

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

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»

В.о. завідувача кафедри

_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)

“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Програмне забезпечення інформаційних
управляючих систем та технологій»**

спеціальності «121 Інженерія програмного забезпечення»

на тему *Комплекс програм для управління та моніторингу замовлень
товарів та послуг (комплексна тема). Загальна частина*

Виконав: студент IV курсу, групи ІП-63 Бугайов Олексій
Олександрович _____ (підпис)
(прізвище, ім'я, по батькові)

студент IV курсу, групи ІП-63 Шелудько Дмитро
Максимович _____ (підпис)
(прізвище, ім'я, по батькові)

Керівник ас. Каф. АСОІУ Храмченко М.С. _____ (підпис)
посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові

**Консультант
з графічної
документації** доц., к.т.н., Ліщук К.І. _____ (підпис)
посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові

Рецензент:
к.т.н. Доц каф. ТК, Лісовиченко Олег Іванович _____ (підпис)
посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові

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

Студент _____ (підпис) _____ (підпис)

Київ – 2020 року

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1004053774

Дата перевірки:
15.06.2020 16:56:35 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2020 17:16:36 EEST

ID користувача:
77149

Назва документу: Bugajov_Scheludko_com_ip63

ID файлу: 1004066689 Кількість сторінок: 41 Кількість слів: 6500 Кількість символів: 50776 Розмір файлу: 104.87 KB

2.63% Схожість

Найбільша схожість: 1.83% з джерело https://ela.kpi.ua/bitstream/123456789/31104/1/Kushch_bakalavr.pdf

1.83% Схожість з Інтернет джерелами	1	Page 43
-------------------------------------	---	---------

2.63% Текстові збіги по Бібліотеці акаунту	66	Page 43
--	----	---------

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

Підміна символів

Не знайдено заміненних символів

**Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет (інститут) Інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматизованих систем обробки інформації і управління
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних
управляючих систем та технологій*

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

(підпис) Олександр ПАВЛОВ

“ ” _____ 2020 р.

**ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТАМ**

Шелудько Дмитру Максимовичу
(прізвище, ім'я, по батькові)

Бугайову Олексію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту «Комплекс програм для управління та моніторингу
замовлень товарів та послуг. (комплексна тема). Загальна
частина»

керівник проєкту Храмченко М.С., ас. каф. АСОІУ

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «12» червня 2020 року

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

*1) Аналіз вимог до програмного забезпечення: основні визначення та терміни,
опис предметного середовища, огляд існуючих технічних рішень та відомих
програмних продуктів, розробка функціональних та нефункціональних вимог*

*2) Моделювання та конструювання програмного забезпечення: моделювання та
аналіз програмного забезпечення, засоби розробки, технічні рішення,
архітектура*

програмного забезпечення

3) Розгортання та впровадження програмного забезпечення

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна станів системи

3) Схема бази даних

4) Схема структурна класів програмного забезпечення

5) Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2020 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення рекомендованої літератури	15.03.2020	
2.	Аналіз існуючих методів розв'язання задачі	22.03.2020	
3.	Постановка та формалізація задачі	25.03.2020	
4.	Аналіз вимог до програмного забезпечення	29.03.2020	
5.	Алгоритмізація задачі	05.04.2020	
6.	Моделювання програмного забезпечення	12.04.2020	
7.	Обґрунтування використовуваних технічних засобів	15.04.2020	
8.	Розробка архітектури програмного забезпечення	20.04.2020	
9.	Розробка програмного забезпечення	30.04.2020	
10.	Налагодження програми	10.05.2020	
11.	Виконання графічних документів	17.05.2020	
12.	Оформлення пояснювальної записки	24.05.2020	
13.	Подання ДП на попередній захист	28.05.2020	
14.	Подання ДП рецензенту		
15.	Подання ДП на основний захист	08.06.2020	

Студент _____ Дмитро ШЕЛУДЬКО

_____ Олексій БУГАЙОВ

Керівник _____ Микола ХРАМЧЕНКО

[illegible]

АНОТАЦІЯ

Пояснювальна записка індивідуальної частини дипломного проекту складається з трьох розділів, 25 таблиць та містить 20 джерел – загалом 87 сторінок.

Об’єкт дослідження: застосування, що можуть використовуватися для забезпечення онлайн замовлення та доставки товарів і послуг.

Мета дипломного проекту: спростити процес замовлення товарів та послуг в мережі Інтернет, поліпшити процес моніторингу за станом замовлення, розробити зручний спосіб доставки та отримання товарів та послуг, пришвидшити процес інтеграції постачальників товарів і послуг, розробити розумну бонусну систему для клієнтів, створити зручну систему аналітики для постачальників, створити рейтингову систему, розробити розумну та зручну систему повідомлення клієнтів

КЛЮЧОВІ СЛОВА: ДОСТАВКА, ТОВАРИ ТА ПОСЛУГИ, ІНТЕРНЕТ ЗАМОВЛЕННЯ, АНАЛІТИКА ЗАМОВЛЕНЬ

					КПІ.ІП-6304.045440.01.81	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ABSTRACT

Explanation note of the individual part of the diploma project is folded with three parts, include 25 tables and 20 sources – in summary 87 lines.

The project is completed, and can be used for online procurement and delivery of goods and services.

The subject of the study is to be researched beforehand: to ensure that the online procurement and delivery of goods and services can be avoided.

Diploma`s purpose: Ask the process of conceiving goods and services in the measure of the Internet, improve process of monitoring behind the mill conceiving, disassemble a detailed description of the delivery and decommissioning of goods and services, see the process of integration suppliers goods and services, dissipate a reasonable bonus system for customers, create a strong analyst system for senior customers, create a rating system, dissipate a reasonable and strong notification system for customers.

KEYWORDS: DELIVERY, COMRADES AND SERVICES, INTERNET CONNECTION, ANALITICS CONNECTION

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ВСТУП.....	11
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	13
1.1 ЗАГАЛЬНІ ПОЛОЖЕННЯ	13
1.2 ЗМІСТОВНИЙ ОПИС І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	14
1.2.1 Система уніфікованої реєстрації та авторизації	15
1.2.2 Управління постачальниками	15
1.2.3 Бонус система	15
1.2.4 Аналітика замовлень	16
1.2.5 Система рекомендацій.....	16
1.2.6 Управління користувачами	16
1.2.7 Система оплати послуг.....	17
1.2.8 Рейтингова система	17
1.2.9 Система повідомлення користувачів	18
1.3 АНАЛІЗ УСПІШНИХ ІТ-ПРОЕКТІВ.....	18
1.3.1 Аналіз відомих технічних рішень	18
1.3.2 Аналіз відомих програмних продуктів	18
1.4 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	19
1.4.1 Розроблення функціональних вимог	19
1.4.2 Розроблення нефункціональних вимог.....	43
1.4.3 Постановка комплексу завдань модулю	43
1.5 ВИСНОВКИ ПО РОЗДІЛУ	44
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	45
2.1 МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	45
2.2 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	46
2.2.1 Вибір та структура бази даних	46
2.2.2 Вибір мови програмування та бібліотек	49
2.2.3 Опис структури програмного коду	50
2.2.4 Опис інструментів розробки	58
2.3 ВИСНОВКИ ПО РОЗДІЛУ	61
3 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	62

3.1	РОЗГОРТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	62
3.2	РОБОТА З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ	62
ВИСНОВКИ.....		63
ПЕРЕЛІК ПОСИЛАНЬ.....		66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface – набір чітко визначених методів для взаємодії різних компонентів ПЗ.

JSON – JavaScript Object Notation – текстовий формат серіалізації даних.

REST – Representational State Transfer – підхід до архітектури мережеских додатків та API.

HTTP – Hypertext Transfer Protocol – протокол прикладного рівня для передачі веб-сторінок та інших даних через мережу.

MIT (ліцензія) – група ліцензій для розповсюдження програмного забезпечення з відкритим програмним кодом.

NoSQL – тип баз даних, що відрізняється від SQL-баз даних переважно відсутністю реляційних зв'язків.

BSON – Binary JSON – бінарний формат серіалізації даних, що використовується в MongoDB для збереження документів.

TDD – Test-Driven Development – підхід до розробки програмного забезпечення, при якому розробка починається з попереднього написання тестів, після чого розробляється код, який має пройти ці тести.

