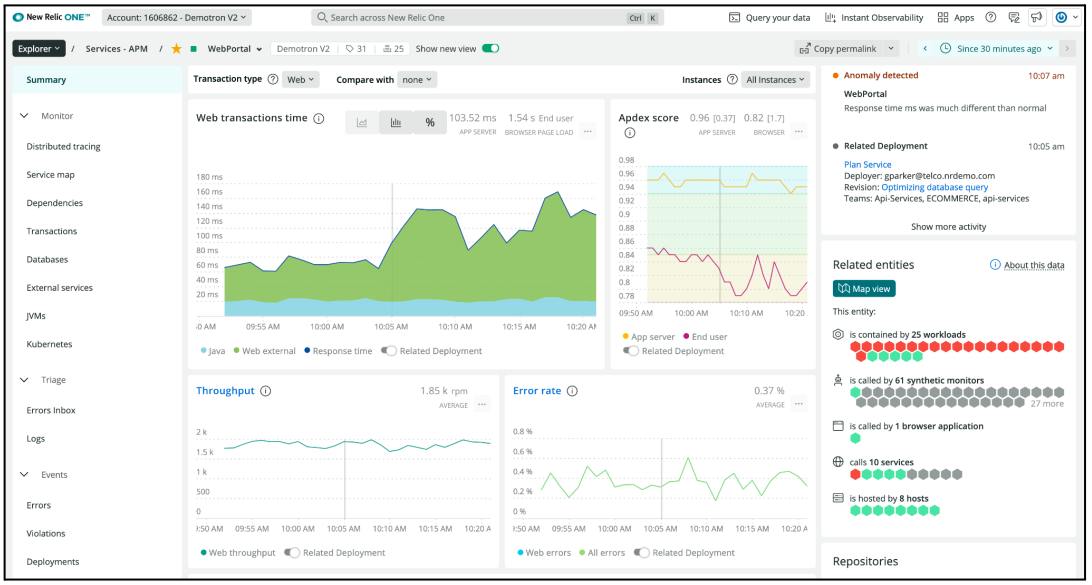


Рис. 9. Знімок екрану вебсторінки сервісу New Relic з аналітичними даними розробленої програмної платформі

4.2. Порівняння розробки з існуючими аналогами

Частиною приготування до створення Програмної платформи для закладів харчування на основі 3D візуалізації було проаналізовано існуючі програмні рішення, виокремлені їх ключові недоліки та переваги, які в деталях розписані в розділі 1.

Розроблена платформа має такі переваги як:

- 3D візуалізація замовлення;

- можливість створення страви з інгредієнтів;
- можливість обрання мови інтерфейсу.

4.3. Рекомендації для подальшого вдосконалення

Подальшими можливостями вдосконалення розробленої Програмної платформи для закладів харчування на основі 3D візуалізації можна вважати наступні:

1. Збільшення кількості опцій: страв, напоїв, інгредієнтів
2. Надання можливості адміністраторам закладів харчування додавати та вилучати опції: страви, напої, інгредієнти
3. Створення додаткової програми для демонстрації поточного стану виконання замовлень в закладі харчування / на веб-ресурсах
4. Покращення якості моделей та текстур об’єктів, які демонструються на програмній платформи з використанням технології перетворення фотографій на готови 3D моделі. Зйомка фотографій – це лише перший крок процесу. Програми-конвертери беруть наявну фотографію та моделюють з неї 3D-рендер. Це найпростіший і найпростіший спосіб створення 3D-зображення. Створені конвертором моделі потребуватимуть ретопології, зменшення кількості полігонів для швидкого рендерінгу об’єктив та ефективної роботи додатку.
5. Вивчити поведінку користувачів. Збирайте дані про взаємодію користувачів з додатком, їхні попередні вибори та дії. Аналізуйте ці дані, щоб зрозуміти, як користувачі використовують додаток та які проблеми виникають. Це допоможе виявити області, які потребують покращення.
6. Спрощувати інтерфейс користувача. Зробіть інтерфейс додатка інтуїтивно зрозумілим і простим у використанні. Зменшіть кількість кроків для виконання завдань і видалити зайві

складнощі, які можуть викликати плутанину або незручності для користувачів.

7. Покращувати швидкість та продуктивність додатку. Оптимізуйте швидкість завантаження додатка та відповідь на дії користувачів. Використовуйте кешування, компресію даних та інші техніки для зменшення обсягу передачі даних. Також підвищуйте продуктивність шляхом оптимізації коду, виправлення пам'яткових витоків та усунення затримок.
8. Забезпечувати стабільність та надійність. Виправляти помилки, які можуть спричиняти збої додатка або призводити до його некоректної роботи. Ретельно тестуйте додаток на різних платформах та пристроях, щоб впевнитись, що він працює безперебійно.
9. Забезпечення оновлення програмного забезпечення. Регулярно оновлюйте операційну систему, веб-сервер та всі програми, що використовуються на сервері. Важливо використовувати останні версії програмного забезпечення, оскільки вони містять покращення безпеки та виправлення вразливостей.
10. Встановлення міцних паролів. Використовуйте складні паролі для всіх облікових записів, включаючи адміністраторський доступ до сервера. Паролі повинні бути довгими, складатися з комбінації великих і малих літер, цифр і спеціальних символів. Також рекомендується використовувати двофакторну аутентифікацію для додаткового рівня захисту.
11. Захист від відомих вразливостей. Перевірте сервер на наявність вразливостей і встановіть необхідні заходи для їх усунення. Слід регулярно сканувати сервер за допомогою засобів, таких як утиліта OpenVAS або Nessus, для виявлення можливих проблем безпеки.

12. Використання захищеного з'єднання. Зашифруйте комунікацію між сервером і користувачем, використовуючи протокол HTTPS. Для цього потрібно встановити SSL-сертифікат і налаштувати сервер таким чином, щоб він вимагав шифрування для всіх з'єднань.

ВИСНОВКИ

Метою цього дипломного проєкту було розроблення Програмної платформи для закладів харчування на основі 3D візуалізації.

Аналіз засобів існуючих, попередньо виконаний в дипломному проєкті, продемонстрував доцільність створення проєкту з використанням ігрового рушія Unity.

Розроблений програмний продукт надає можливість створення замовлення клієнтом закладу харчування, передбачає надання унікального номера-ідентифікатора сформованому замовленню, надає можливості розширення структури програмного модуля та сервера.

Розробка виконана у повному обсязі, всі вимоги враховані, тестування продукту виконано відповідно до затвердженої програми та методики тестування.

Використання розробленої платформи дозволить надавати можливість клієнтам закладів харчування сформувати замовлення, обираючи 3D-візуалізовані позиції; формувати страви з набору інгредієнтів.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ЗАКЛАДІВ ХАРЧУВАННЯ НА ОСНОВІ 3D ВІЗУАЛІЗАЦІЇ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Матвій СУСЛЕНКО

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмна платформа для закладів харчування на основі 3D візуалізації, створена з використанням ігрового рушія Unity3d.

2. МЕТА ТЕСТУВАННЯ

Під час тестування має бути перевірено:

1) робота елементів екранів (сторінок);
2) забезпечення належного рівня безпеки даних;
3) зручність роботи з програмним забезпеченням;
4) відповідність дизайну вимогам, описаним в Технічному завданні.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування ведеться з використанням методу Gray Box Testing. У процесі тестування досліджується як код, так і готовий програмний продукт на відповідність функціональним вимогам.

Будуть використовуватись вказані методи:

1) базове функціональне тестування на рівні Critical path test;
2) перевірка продуктивності ПЗ, а саме Stability testing (тестування стабільності) та Load testing (тестування навантаженням);
3) тестування графічного інтерфейсу користувача.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування буде виконуватись засобами SpecFlow.

Працездатність програмного продукту буде перевірятись шляхом:

1) динамічного ручного тестування;
2) статичного тестування коду;

3) тестування ПЗ на різних платформах: Android, iOS, Windows, Linux, Web;
4) тестування стабільного функціонування при різних штучних умовах;
5) тестування зручності використання;
6) тестування графічного інтерфейсу користувача;
7) тестування засобами SonarQube.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ЗАКЛАДІВ ХАРЧУВАННЯ НА ОСНОВІ 3D ВІЗУАЛІЗАЦІЇ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Матвій СУСЛЕНКО

2023

ЗМІСТ

1. ПРОЦЕС ВЗАЄМОДІЇ З ПРОГРАМНОЮ ПЛАТФОРМОЮ

На старті роботи з програмною платформою користувач знаходиться на Початковому екрані (рис. 9). У кожному стані програми можна змінити мову з української на англійську та навпаки.