Розгортання – процес встановлення програми на комп'ютер

MITM – Man in the middle - термін в криптографії, що позначає ситуацію, коли атакуючий здатний читати та видозмінювати за своєю волею повідомлення, якими обмінюються кореспонденти, причому жоден з останніх не може здогадатися про його присутність в каналі.

Горизонтальне масштабування – процес зміни кількості паралельно запущених програм

DI – Dependency injection - проектування програмного забезпечення, при якому підключення зовнішніх залежностей відбувається автоматично

Скрипт – частина програмного коду, зазвичай малого розміру

					КП.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

IDE – Interactive development environment - інтегроване середовище розробки

NUGET – засіб пошуку та інсталяції програмних залежностей в середовищі .NET

Framework – поняття, що описує готовий програмний продукт, на базі якого можна вести розробку

SWAGGER – програмний засіб, за допомогою якого можна автоматично згенерувати опис усіх точок доступу

GIT – утиліта за допомогою якої відбувається побудова системи версій для програмного коду

					КПІ.ІП-6304.045440.01.81	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

На сьогоднішній день рівень комп'ютеризації проник в усі сфери життя та діяльності людини. Починаючи від цифрових сервісів з прослуховування музики, закінчуючи викликом таксі за допомогою мобільного додатку.

Доставка товарів чи замовлення послуг є повсякденною задачею сучасної людини. Особливо актуальною дана галузь програмного забезпечення стала в реаліях світового карантину, спричиненого поширенням нового вірусу, і тепер питання онлайн замовлення товарів стало не лише питанням комфорту, але і питанням безпеки кожного з нас.

На даний момент ринок цифрових послуг надає будь-кому можливість зробити онлайн замовлення потрібних товарів чи послуг не відходячи від комп'ютеру, або ж навіть за допомогою телефону. Таким чином, на сьогоднішній день ми маємо величезну частину ринку програмних продуктів, спрямовану саме на вирішення цієї проблеми.

Майже кожного дня, ринок України поповнюється новими сервісами та пропозиціями, не винятком стала й галузь онлайн замовлень та доставки продуктів та послуг. Все частіше і частіше ми бачимо різні рекламні оголошення та заохочення скористатись різними додатками та сервісами з виконання онлайн замовлень. Але не завжди такі продукти є перевіреними, зручними та орієнтованими на українську аудиторію.

Тому, наразі, представники малого та середнього бізнесу потребують зручної системи, яка б допомогла їм легко інтегрувати процес обліку та моніторингу власних онлайн замовлень, адже не кожен підприємець має власні ресурси для створення і підтримки такої системи, а як показує практика – зазвичай лише представники великого бізнесу мають достатню кількість матеріальних і людських ресурсів для впровадження такого типу функціоналу.

Тому, метою даного проекту стало створення уніфікованої та зручної системи для поширення будь-яких товарів та послуг за допомогою мережі Інтернет. Таким чином, ми можемо побачити, що програмне забезпечення для

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

дистанційного обліку та моніторингу замовлень є затребуваним, а його розробка і впровадження вкрай актуальним завданням.

					КПІ.ІП-6304.045440.01.81	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

На сьогоднішній день велика кількість людей віддає перевагу отриманню послуг онлайн, неважливо, це продукти із улюбленого ресторану чи, наприклад, товари з продуктового магазину. У сучасних компаніях, особливо в тих, які дотримуються політики дистанційної роботи, усі повсякденні дзвінки відбуваються онлайн. Особливо даний напрямок став дуже актуальним під час карантину, адже кожна свідома людина намагається дотримуватися соціальної ізоляції, проте кожен із нас має свої потреби і хоче їх задовольняти, незважаючи на карантин.

Тож на сьогоднішній день будь-які системи онлайн замовлень стали актуальними. Проте основна проблема полягає у тому, що для того щоб зробити замовлення в будь-якому ресторані чи магазині потрібно або знати їх номер і зробити замовлення у телефонному режимі, або завантажити власний додаток конкретного закладу. Ця ситуація не є зручною для споживачів, адже доведеться завантажувати сотні різних додатків із різними інтерфейсами, і розбиратися з ними. Усі ці додатки займають фізичну пам'ять пристрою і усі вони надсилають власні повідомлення, які відволікають споживачів від своїх власних справ. Тому інтуїтивно на ринку сформувалась потреба на додатки — агрегатори, які б змогли умістити в собі безліч різних закладів, і таким чином спростити проблему отримання товарів і послуг кінцевим споживачем.

Принцип роботи даних додатків наступний. Є дві основні протилежні групи користувачів даного додатку. Перший — це постачальники послуг, заклади, другий — це споживачі даних послуг. Суть даного додатку полягає у тому, що будь-який заклад може легко зареєструватись у даному додатку, та заповнити усі послуги які він може надати своїм споживачам, після цього заклад проходить деяку валідацію, і потрапляє на площину, де будь-який споживач може зайти у розділ із даним закладом і зробити онлайн замовлення. Після того,

як користувач додатку знаходить необхідний йому товар у якогось постачальника товарів, він має можливість зробити онлайн замовлення даних товарів. Постачальник у свою чергу отримує повідомлення про те, що хтось хоче зробити замовлення і підтверджує його, або навпаки, відмовляється його виконувати з тих або інших причин. Якщо постачальник товару зробив підтвердження замовлення, клієнт отримує повідомлення про успішне бронювання. З цього моменту постачальник має можливість встановлювати статус замовлення у своєму власному застосунку, таким чином інформуючи клієнта про процес виконання замовлення.

На даний момент на світовому ринку є декілька нових, особливо в контексті України гігантів, які пропонують своє рішення проблеми агрегації різних закладів всередині одного зручного додатку, проте як завжди кожен додаток має свої переваги, і як не прикро — недоліки, які ми плануємо виправити.

1.2 Змістовний опис і аналіз предметної області

В рамках даної роботи була створена комплексна багатокomпонентна система, що складається із 8 основних модулів:

- управління постачальниками;
- бонус система;
- аналітика замовлень;
- система рекомендацій;
- управління користувачами;
- система оплати послуг;
- рейтингова система;
- система повідомлень користувачів;
- система уніфікованої реєстрації та авторизації.

Вони складають базовий функціонал та є максимально незалежними один від одного. Детально розглянемо кожен компонент та функції, які він виконує в системі.

1.2.1 Система уніфікованої реєстрації та авторизації

Зазначений модуль програмного забезпечення відповідає за спільну систему автентифікації та авторизації в системі додатку. Дана підсистема працює як окремий веб сервер який буде відповідати за цілісність та безпеку даних користувача. Для реалізації даного механізму використовується модуль ASP.NET Identity і ASP.NET Core розгорнутий на веб сервері Kestrel. При відвідуванні користувачем або постачальника відповідного веб сайту, якщо користувач не є авторизованим у системі його запит буде передано на сервер авторизації, де після реєстрації або логіну він буде повернутий на відповідний сайт, таким чином ми маємо спільну недублюючу логіку роботи з даними користувачів.

1.2.2 Управління постачальниками

Даний компонент складається із двох частин: серверної та клієнтської.

Клієнтська частина представляє собою веб-сайт. Постачальники можуть отримати список всіх своїх товарів, змінити чи видали товар. Для цього формуються спеціальні запити, які клієнтська частина надсилає на серверну.

Серверна частина – це WEB API, що дозволяє отримати, видалити, змінити та додати інформацію про постачальників, їх товари та послуги.

1.2.3 Бонус система

Даний компонент складається із серверної та клієнтської частини.

Серверна частина – це WEB API, що дозволяє зберегти, видалити та змінити бонусну систему в БД.

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

Клієнтська частина – це окрема вкладка на веб-сайті для постачальників, де вони можуть керувати бонус системою.

Для створення нової бонус системи, потрібно вибрати критерій, за яким будуть надаватися бонуси, наприклад, при покупці більше ніж 3 одиниць товару, давати знижку 5 відсотків. Також постачальнику доступна модифікація та видалення системи бонусів.

1.2.4 Аналітика замовлень

Даний компонент складається із серверної та клієнтської частини.

Серверна частина – це WEB API, що дозволяє отримувати аналітичні дані за певний проміжок часу, наприклад, кількість замовлень, найпопулярніші та найменш популярні товари.

Клієнтська частина – це окрема вкладка на веб-сайті для постачальників, де вони можуть переглядати аналітику їх бізнесу за певний проміжок часу.

Розглянемо кроки, що проходить постачальник для перегляду аналітичних даних. Спершу постачальник переходить на сторінку аналітики, обирає фільтри (проміжок часу, регіон, дні тижня...) та застосовує їх. Після цього, веб-сайт посилає запит на серверну частину, де формуються дані, та відправляються назад на клієнтську частину, яка, в свою чергу, відображає їх постачальнику.

1.2.5 Система рекомендацій

Даний компонент представляє собою серверну частину. Вона проводить аналіз попередніх замовлень клієнта, та на основі отриманих даних і бонус системи пропонує той чи інший товар. Для надсилання повідомлень на веб-сайт, використовується інший компонент – система повідомлення клієнтів.

1.2.6 Управління користувачами

Даний компонент представляє собою серверну та клієнтську частини для створення, обробки, перегляду та модифікації замовлень, зроблених клієнтом.

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

Розглянемо кроки, що проходить клієнт, для формування та отримання замовлення. Потрапивши на сайт, клієнт вибирає товари, що хоче отримати. Після формування списку покупок, клієнт вибирає спосіб оплати(для цього веб-сайт використовує компонент оплати послуг) та вводить свої дані(адреса, коментар...), після чого замовлення надсилається на серверну частину, де зберігається в БД та передається постачальникам для обробки. Після цього замовлення сформовано та клієнт може переглянути його в своєму кабінеті.

1.2.7 Система оплати послуг

Підсистема оплати послуг представляє собою інтеграцію із сервісом, що надає послуги оплати в мережі Інтернет. Основну частину роботи, сервіс бере на себе, а нам потрібно лише створити обліковий запис та спеціальний токен для подальшої можливості автентифікації в системі сервісу оплати послуг.

На серверній частині створений спеціальний контролер для відображення та обробки запитів на оплату послуг, а на клієнтській частині, спеціальна форма для взаємодії із контролером.

1.2.8 Рейтингова система

Даний компонент відповідає за формування рейтингу клієнтів та постачальників. На веб-сайті клієнтів та постачальників є можливість перейти на вкладку замовлення та поставити оцінку сервісу та клієнту відповідно. Якщо клієнт незадоволений товаром чи доставкою, він може поставити погану оцінку, що в майбутньому негативно вплине на рейтинг сервісу. З іншого боку, постачальник може поставити негативну оцінку клієнту, який не оплатив послуги чи відмінив замовлення без поважної причини.

Рейтингова система складається із серверної частини, що розроблена з використанням Asp.Net Core платформи та MongoDB БД.

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

1.2.9 Система повідомлення користувачів

Даний компонент дає можливість надіслати повідомлення клієнту. Модуль представляє собою серверну частину, яку використовують інші компоненти системи, такі як система рекомендацій та рейтингова система. Сервер формує чергу повідомлень певного клієнта, а веб-сайт періодично робить запит на отримання своєї черги та відображає отримані повідомлення.

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

Основна проблема створення програмного продукту такого плану полягає у тому, що досить важко налаштувати автоматизований процес залучення нових постачальників послуг у систему, що сповільнює розширення спектру послуг які дана система може надати, ця проблема створює ще одну проблему у системі — кастовість, адже для того щоб стати частиною бази постачальників — потрібно щоб постачальник задовольняв власників продукту за деякими параметрами. Ідея ж цього сервісу полягає у тому що процес інтеграції нових закладів у систему стане максимально простим. Для цього потрібно лише зареєструватись і сформулювати список своїх послуг.

1.3.2 Аналіз відомих програмних продуктів

У світі існує безліч додатків із схожим функціоналом, проте мало з яких існують на ринку України. Два найбільш відомих стали: UberEats і Glovo, проте нещодавно стало відомо що UberEats вирішив покинути ринок України, тому фактично наразі ми маємо лише один додаток такого плану на ринку. Glovo — агрегатор закладів харчування і продуктових магазинів, він надає можливість зробити онлайн замовлення в будь-якому закладі і отримати доставку, проте проблемою Glovo є те, що він містить у собі в основному лише заклади харчування, а для того щоб стати частиною платформи Glovo — потрібно пройти

					КП.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

складну систему реєстрації та валідації, що не є досить зручно для постачальників.

1.4 Аналіз вимог до програмного забезпечення

Програмне забезпечення має дві окремі сторони взаємодії, а отже і два окремих наборів вимог.

З точки зору клієнта. Користувач має можливість зареєструватися в системі, після чого користувач може отримати список товарів та послуг, з потрібної категорії, в різних категоріях користувач має можливість сортувати послуги за рейтингом, відстанню та ціною. У разі оформлення замовлення клієнту надається можливість відслідковування статусу його замовлення. Якщо клієнт задоволений якістю замовлення — він має можливість залишити позитивний відгук, в іншому випадку надається можливість залишити розгорнутий негативний відгук.

З точки зору постачальника послуг. Користувач також має можливість зареєструватися в системі, після чого в спеціальній версії додатку, користувач заповняє дані стосовно переліку послуг які він надає. Постачальник має власний кабінет — де він має можливість підтверджувати або ж відхиляти замовлення, також у процесі виконання замовлення постачальник виставляє статус замовлення, який буде відправлятися клієнту, який розмістив дане замовлення. У своєму кабінеті постачальник має можливість формувати звітність стосовно його діяльності.

1.4.1 Розроблення функціональних вимог

Для того щоб визначити перелік функціональних вимог програмного комплексу потрібно виділити, які типи користувачів будуть наявні в системі, даний перелік міститиме:

- неавторизований користувач;
- авторизований клієнт;

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

— авторизований постачальник.

Неавторизований користувач має можливість лише зареєструватися в системі.

Авторизований клієнт має можливість переглядати веб-сайт з функціоналом для клієнтів.

Авторизований постачальник має можливість переглядати веб-сайт з функціоналом для постачальників.

В цілому система повинна мати наступні можливості:

- можливість зареєструватися та авторизуватися в системі (додаток для клієнтів);
- можливість отримати список всіх постачальників (додаток для клієнтів);
- можливість переглянути список товарів постачальника (додаток для клієнтів);
- додавання до кошику товарів від постачальника (додаток для клієнтів);
- замовлення продуктів із кошику (додаток для клієнтів);
- оплата замовлення онлайн в мережі Інтернет (додаток для клієнтів);
- моніторинг статусу замовлення (додаток для клієнтів);
- отримання повідомлень в реальному часі (додаток для клієнтів);
- можливість оцінити якість роботи (додаток для клієнтів);
- отримання рекомендацій щодо товарів на основі попередніх замовлень та існуючих бонусів (додаток для клієнтів);
- можливість реєстрації та авторизації в системі (додаток для постачальників);
- перегляд списку своїх товарів (додаток для постачальників);
- редагування списку своїх товарів (додаток для постачальників);
- управління замовленнями (додаток для постачальників);

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

- перегляд аналітичних та статистичних даних (додаток для постачальників);
- перегляд рейтингу (додаток для постачальників);
- інтеграція бонус системи (додаток для постачальників);
- отримання повідомлень в реальному часі (додаток для постачальників).

Даний список послугував прототипом для створення низки тест-кейсів, що стали незамінними під час проведення функціонального тестування та регресії.

В даному програмному комплексі передбачено наступні можливі варіанти використання програмного забезпечення:

Таблиця 1.1 - Варіант використання USS001

Назва	Реєстрація клієнта.
Опис	Клієнт має можливість зареєструватися.
Учасники	Клієнт.
Передумови	Клієнт має бути незареєстрованим.
Постумови	Клієнт стає зареєстрованим.
Основний сценарій	При відвідування буд-якої клієнтської сторінки у неавторизованому стані, клієнт потрапляє на сторінку з реєстрацією, де він має можливість ввести дані для реєстрації та натиснути кнопку «Зареєструватися».
Розширення сценаріїв	– Система виявляє занадто слабкі паролі. – При виникненні помилки, сервер повідомляє про помилку клієнта.