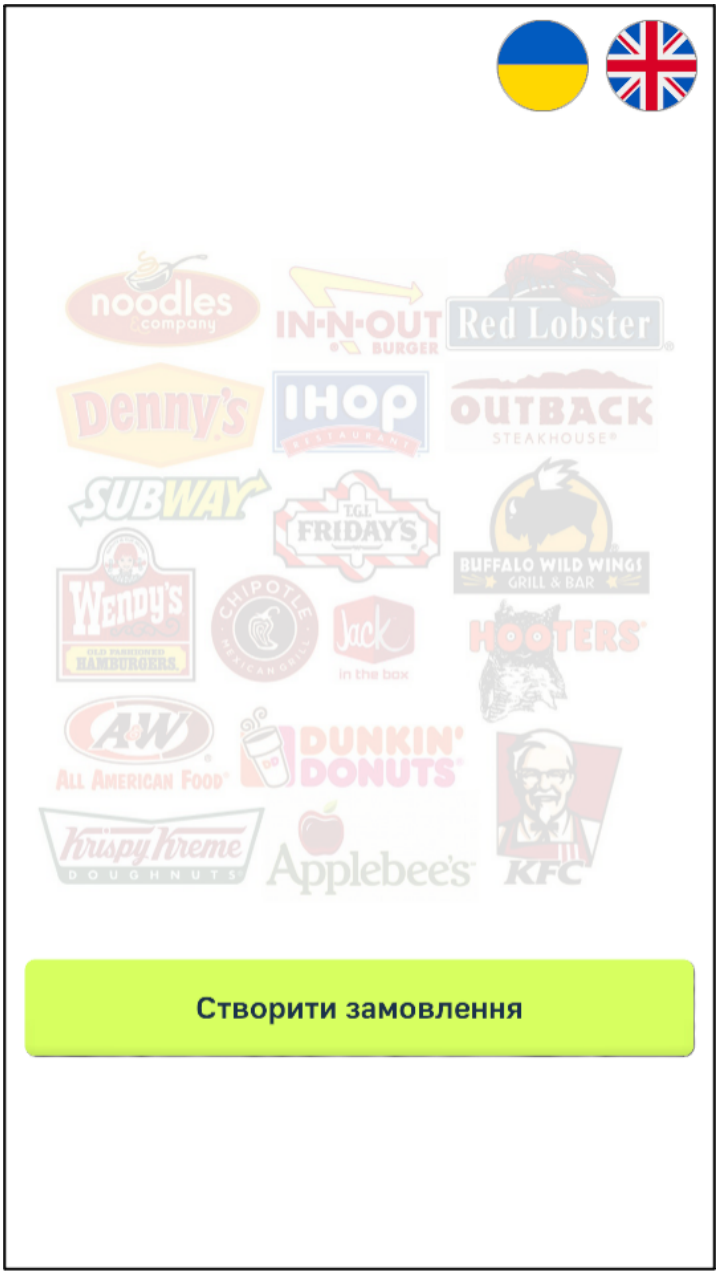


Рис. 1. Початковий екран

Після натискання на кнопку «Створити замовлення», користувач потрапляє на Екран основного меню (рис. 10).

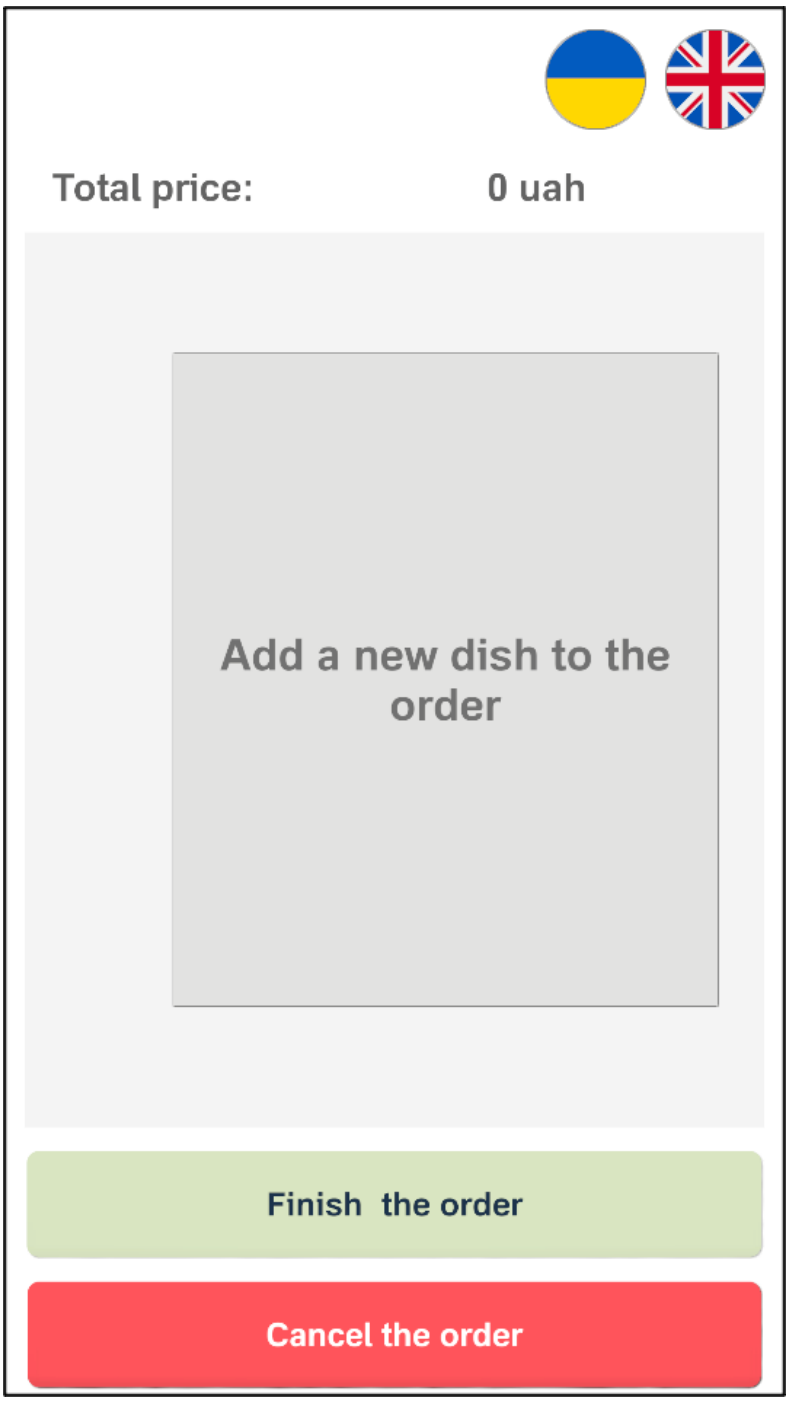


Рис. 2. Екран основного меню

Знаходячись на Екрані основного меню, користувач може скасувати замовлення, підтвердити замовлення у разі наявності обраних або створених позицій або додати позицію до замовлення.

У Меню вибору страв (рис. 11) доступні наступні опції: канапка, напій, додатково. Кожна з кнопок веде до Конструктору канапок, Меню вибору напоїв та Меню вибору додаткових страв відповідно.