Таблиця 1.2 - Варіант використання UCC002

Назва	Авторизація клієнта.
Опис	Клієнт має можливість авторизуватися.
Учасники	Клієнт.
Передумови	Клієнт має бути зареєстрованим та неавторизованим.
Постумови	Клієнт стає авторизованим.
Основний сценарій	При відвідування сторінки з авторизацією, клієнт має можливість заповнити поля для авторизації. Для цього клієнту потрібно ввести логін та пароль та натиснути кнопку «Увійти». Після цього у разі якщо введенні дані є вірними, клієнт стає авторизованим.
Розширення сценаріїв	– Система виявляє неправильно введений пароль. – При виникненні помилки, сервер повідомляє про помилку клієнта.

Таблиця 1.3 - Варіант використання UCC003

Назва	Клієнт отримує список усіх постачальників.
Опис	Клієнт має можливість отримати список усіх постачальників

Продовження таблиці 1.3

Учасники	Клієнт.
Передумови	Клієнт має бути авторизованим.
Постумови	Клієнт отримує список усіх постачальників.
Основний сценарій	При відвідування спеціальної вкладки «Постачальники» на клієнтському порталі, клієнт отримує список усіх активних постачальників, з описом кожного з наведених постачальників.
Розширення сценаріїв	

Таблиця 1.4 - Варіант використання USS004

Назва	Клієнт отримує список товарів.
Опис	Клієнт має можливість отримати список товарів конкретного постачальника.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт знаходиться на сторінці зі списком усіх постачальників.
Постумови	Клієнт отримує список усіх товарів конкретного постачальника.

Продовження таблиці 1.4

Основний сценарій	При натисканні на елемент зі списку усіх постачальників, клієнт потрапляє на наступну сторінку, на якій він може переглянути увесь асортимент товарів вибраного постачальника.
Розширення сценаріїв	

Таблиця 1.5 - Варіант використання USS005

Назва	Клієнт додає товари у кошик.
Опис	Клієнт має можливість додати конкретні товари конкретного постачальника у кошик.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт знаходиться на сторінці зі списком товарів конкретного постачальника.
Постумови	Клієнт має збережені у кошику товари.
Основний сценарій	При натисканні кнопки «Купити» у будь-якого товару, з переліку товарів конкретного постачальника, даний товар додається у кошик клієнта.
Розширення сценаріїв	

Таблиця 1.6 - Варіант використання USS006

Назва	Оформлення замовлення.
Опис	Клієнт має можливість замовити товари з кошику.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт знаходиться на сторінці зі списком товарів конкретного постачальника. – Клієнт додав товари до кошику.
Постумови	Клієнт отримує активне замовлення.
Основний сценарій	При натисканні кнопки «Оформити замовлення» у разі якщо клієнт додав будь-які товари до кошику, та після вдалої оплати, клієнт отримує нове активне замовлення.
Розширення сценаріїв	У разі проблем з оплатою, сервер генерує повідомлення про помилку.

Таблиця 1.7 - Варіант використання USS007

Назва	Моніторинг статусу активного замовлення.
Опис	Клієнт має можливість переглянути статус замовлення.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт має активне замовлення.

Продовження таблиці 1.7

Постумови	Клієнт отримує інформацію про статус замовлення.
Основний сценарій	Після оформлення будь-якого замовлення, клієнт може відвідати спеціальний розділ веб-сайту для перегляду статусу замовлень.
Розширення сценаріїв	

Таблиця 1.8 - Варіант використання USS008

Назва	Перегляд історії замовлень.
Опис	Клієнт має можливість переглянути історію усіх замовлень.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт має завершене замовлення.
Постумови	Клієнт отримує інформацію про історію замовлень.
Основний сценарій	У разі якщо клієнт має завершенні замовлення, при відвідуванні спеціальної сторінки з переліком історії замовлень, клієнт отримує список с інформацією про кожне завершене замовлення.
Розширення сценаріїв	

Таблиця 1.9 - Варіант використання USS009

Назва	Отримання статус повідомлень.
Опис	Клієнт отримує повідомлення про зміну статусу замовлення.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт має активне замовлення.
Постумови	Клієнт отримує повідомлення про кожну зміну статусу замовлення.
Основний сценарій	Після оформлення будь-якого замовлення, у разі якщо постачальник змінив статус замовлення клієнт отримує відповідне повідомлення про зміну статусу замовлення.
Розширення сценаріїв	Після отримання повідомлення, спеціальний розділ «Повідомлення» стає червоним.

Таблиця 1.10 - Варіант використання USS010

Назва	Отримання рекомендаційних повідомлень.
Опис	Клієнт отримує повідомлення про наявну рекомендацію.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт має активну рекомендацію.

Продовження таблиці 1.10

Постумови	Клієнт отримує повідомлення про наявну та активну рекомендацію.
Основний сценарій	У разі, якщо спеціальний сервіс з генерації рекомендацій створив рекомендацію для користувача, останній отримує повідомлення про дану подію.
Розширення сценаріїв	Після отримання повідомлення, спеціальний розділ «Повідомлення» стає червоним.

Таблиця 1.11 - Варіант використання USS011

Назва	Клієнт залишає відгук.
Опис	Клієнт має можливість залишити відгук після виконаного замовлення.
Учасники	Клієнт.
Передумови	– Клієнт має бути авторизованим. – Клієнт має завершене замовлення.
Постумови	Клієнт залишив відгук про постачальника.
Основний сценарій	Після того, як постачальник завершив замовлення, клієнт отримує форму з можливістю залишити відгук.
Розширення сценаріїв	

Таблиця 1.12 - Варіант використання USS012

Назва	Клієнт завершає авторизацію.
Опис	Клієнт може примусово завершити сесію.
Учасники	Клієнт.
Передумови	Клієнт має бути авторизованим.
Постумови	Клієнт стає неавторизованим
Основний сценарій	При натисканні кнопки «Вийти», яка знаходиться на клієнтському веб-порталі, клієнт втрачає сесію.
Розширення сценаріїв	

Таблиця 1.13 - Варіант використання UCP001

Назва	Реєстрація постачальника.
Опис	Постачальник має можливість зареєструватися.
Учасники	Постачальник.
Передумови	Постачальник має бути незареєстрованим.
Постумови	Постачальник стає зареєстрованим.

Продовження таблиці 1.13

Основний Сценарій	При відвідування буд-якої постачальницької сторінки у неавторизованому стані, постачальник потрапляє на сторінку з реєстрацією, де він має можливість ввести дані для реєстрації та натиснути кнопку «Зареєструватися».
Розширення сценаріїв	– Система виявляє занадто слабкі паролі. – При виникненні помилки, сервер повідомляє про помилку постачальника.

Таблиця 1.14 - Варіант використання UCP002

Назва	Авторизація постачальника.
Опис	Постачальник має можливість авторизуватися.
Учасники	Постачальник.
Передумови	Постачальник має бути зареєстрованим та неавторизованим.
Постумови	Постачальник стає авторизованим.

Продовження таблиці 1.14

Основний сценарій	При відвідування сторінки з авторизацією, постачальник має можливість заповнити поля для авторизації. Для цього постачальнику потрібно ввести логін та пароль та натиснути кнопку «Увійти». Після цього у разі якщо введенні дані є вірними, постачальник стає авторизованим.
Розширення сценаріїв	– Система виявляє неправильно введений пароль. – При виникненні помилки, сервер повідомляє про помилку постачальника.

Таблиця 1.15 - Варіант використання UCP003

Назва	Постачальник переглядає список товарів.
Опис	Постачальник має можливість переглянути власний список товарів.
Учасники	Постачальник.
Передумови	Постачальник має бути авторизованим.
Постумови	Постачальник отримує список власних товарів.

Продовження таблиці 1.15

Основний сценарій	При відвідуванні сторінки з переліком власних товарів на порталі для постачальників, останній отримує перелік власних активних товарів та їх опис, які є доступними для замовлення.
Розширення сценаріїв	

Таблиця 1.16 - Варіант використання UCP004

Назва	Постачальник додає новий товар.
Опис	Постачальник має можливість додати новий товар до власного асортименту
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з переліком власних товарів.
Постумови	Постачальник створив новий товар

Продовження таблиці 1.16

Основний сценарій	При натисканні відповідної кнопки «Додати» постачальник отримує форму, в якій він має можливість заповнити дані про новий товар, у разі якщо усі введенні дані є правильними, та при натисканні кнопки «Створити» створюється новий товар.
Розширення сценаріїв	Сервер перевіряє введені дані на правильність, та у разі виникнення помилки, інформує про це постачальника

Таблиця 1.17 - Варіант використання UCP005

Назва	Постачальник видаляє товар.
Опис	Постачальник має можливість видалити товар з власного асортименту
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з переліком власних товарів.
Постумови	Постачальник видалив товар

Продовження таблиці 1.17

Основний сценарій	При натисканні відповідної кнопки «Видалити» постачальник отримує форму, в якій він має можливість заповнити підтвердити намір видалити деякий товар зі свого асортименту, у разі, якщо постачальник натискає кнопку «Так», товар видаляється.
Розширення сценаріїв	

Таблиця 1.18 - Варіант використання UCP006

Назва	Постачальник змінює статус замовлення.
Опис	Постачальник має можливість змінити статус активного замовлення
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з переліком активних замовлень. – Постачальник має активне замовлення.
Постумови	Постачальник змінив статус замовлення.

Продовження таблиці 1.18

Основний Сценарій	При відвідуванні сторінки з переліком активних замовлень, постачальник має можливість перевести будь-яке з них у наступний етап, за допомогою кнопки «Наступний етап».
Розширення сценаріїв	При досягненні останнього за порядком статусу замовлення кнопка «Наступний етап» має зникнути.

Таблиця 1.19 - Варіант використання UCP006

Назва	Постачальник відхиляє нове замовлення.
Опис	Постачальник відхиляє замовлення.
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з переліком активних замовлень. – Постачальник має активне замовлення.
Постумови	Постачальник відхилив замовлення.

Продовження таблиці 1.19

Основний сценарій	При відвідуванні сторінки з переліком активних замовлень, постачальник має можливість будь-яке замовлення, яке знаходиться у статусі «До розглядання».
Розширення сценаріїв	

Таблиця 1.20 - Варіант використання UCP007

Назва	Постачальник переглядає аналітику.
Опис	Постачальник має можливість дослідити власну аналітику.
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з аналітикою. – Постачальник має завершені замовлення.
Постумови	Постачальник переглянув аналітику.
Основний сценарій	При відвідуванні вкладки з аналітикою на порталі для постачальників, останній має можливість дослідити спеціально створену власну аналітику, згідно до його виконаних замовлень.

Продовження таблиці 1.20

Розширення сценаріїв	
-------------------------	--

Таблиця 1.21 - Варіант використання UCP008

Назва	Постачальник переглядає рейтинг.
Опис	Постачальник має можливість переглянути власний рейтинг.
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з рейтингом. – Постачальник має завершені замовлення.
Постумови	Постачальник переглянув власний рейтинг.
Основний сценарій	При відвідуванні вкладки з рейтингом на порталі для постачальників, останній має можливість дослідити власний рейтинг, який є середнім арифметичним серед усіх залишених про нього відгуків.
Розширення сценаріїв	

Таблиця 1.22 - Варіант використання UCP009

Назва	Постачальник створює бонус.
Опис	Постачальник має можливість створювати власні бонуси.
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник знаходиться на сторінці з бонус програмою. – Постачальник має товари в асортименті.
Постумови	Постачальник створив власний бонус.
Основний сценарій	При відвідуванні сторінки, яка відповідає за створення бонусів, постачальник має можливість створити власний спеціальний бонус, та вибрати товари з власного асортименту, до яких даний бонус буде закріплений.
Розширення сценаріїв	

Таблиця 1.23 - Варіант використання UCP010

Назва	Отримання постачальником повідомлення про створення нового замовлення.
-------	--

Продовження таблиці 1.23

Опис	Постачальник отримує повідомлення про створення у нього нового замовлення
Учасники	Постачальник.
Передумови	– Постачальник має бути авторизованим. – Постачальник має товари в асортименті.
Постумови	Постачальник отримав повідомлення про створення нового замовлення.
Основний сценарій	У разі, якщо, будь-який клієнт створив нове замовлення у даного конкретного постачальника, останній отримує про це повідомлення.
Розширення сценаріїв	Після отримання повідомлення, спеціальний розділ «Повідомлення» стає червоним.

Таблиця 1.24 - Варіант використання UCP011

Назва	Отримання повідомлення про відгук.
Опис	Постачальник отримує повідомлення про новий відгук
Учасники	Постачальник.

Продовження таблиці 1.24

Передумови	– Постачальник має бути авторизованим. – Постачальник має товари в асортименті. – Постачальник має завершене замовлення
Постумови	Постачальник потримав повідомлення про новий відгук.
Основний сценарій	У разі, якщо, будь-який клієнт створив залишив відгук про даного конкретного постачальника, останній отримує про це повідомлення.
Розширення сценаріїв	Після отримання повідомлення, спеціальний розділ «Повідомлення» стає червоним.

Таким чином ми отримали перелік із усіх варіантів використання системи, які за своєю сутністю можна поділити на дві категорії. Перша категорія варіантів використання відповідає за функціонал клієнтської частини та має кодові назви у форматі USS. Друга категорія варіантів використання відповідає за функціонал постачальницької частини та має кодові назви у форматі UCP.

Взаємозв'язки між функціональними вимогами клієнтського веб-сайту зображені рисунку 1.1.

	UCC001 - Реєстрація клієнта	UCC002 - Авторизація клієнта	UCC003 - Клієнт отримує список усіх постачальників	UCC004 - Клієнт отримує список товарів	UCC005 - Клієнт додає товари у кошик	UCC006 - Оформлення замовлення	UCC007 - Моніторинг статусу активного замовлення	UCC008 - Перегляд історії замовлень	UCC009 - Отримання статус повідомлень	UCC010 - Отримання рекомендаційних повідомлень	UCC011 - Клієнт завершає авторизацію	UCC012 - Клієнт залишає відгук
UCC001 - Реєстрація клієнта												
UCC002 - Авторизація клієнта												
UCC003 - Клієнт отримує список усіх постачальників												
UCC004 - Клієнт отримує список товарів												
UCC005 - Клієнт додає товари у кошик												
UCC006 - Оформлення замовлення												
UCC007 - Моніторинг статусу активного замовлення												
UCC008 - Перегляд історії замовлень												
UCC009 - Отримання статус повідомлень												
UCC010 - Отримання рекомендаційних повідомлень												
UCC011 - Клієнт завершає авторизацію												
UCC012 - Клієнт залишає відгук												

Рисунок 1.1 - Матриця залежності вимог клієнтського веб-сайту

Взаємозв'язки між функціональними вимогами постачальницького веб-сайту зображені рисунку 1.2.

	УСР001 - Реєстрація постачальника	УСР002 - Авторизація постачальника	УСР003 - Постачальник переглядає список товарів	УСР004 - Постачальник додає новий товар	УСР005 - Постачальник видаляє товар	УСР006 - Постачальник змінює статус замовлення	УСР007 - Постачальник відхиляє нове замовлення	УСР008 - Постачальник переглядає аналітику	УСР009 - Постачальник переглядає рейтинг	УСР010 - Постачальник створює бонус	УСР011 - Отримання постачальником повідомлення про створення нового замовлення	УСР012 - Отримання повідомлення про відгук
УСР001 - Реєстрація постачальника												
УСР002 - Авторизація постачальника												
УСР003 - Постачальник переглядає список товарів												
УСР004 - Постачальник додає новий товар												
УСР005 - Постачальник видаляє товар												
УСР006 - Постачальник змінює статус замовлення												
УСР007 - Постачальник відхиляє нове замовлення												
УСР008 - Постачальник переглядає аналітику												
УСР009 - Постачальник переглядає рейтинг												
УСР010 - Постачальник створює бонус												
УСР011 - Отримання постачальником повідомлення про створення нового замовлення												
УСР012 - Отримання повідомлення про відгук												

Рисунок 1.2 - Матриця залежності вимог клієнтського веб-сайту

1.4.2 Розроблення нефункціональних вимог

Даний програмний комплекс мусить відповідати наступним нефункціональним вимогам:

- мати локалізацію інтерфейсу на усіх основних мовах європейського регіону;
- підтримувати режим комунікації з клієнтами лише за допомогою захищених каналів зв'язку;
- містити в собі можливість для легкого горизонтального масштабування;
- відповідати можливостям для легкого вертикального масштабування;
- взаємодія між клієнтом та сервером має відбуватися за парадигмою REST;
- реалізація користувацького інтерфейсу у вигляді веб-сайту.