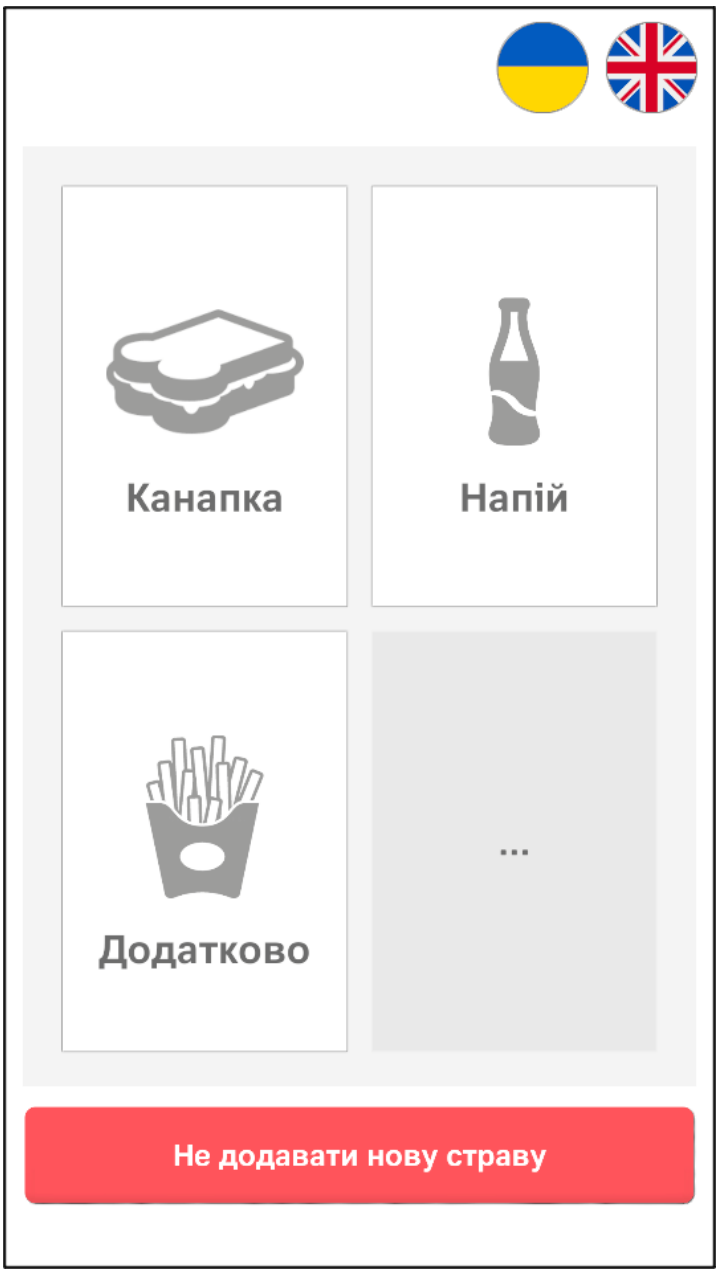


Рис. 3. Меню вибору страв

Усі обрані напої та страви відображаються на Екрані основного меню (рис. 12). Також є можливість видалити позицію із замовлення.

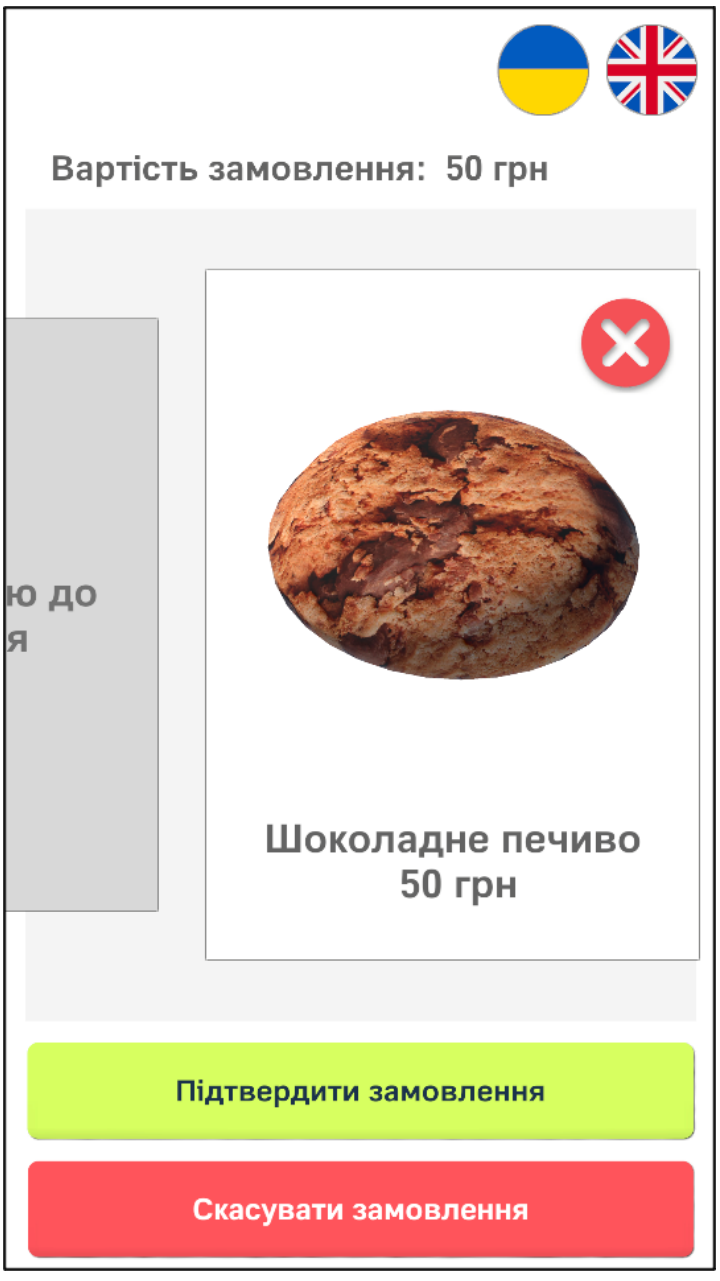


Рис. 4. Екран основного меню з обраною стравою

Після підтвердження замовлення користувач потрапляє на Завершальний екран (рис. 13) із унікальним ідентифікатором замовлення, який наданий сервером.

Рис. 5. Завершальний екран

Перебуваючи на Завершальному екрані, користувач маю змогу повернутись на Початковий екран.

2. ОПИС ЗАСТОСУНКУ

Функціональність застосунку забезпечує процес формування замовлення клієнтом закладу харчування з можливістю обирати готові страви та напої, а також створювати страви зі списку окремих інгредієнтів. Замовлення можна підтвердити та отримати унікальний ідентифікатор замовлення або ж відмінити, що повністю очищає список сформованих позицій.

Користувач може змінити мову застосунку, знаходячись на будь-якому екрані додатку, в будь-якому його стані. Мовна локалізація розповсюджується на всі текстові написи.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