1.4.3 Постановка комплексу завдань модулю

Дане програмне забезпечення створюється для спрощення процесу інтеграції функціональності онлайн замовлень для представників малого та середнього бізнесу.

Мета створення даної роботи – спрощення процесі інтеграції нових постачальників в систему онлайн замовлень, покращення процесів замовлення товарів та послуг в мережі Інтернет.

Для досягнення поставленої мети, застосунок повинен вирішувати наступні задачі:

- інтеграція нових постачальників;
- додавання нових продуктів;
- створення замовлень;
- створення аналітичних даних;

- спрощена інтеграція бонус системи постачальників;
- можливість оцінювати якість обслуговування;
- можливість отримувати повідомлення в реальному часі;
- розумна система рекомендацій.

Комплекс програм повинен працювати на серверах під управлінням ОС Windows або ОС Linux з встановленими утилітами docker та docker-compose.

1.5 Висновки по розділу

В даному розділі були детально описані та розглянуті функціональні та нефункціональні вимоги до програмного комплексу. Було виділено низку обов'язкових функцій, що система має надавати, а також сформовано набір нефункціональних вимог до комплексу програм, дотримання яких є необхідним для нормальної роботи системи.

Для виділення функціональних вимог, був створений короткий опис кожного компоненту, що дозволило виділити його основні цільові функції. Також був проведений аналіз відомих та успішних проектів, що мають схожу ідею. Це дозволило виокремити додаткові необхідні функції, та створити нові, яких ще немає у конкурентів.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Система управління та моніторингу замовлень товарів та послуг є багатокомпонентною системою. При розробці додатку, за основу були взяті принципи мікро сервісної архітектури, де кожен компонент є ізольованим, запускається та керується незалежно від інших модулів системи. Це дозволить оновлювати функціонал компонентів не перешкоджаючи роботі системи. Також, при розробці окремо взятого компоненту, був використаний принцип модуль як сервіс. Для цього, кожен компонент системи, представляє інтерфейс взаємодії з ним самим. Усі модулі, які для повноцінного функціонування потребують взаємодії з іншими компонентами, використовують цей інтерфейс у зручному для себе вигляді. Так як компоненти є максимально ізольованими один від одного, то помилка в одному взятому модулі не зупинить роботу всієї системи, а лише тієї її частини, що використовує цей компонент. Завдяки розбиттю на сервіси, з'являється можливість підвищити ефективність роботи системи, завдяки її горизонтальному масштабуванні. Також є можливість створити декілька копій одного сервісу, що будуть знаходитися на різних фізичних серверах. Це збереже систему від критичних поломок, стихійних явищ, тощо. Наприклад, якщо у зв'язку зі стихійним лихом, певні із серверів вийдуть з ладу, то інші, де запущена копія модуля успішно продовжать свою роботу. Це звичайно негативно вплине на ефективність системи, бо вона знизиться, в зв'язку зі зменшенням фактору горизонтального масштабування, але все ж залишиться працездатною.

2.2 Архітектура програмного забезпечення

2.2.1 Вибір та структура бази даних

В рамках даної роботи була розроблена розподілена система, що складається із дев'яти компонентів. Кожен із них має свою специфіку роботи та обробки даних. Постало питання вибору оптимального сховища даних та його структури.

Спершу були виявлені основні критерії, яким має задовольняти сховище даних, а потім на їх основі розглянуто декілька варіантів та вибрано найкращий.

Першим і одним із найважливіших є легкість горизонтального масштабування. Оскільки система є легко масштабуємо і в моменти піку навантаження може ставати досить сильним, то сховище даних повинно легко масштабуватися незалежно від масштабування самих компонентів. Другим, але не менш важливим критерієм стала надійність, оскільки в процесі роботи, система оперує грошима користувачів, то надійність має бути високою. Під терміном надійність, ми розглядали високий рівень реплікації даних та легкість збільшення загальної кількості реплік, а також принцип “fault tolerance”.

Кандидатів на сховище даних, ми шукали серед нереляційних баз даних, оскільки вони дуже добре показують себе під час горизонтального масштабування та реплікації.

Спершу була розглянута таке сховище даних, як Redis, розглянемо рисунок 2.1, де зображена схема роботи Redis.

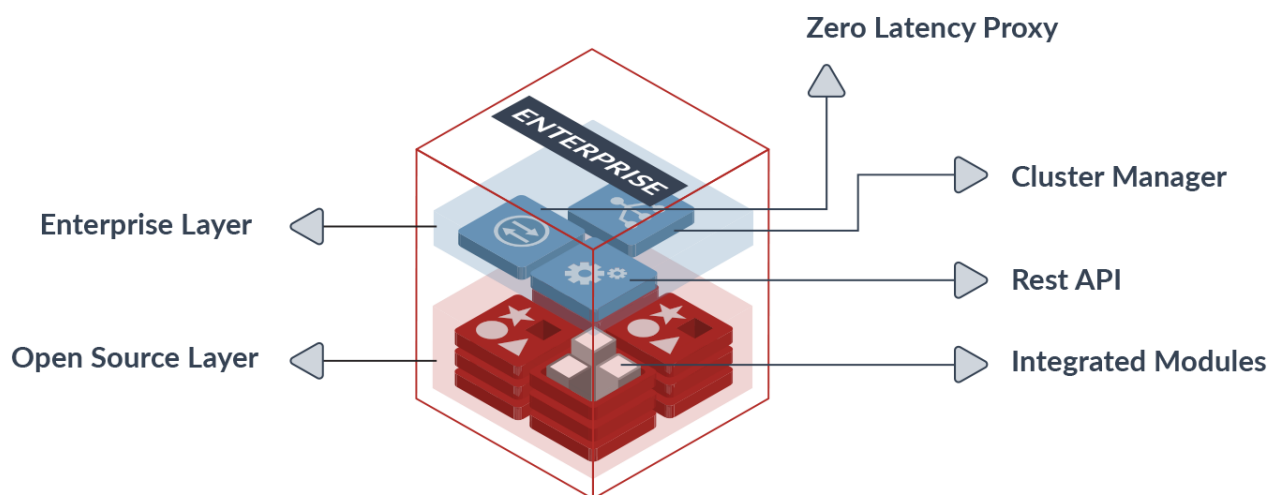


Рисунок 2.1 – Схема Redis

Redis добре масштабується та дозволяє легко налаштувати реплікацію. Також у нього існує клієнт для платформи .NET та мови програмування C#, розглянемо рисунок 2.2.

Clients

The recommended client(s) for a language are marked with a ★.

Clients with some activity in the official repository within the latest six months are marked with a ☺.

Want **your client listed here**? Please fork the [redis-doc repository](#) and edit the clients.json file. **Submit a pull request** and you are done.

Browse by language:

ActionScript	Bash	C	C#	C++	Clojure
Common Lisp	Crystal	D	Dart	Delphi	Elixir
emacs lisp	Erlang	Fancy	gawk	GNU Prolog	Go
Haskell	Haxe	Io	Java	Julia	Lasso
Lua	Matlab	mruby	Nim	Node.js	Objective-C
OCaml	Pascal	Perl	PHP	PL/SQL	Pure Data
Python	R	Racket	Rebol	Ruby	Rust
Scala	Scheme	Smalltalk	Swift	Tcl	VB
VCL	Hojo	Zig			

Рисунок 2.2 - Огляд сховища даних Redis

Тобто процес розробки комунікацій із сховищем даних буде зручним та швидким, завдяки використанню готової сторонньої бібліотеки.

Після проведення детального аналізу можливостей сховища даних Redis, було прийняте рішення продовжити пошук, у зв'язку з обмеженою кількістю можливих команд по обробці та пошуку даних.

Наступний кроком, ми розглянули таке сховище даних, як Cassandra. Основним її плюсом перед іншими базами даних, стало її вузько-направленість та спрощеність процесів масштабування та реплікації. Розробники Cassandra давали гарантії щодо хороших показників високого рівня масштабування, доступності та надійності, зображено на рисунку 2.3.