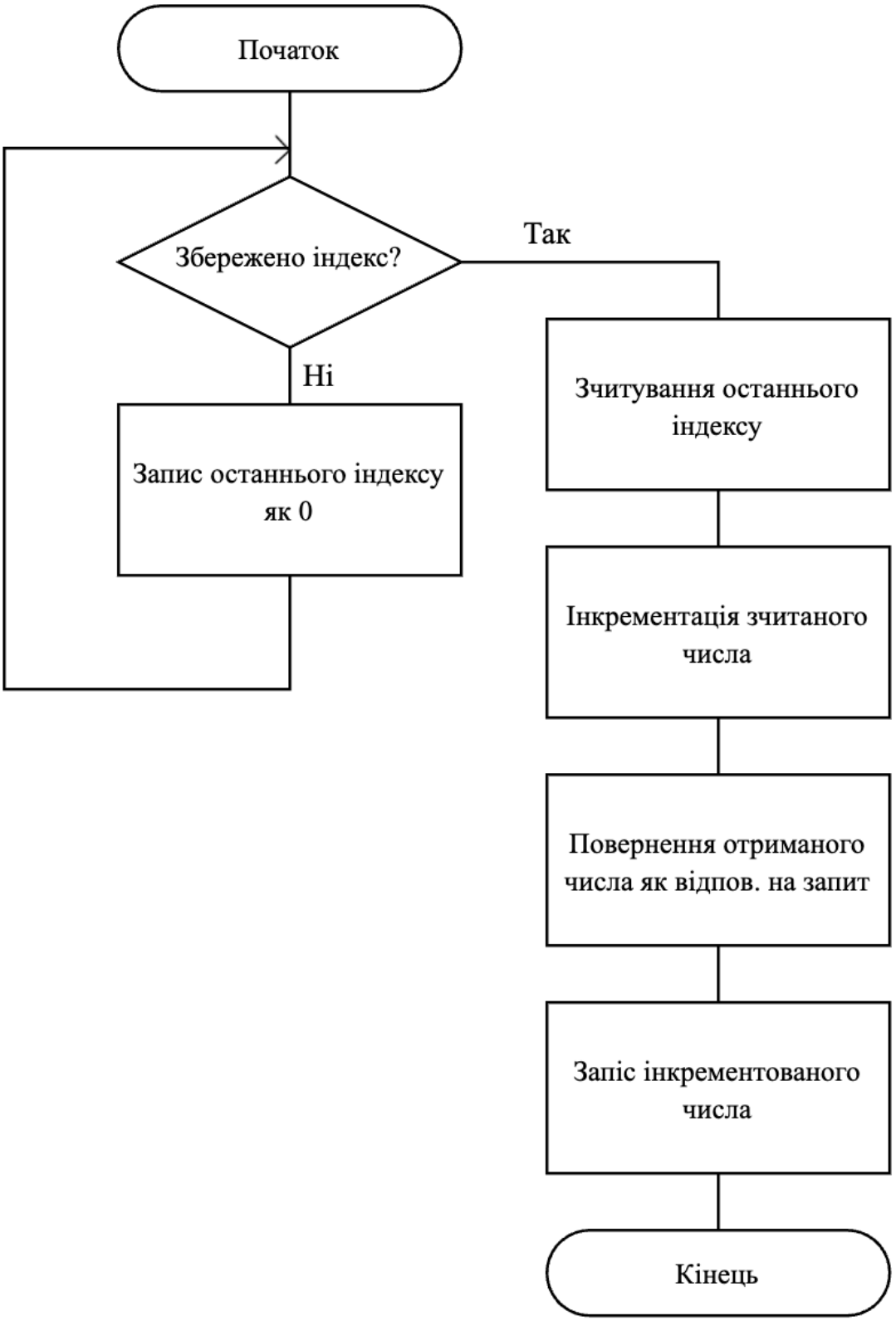
1. Stud File [Електронний ресурс]. — Режим доступу: https://studfile.net. Дата доступу: квітень 2023. Назва з екрану.
2. Godot Engine [Електронний ресурс]. — Режим доступу: https://godotengine.org. Дата доступу: квітень 2023. Назва з екрану.
3. Unreal Engine [Електронний ресурс]. — Режим доступу: https://www.unrealengine.com/. Дата доступу: квітень 2023. Назва з екрану.
4. Free Code Map [Електронний ресурс]. — Режим доступу: https://www.freecodecamp.org. Дата доступу: квітень 2023. Назва з екрану.
5. Unity3d [Електронний ресурс]. — Режим доступу: https://docs.unity3d.com. Дата доступу: квітень 2023. Назва з екрану.
6. Unity3d [Електронний ресурс]. — Режим доступу: https://docs.unity3d.com. Дата доступу: квітень 2023. Назва з екрану.
7. Microsoft — Режим доступу: https://microsoft.com. Дата доступу: квітень 2023. Назва з екрану.
8. Educational — Режим доступу: https://educational.mariroz.com. Дата доступу: квітень 2023. Назва з екрану.
9. QA Light UA [Електронний ресурс]. — Режим доступу: https://qalight.ua. Дата доступу: квітень 2023. Назва з екрану.
10. Димове тестування [Електронний ресурс]. — Режим доступу: http://bit.do/eTKS7. Дата доступу: травень 2019. Назва з екрану.
11. Shostack, A. Threat Modeling: Designing for Security. Indianapolis: Wiley, 2014. 304 p.
12. Haverbeke, M. Eloquent JavaScript: A Modern Introduction to Programming: No Starch Press, 2018. 472 p.
13. Javascript Engine & Performance Comparison (V8, Chakra, Chakra Core): [Електронний ресурс]. — Режим доступу:

https://developers.redhat.com/blog/2016/05/31/javascript-engine-performance-comparison-v8-charkra-chakra-core-2/.
14. Stefanov S. React Up and Running: O'Reilly Media, 2016. 298 p.
15. Ruebbelke L. AngularJS in Action: Manning Publications, 2015. 192 p.
16. Garcia-Molina H., Ullman J. D., Widom, J. Database Systems: The Complete Book: Pearson, 2008. 1296 p.
17. Eseme S. Architecting Vue.js 3 Enterprise-Ready Web Applications: Build and deliver scalable and high-performance, enterprise-ready applications with Vue and JavaScript, 2023. 272 p.
18. Effective Software Test Automation: Developing an Automated Software Testing Tool / Li K., Wu M., Li T., Chen D. : CRC Press, 2017. 400 p.
19. Yablonski J. Laws of UX: Using Psychology to Design Better Products & Services, 2020. 150 p.
20. What Is the Cost of a Requirement Error: [Електронний ресурс]. — Режим доступу: https://www.stickyminds.com/requirement-error.
21. Rider: [Електронний ресурс]. — Режим доступу: https://www.jetbrains.com/rider/.

ДОДАТКИ

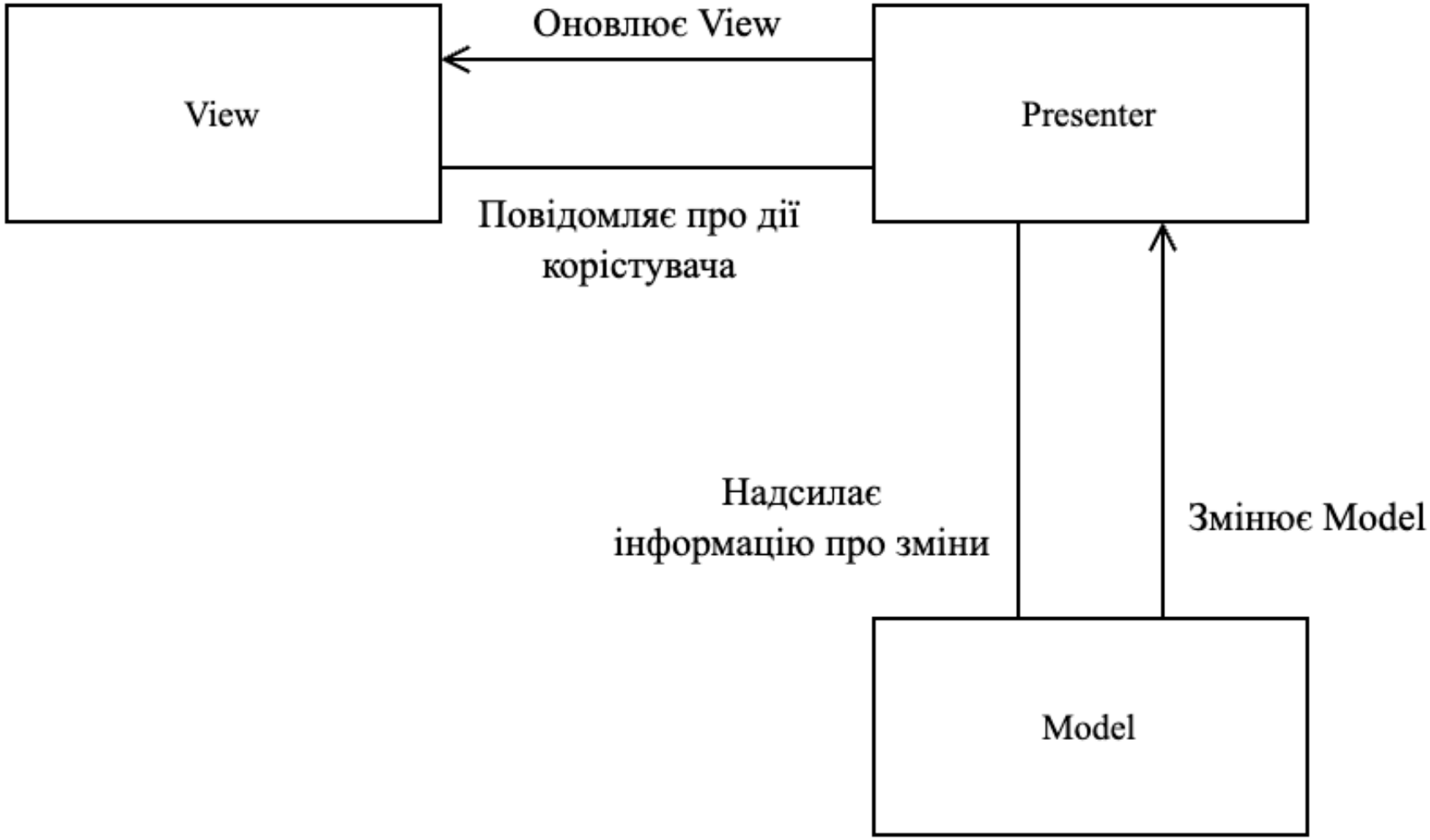
Додаток 1

Копії схематичних матеріалів

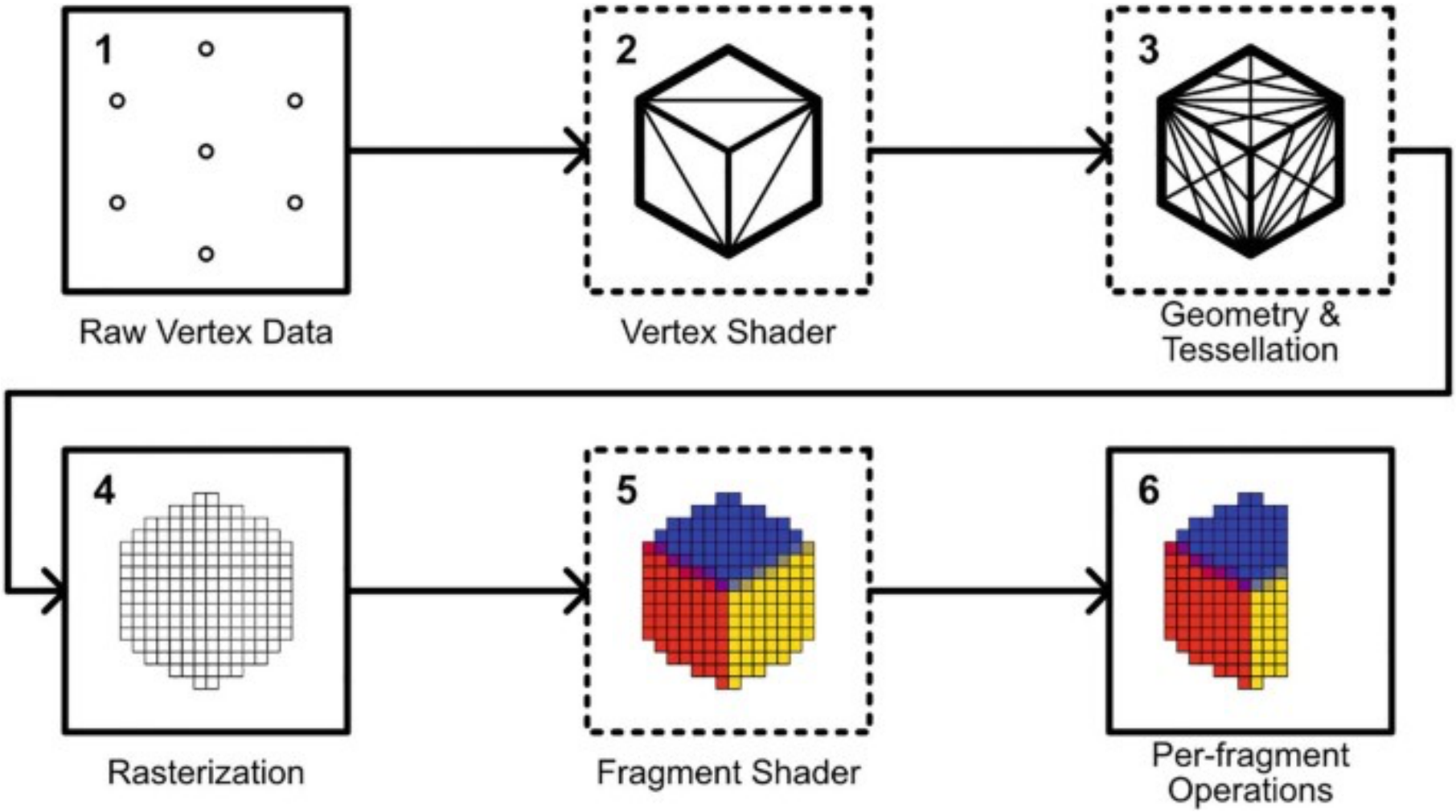


ДП.045440-06-99

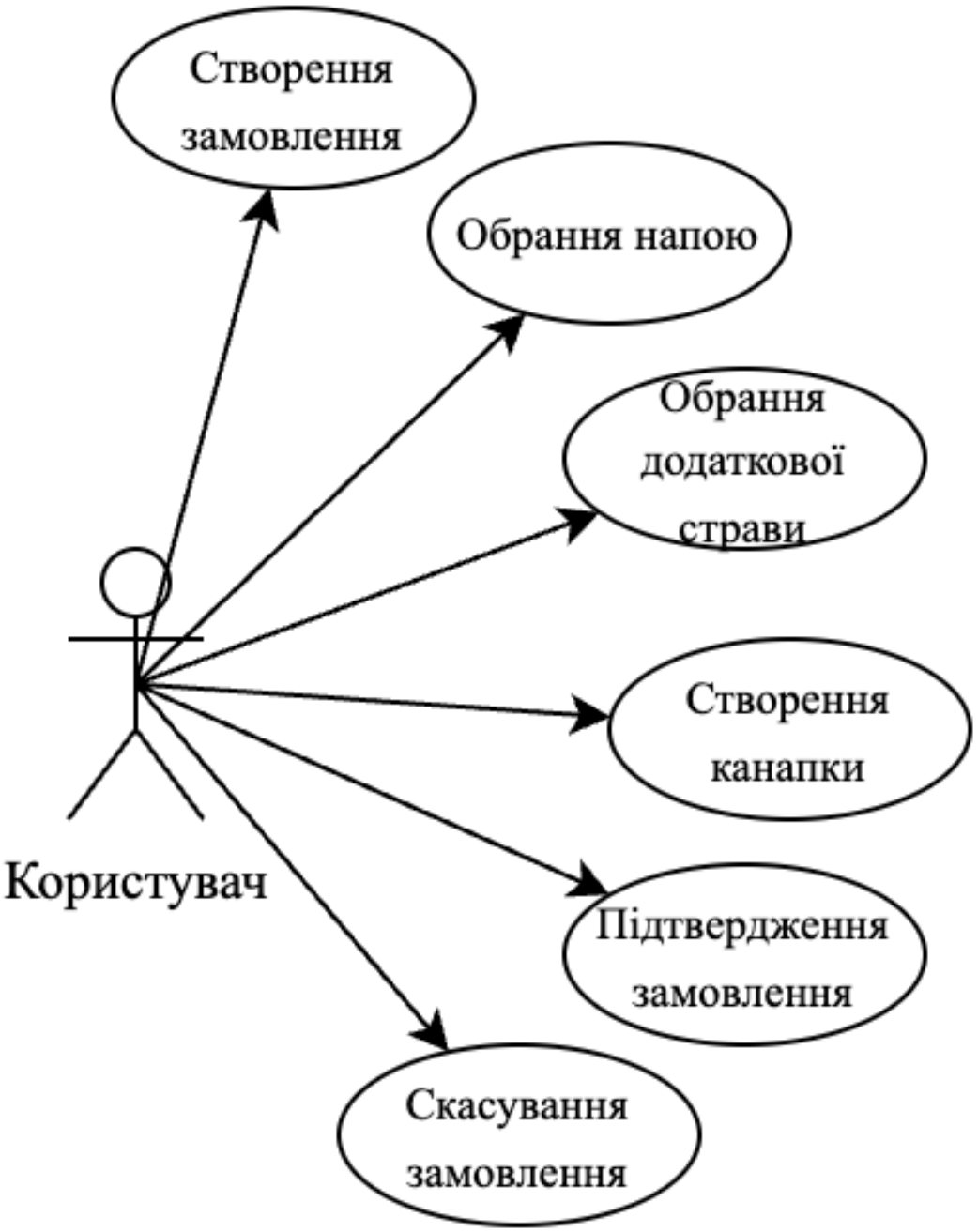
Програмна платформа для закладів харчування на основі 3D візуалізації. Надання унікального номера-ідентифікатора сформованому замовленню. Блок-схема



ДП.045440-07-99
Програмна платформа для закладів харчування на основі 3D візуалізації. Model View Presenter. Схема архітектури. Діаграма класів