High Scalability

Cassandra is a highly scalable storage system in which nodes may be added/removed as needed. Using gossip-based protocol a unified and consistent membership list is kept at each node.

High Availability

Cassandra guarantees high availability of data by implementing a fault-tolerant storage system. Failure detection in a node is detected using a gossip-based protocol.

Durability

Cassandra guarantees data durability by using replicas. Replicas are multiple copies of a data stored on different nodes in a cluster. In a multi-datacenter environment the replicas may be stored on different datacenters. If one replica is lost due to unrecoverable node/datacenter failure the data is not completely lost as replicas are still available.

Рисунок 2.3 - Гарантії Cassandra

У зв'язку з великою складністю розгортання даної бази даних, було вирішено продовжити пошуки.

Фінальним та остаточним варіантом виявилася MongoDB. Це нереляційна документ-орієнтована база даних, що дозволяє легко масштабуватися та створювати велику кількість реплік, також вона має велику кількість доступних команд для пошуку та обробки даних. Також дане сховище даних є досить легким для встановлення, тому проблем під час встановлення не виникло. Також MongoDB підтримує мову програмування C#, що полегшило задачу по розробці комунікацій із сховищем даних.

Оскільки MongoDB не має чіткої схеми, то не має потреби в завчасному створенню та налагодженню схема для кожної сутності в системі.

Так, як кожен компонент системи є максимально незалежним від інших, він використовує персональну базу даних, завдяки конфігураціям, можна

					КПІ.ІП-6304.045440.01.81	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

створити спільну для всіх модулів базу, але за замовченням, у кожного компонента своя персональна, як зображено на рисунку 2.4.

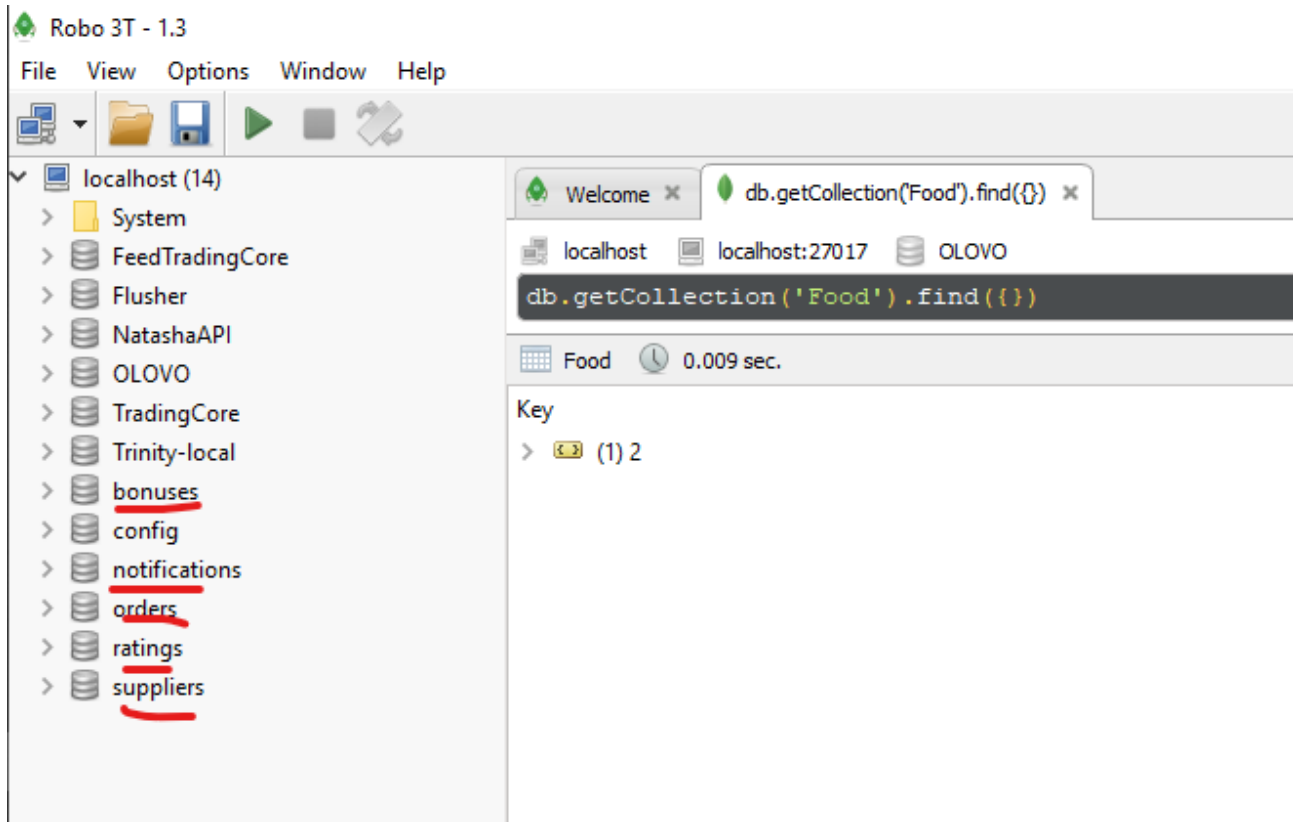


Рисунок 2.4 – Список баз даних у MongoDB

Оскільки дані представлені у вигляді колекції документів та не мають ключів на інші колекції, то у випадках, коли з'являється необхідність поєднання даних, використовуються спеціальні унікальні ідентифікатори, за допомогою яких можна зв'язати одні дані з іншими.

2.2.2 Вибір мови програмування та бібліотек

Для розробки даної системи, була вибрана платформа Asp.Net Core 3, та мова програмування C#. Це дозволило скористатися усіма перевагами об'єктно-орієнтованої типізованої мови програмування та готовими рішеннями, що надає платформа Asp.Net.

Завдяки використанню платформи Asp.Net та мови програмування C# вдалося досягти наступного:

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

- спростити процес хостингу веб-серверу – в платформу вбудований веб-сервер, що має назву Kestrel;
- позбутися залежностей на стороні DI-контейнери – у платформу вбудований DI-контейнер, що дозволяє дуже просто та зручно створювати легко зв'язаний код;
- легкий процес конфігурації – платформа інкапсулює роботу з аргументами командної лінії, файлами, змінними середовища;
- висока пропускну здатність – без додаткових конфігурацій та налаштувань, платформа видає хорошу пропускну здатність та швидкість обробки запитів;
- багато готових рішень для обробки вхідних запитів – платформа спрощує процес серіалізації/десеріалізації даних, роутингу запитів, валідації вхідних даних, тощо;
- легкість вертикального масштабування – завдяки вбудованим механізмам для роботи з пулами потоків, дозволяє легко масштабуватися;
- легкість створення асинхронного серверу – мова С# дозволяє дуже швидко та зручним чином створювати асинхронні методи;
- підтримка grpc – платформа підтримує дану технологію “з коробки”, тому буде досить просто перейти замість традиційного HTTP + JSON на більш новий та перспективний grpc + protobuf.

Оскільки дана система є багатокomпонентною, то це дозволяє розробляти кожен компонент на різних платформах. Це буде дуже актуально під час появи нетипових для Asp.Net Core та С# задач, але на даний момент, усі модулі використовують самі це поєднання.

2.2.3 Опис структури програмного коду

В зв'язку із великою складністю та комплексністю проекту, велика увага була приділена структурі створюваних модулів, архітектурі всередині кожного

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		50

із них. Наразі існує 9 модулів, кожен із яких розміщений в окремій папці в репозиторії, як зображено на рисунку 2.5.