Сусленко Матвій, група КП-92



Сусленко Матвій, група КП-92

Додаток 2

Копії графічних матеріалів

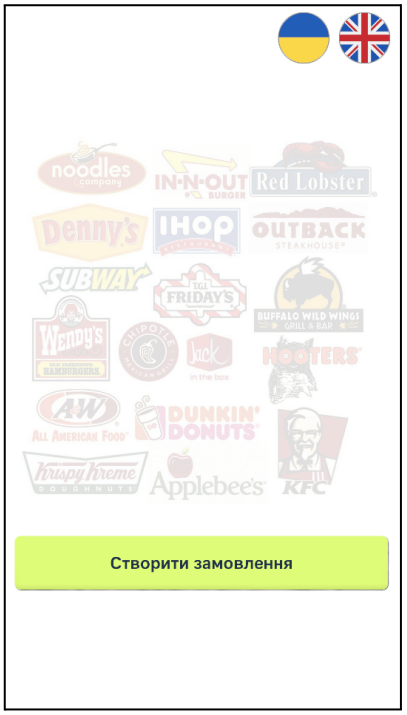


Рис. 2.1. Початковий екран українською

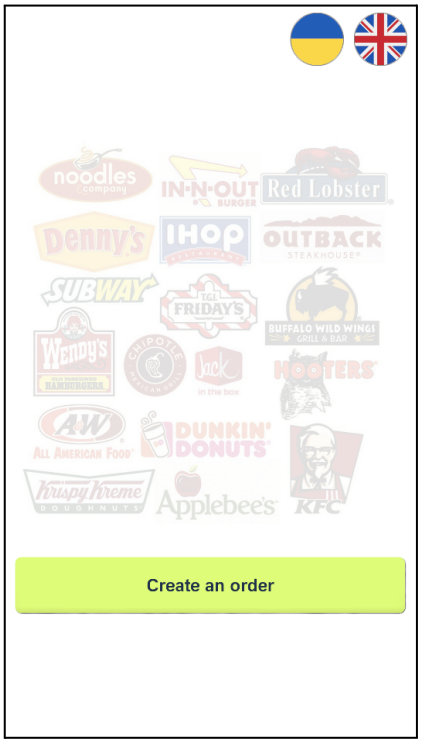


Рис. 2.2. Початковий екран англійською

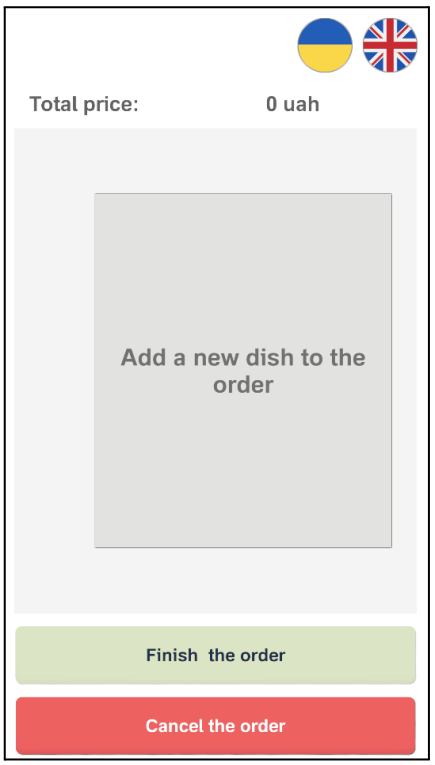


Рис. 2.3. Екран основного меню без обраних страв англійською’’

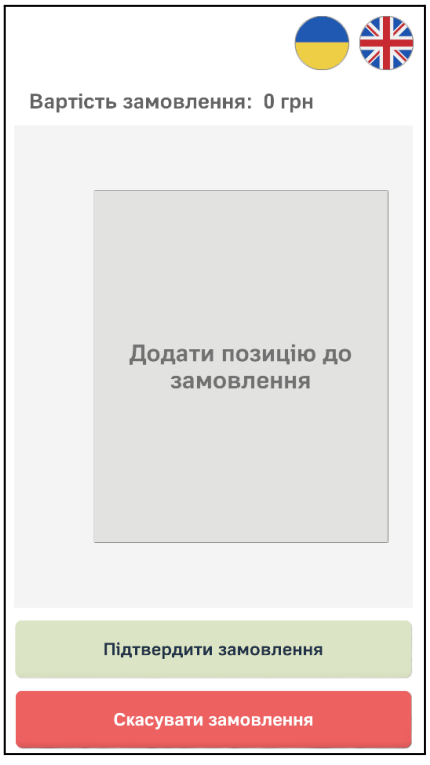


Рис. 2.4. Екран основного меню без обраних страв українською

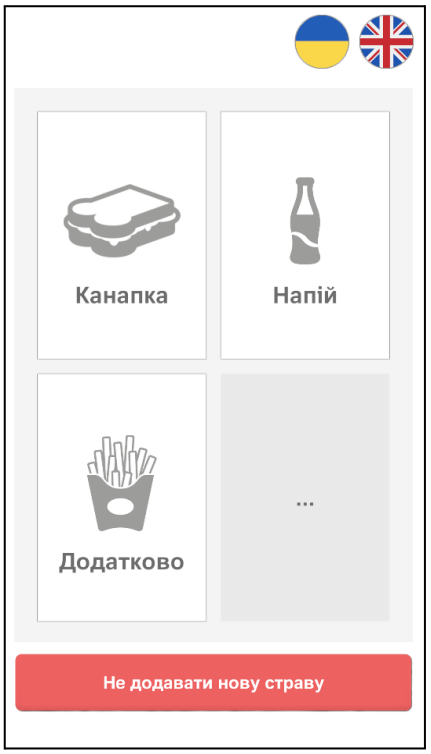


Рис. 2.5. Меню вибору страв українською

Рис. 2.6. Меню вибору страв англійською

Рис. 2.7. Меню вибору напоїв англійською

Рис. 2.8. Меню вибору напоїв українською

Рис. 2.9. Екран основного меню з доданим напоєм

Рис. 2.10. Меню вибору додаткових страв

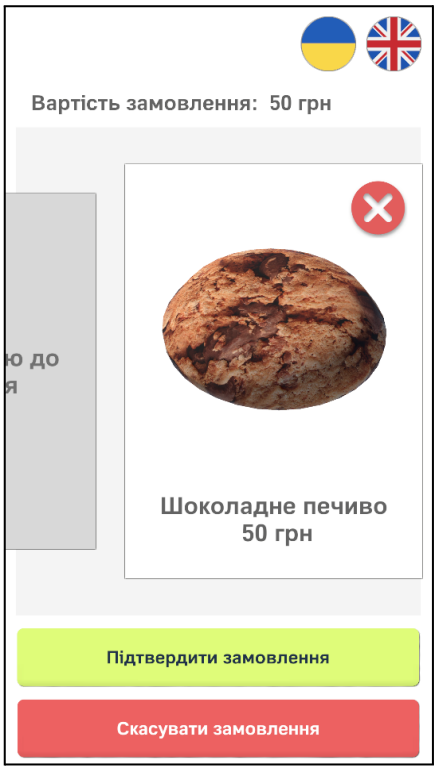


Рис. 2.11. Екран основного меню з доданою додатковою стравою

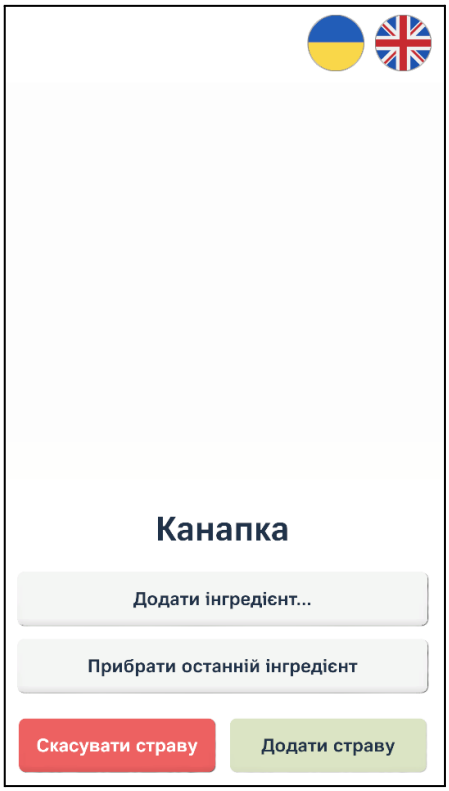


Рис. 2.12. Початковий стан Конструктору канапок українською

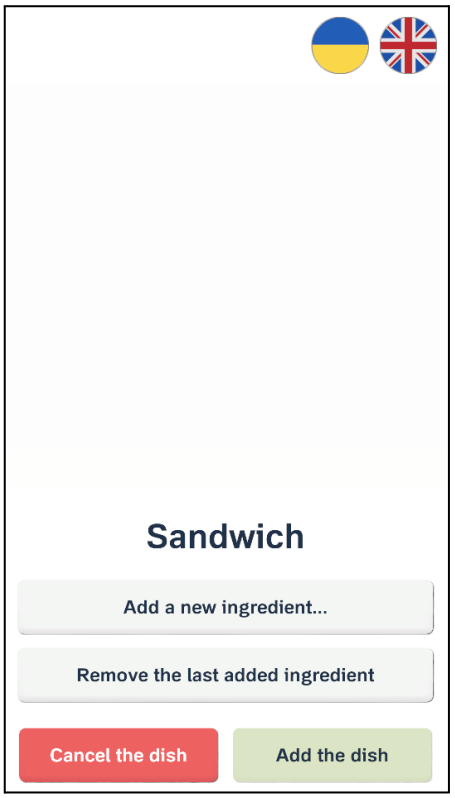


Рис. 2.13. Початковий стан Конструктору канапок англійською

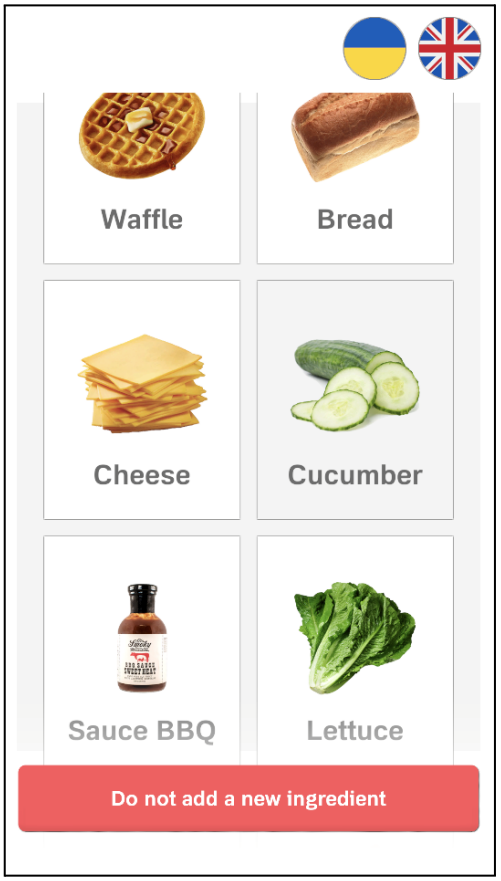


Рис. 2.14. Меню інгредієнтів канапки англійською мовою

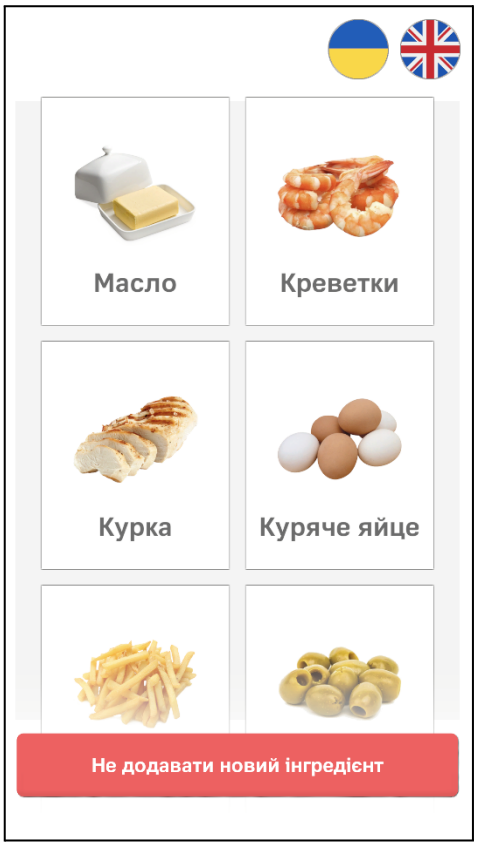


Рис. 2.15. Меню інгредієнтів канапки українською мовою

Рис. 2.16. Конструктор канапок з декількома обраними інгредієнтами

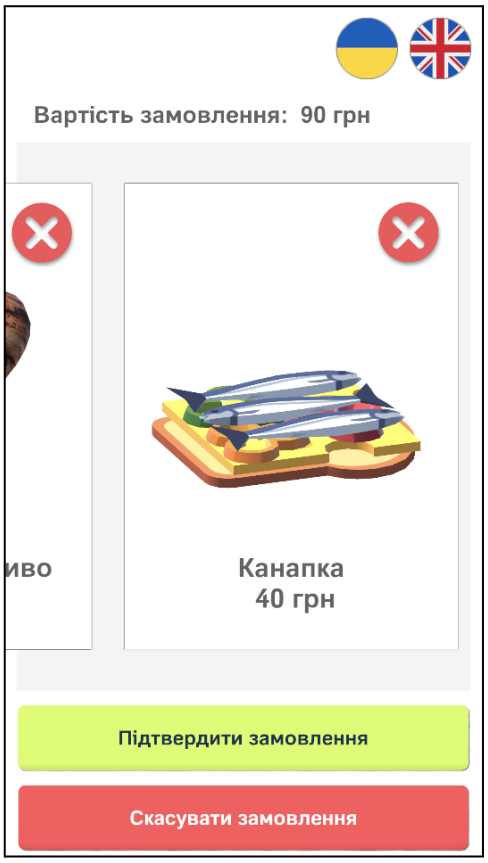


Рис. 2.17. Створена канапка в Основному меню

Рис. 2.18. Завершальний екран українською мовою

Рис. 2.19. Завершальний екран українською мовою

Рис. 2.20. Одне сформоване замовлення на сервері

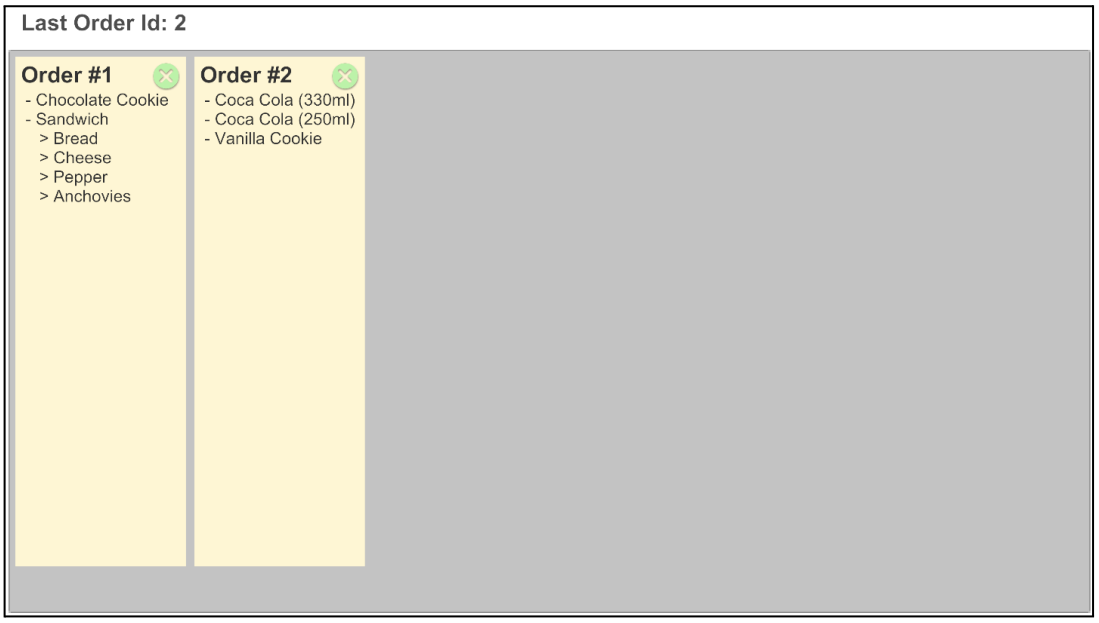


Рис. 2.21. Два сформованих замовлення на сервері

Додаток 3

Лістинг програми

Фрагменти коду логіки оброблення замовлення

```
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Net.Http;
using UnityEngine;
using System.Text;
using System.Linq;
using System;
using Data;

namespace Models
{
    public class OrderModel : Model
    {
        public event Action<int> NewOrderReceived;

        private readonly HttpClient _httpClient = new HttpClient();
        private const string Address = "http://localhost:8000/";
        private const string Endpoint = "order";

        public event Action OrderCleared;
        public event Action<int, DishBehaviour> DishAdded;
        public event Action<int> PriceChanged;
        public int LastOrderId => _lastOrderId;
        public int CurrentPrice => _price;

        private Dictionary<int, IDishInfo> _dishes = new Dictionary<int,
IDishInfo>();

        private int _lastOrderId = 0;
        private int _lastIndex = 0;
        private int _price = 0;

        public void AddDish(DishBehaviour dish)
        {
            _lastIndex++;
            _dishes.Add(_lastIndex, dish.DishInfo);
            DishAdded?.Invoke(_lastIndex, dish);

            _price += dish.DishInfo.Price;
            PriceChanged?.Invoke(_price);
        }

        public async Task SendOrder()
        {
            var body = string.Empty;
            foreach (var dish in _dishes)
            {
                body += $" - {dish.Value.Name._englishName}\n";

                foreach (var ingredient in dish.Value.Ingredients)
                {
                    body += $"    > {ingredient}\n";
                }
            }

            var request = new HttpRequestMessage
            {
                Method = HttpMethod.Post,
                RequestUri = new Uri(Address + Endpoint),
                Content = new StringContent(body, Encoding.UTF8,
"application/json"),
            };

            try
            {
                var response = await
_httpClient.SendAsync(request).ConfigureAwait(true);
```

```
                var responseBody = await
response.Content.ReadAsStringAsync().ConfigureAwait(true);
                NewOrderReceived?.Invoke(int.Parse(responseBody));
            }
            catch (Exception e)
            {
                Debug.LogWarning(e.Message);
            }
        }

        public void CancelDish(int dishId)
        {
            if (!_dishes.ContainsKey(dishId))
            {
                Debug.LogError($"No Dish to Cancel with id: {dishId}");
                return;
            }

            _price -= _dishes[dishId].Price;
            PriceChanged?.Invoke(_price);
            _dishes.Remove(dishId);

            if (!_dishes.Any())
            {
                OrderCleared?.Invoke();
            }
        }

        public void ClearOrder()
        {
            _dishes.Clear();
            OrderCleared?.Invoke();
            _price = 0;
            PriceChanged?.Invoke(_price);
        }
    }
}
```

Фрагменті коду логіки створення канапки

```
using System.Collections.Generic;
using Data.Ingredients;
using UnityEngine;
using System.Linq;
using System;

namespace Models
{
    public class SandwichModel : Model
    {
        public event Action<int, SandwichIngredient> IngredientAdded;
        public event Action LastIngredientRemoved;
        public event Action SandwichCleared;

        public Transform SandwichObjectRoot => _sandwichObjectRoot;
        [SerializeField] private Transform _sandwichObjectRoot;

            private readonly List<KeyValuePair<int, SandwichIngredient>>
_ingredients = new List<KeyValuePair<int, SandwichIngredient>>();

        public float GetTotalCalories()
        {
            return _ingredients.Select(i => i.Value).Sum(ingredient =>
ingredient.Calories);
        }

        public float GetTotalWeight()
        {
```

```
            return _ingredients.Select(i => i.Value).Sum(ingredient =>
ingredient.Weight);
        }

        public int GetTotalPrice()
        {
            return _ingredients.Select(i => i.Value).Sum(ingredient =>
ingredient.Price);
        }

        public void AddIngredient(int index, SandwichIngredient sandwichIngredient)
        {
            _ingredients.Add(new KeyValuePair<int, SandwichIngredient>(index,
sandwichIngredient));
            IngredientAdded?.Invoke(index, sandwichIngredient);
        }

        public void RemoveLastIngredient()
        {
            if (!_ingredients.Any())
            {
                return;
            }

            var lastAdded = _ingredients.Last();
            _ingredients.Remove(lastAdded);

            if (!_ingredients.Any())
            {
                SandwichCleared?.Invoke();
            }
            else
            {
                LastIngredientRemoved?.Invoke();
            }
        }

        public void ClearSandwich()
        {
            _ingredients.Clear();
            SandwichCleared?.Invoke();
        }

        public string[] GetAllIngredientNames()
        {
            return _ingredients.Select(i => i.Value.Name._englishName).ToArray();
        }
    }
}
```

Додаток 4

Копія презентації

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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ЗАКЛАДІВ ХАРЧУВАННЯ НА ОСНОВІ 3D ВІЗУАЛІЗАЦІЇ

Виконав: Сусленко Матвій Олегович

Науковий керівник: Сулема Євгенія Станіславівна

Київ 2023

Актуальність розробки

В дипломному проєкті переглянуті та досліджені можливості найбільш розповсюджених існуючих програмних засобів. Метою проведеного аналізу є порівняння Програмних платформ для закладів харчування з розміщенням меню для клієнтів.

Були досліджені дві різні Програмні платформи для закладів харчування з розміщенням меню для клієнтів:

1) Expirenza by Mono;
2) Resto Market.

Відсутність мовної локалізації, неможливість використовувати програму на інтерактивних стендах для замовлення їжі, відсутність опції замовляти страви дистанційно та створювати страви з окремих інгредієнтів створюють помітні незручності для автоматизації гастрономічного бізнесу.

Отже, проблема є актуальною та запропоноване рішення має можливість отримати свою стійку позицію на ринку.

Вибір засобів реалізації

Для розробки клієнтської частини програмного забезпечення використано ігровий рушій Unity, який є легким для розробки, має зручний інтерфейс, надає реалістичну картинку навіть із параметрами за замовченням, підтримує велику кількість платформ, має потужну спільноту розробників, розгорнуті інструкції та довідники, багату базу допоміжної літератури.

З конвеєрів візуалізації Unity обрано Universal Render Pipeline, який є оптимальним рішенням для швидкої роботи додатку на багатьох платформах, у тому числі на мобільних.

Сервер створено, використовуючи мову C#, яка є основною мовою розробки на Unity.
В якості середовища для розроблення програмного забезпечення було обрано Rider, як найбільш зручне та сучасне інтегроване середовище розробки у парі з Unity.

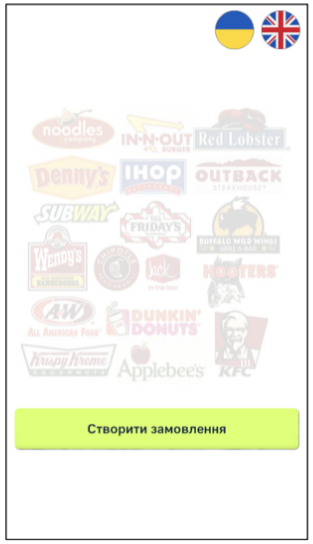
Вибір засобів реалізації

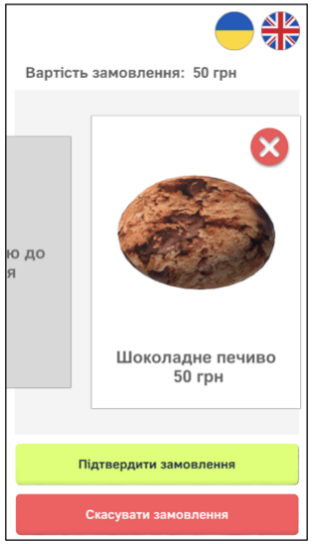
Використано клієнт-серверну архітектуру.

Структура клієнтської частини програмного забезпечення складається з наступних модулів:

- модуль обробки та зберігання інформації про замовлення
- модуль створення канапки з інгредієнтів
- модуль доступних напоїв
- модуль доступних додаткових готових страв
- модуль клієнт-серверної комунікації
- модуль мовної локалізації

Демонстрація роботи додатку





Унікальність

UNICHECK
by Turnitin

User name:
Сулема Євгенія Станіславівна

Check ID:
1015592404

Check date:
13.06.2023 22:01:38 EEST

Check type:
Doc vs Internet + Library

Report date:
16.06.2023 13:30:28 EEST

User ID:
83363

File name: **2023_Bachelor_Suslenko**

Page count: **37** Word count: **5721** Character count: **45776** File size: **295.53 KB** File ID: **1015222382**

9.54% Matches