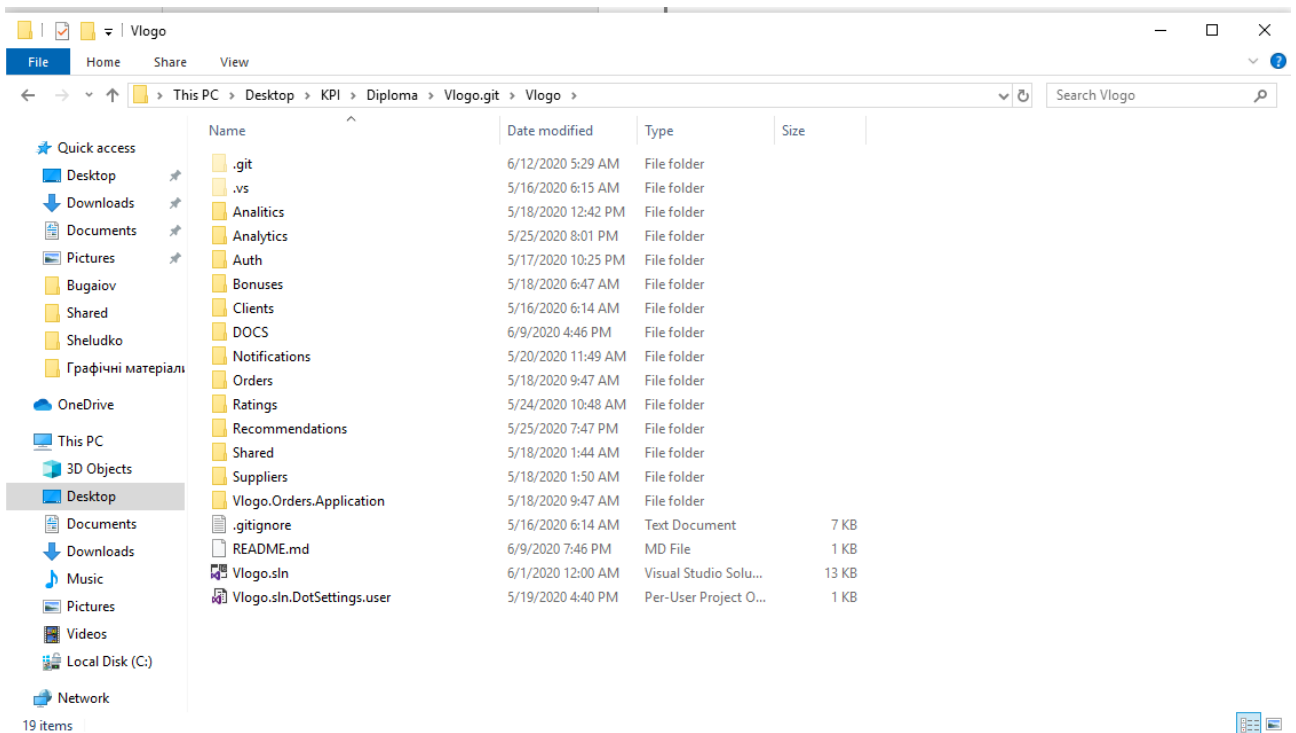


Рисунок 2.5 - Структура директорії

Аналогічна ситуація і в інтегрованому середовищі розробки, кожній фізичній папці співставленні свої логічні папки для групування логічно зв'язаних проектів, як зображено на рисунку 2.6.

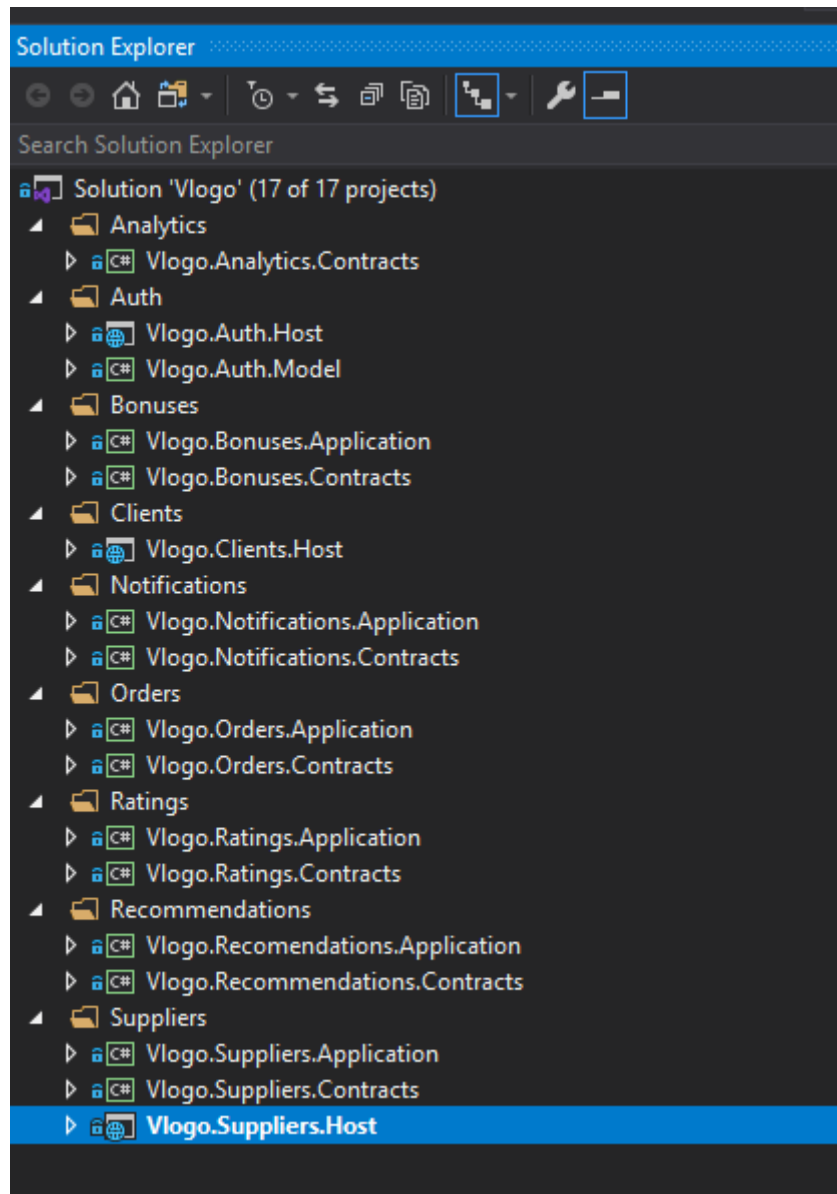


Рисунок 2.6 - Структура проектів

Кожен модуль може мати необмежену кількість проектів, але обов'язково повинен мати окремі проект з контрактами, окремий проект з бізнес-логікою та окремий проект для хостингу. Детальніше існуючі проекти та їх призначення розглянуто в таблиці 2.1.

Таблиця 2.1 - Опис проєктів

Назва модуля	Назва проєкту	Опис
Система аналітики	Vlogo.Analytics.Contracts	Містить контракти для роботи із аналітикою та модулем аналітики, що можуть використовуватися будь-яким іншим компонентом системи.
Система уніфікованої реєстрації та авторизації	Vlogo.Auth.Model	Містить класи призначені для конфігурування системи автентифікації та реєстрації, моделі та контракти, що для цього використовуються

Продовження таблиці 2.1

Система уніфікованої реєстрації та авторизації	Vlogo.Auth.Host	Містить контролери, представлення, моделі, логіку обробки запитів на автентифікацію та реєстрацію, конфігурування веб-серверу.
Система бонусів	Vlogo.Bonuses.Contracts	Містить класи контрактів для роботи з бонусами. Їх використовують сервіси, що потребують комунікації із модулем для роботи з бонусами.
Система бонусів	Vlogo.Bonuses.Application	Містить бізнес- логіку та роботу із сховищем даних. Використовується для обробки запитів згідно бізнес-правилам модуля.

Продовження таблиці 2.1

Система для роботи клієнтів	Vlogo.Clients.Host	Містить контролери, представлення для обробки запитів клієнтів. Також відповідальна за конфігурування веб-серверу для роботи клієнтів.
Система повідомлень	Vlogo.Notifications.Contracts	Відповідальна за контракти, що використовуються компонентом керування повідомленнями.
Система повідомлень	Vlogo.Notifications.Application	Містить логіку для надсилання повідомлень клієнтам та постачальникам.
Система керування замовленнями	Vlogo.Orders.Contracts	Містить контракти, що використовує система керування замовленнями та інші модулі для роботи з цією підсистемою

Продовження таблиці 2.1

Система керування замовленнями	Vlogo.Orders.Application	Відповідає за бізнес-логіку керування замовленнями.
Рейтингова система	Vlogo.Ratings.Contracts	Містить контракти для роботи з рейтинговою системою.
Рейтингова система	Vlogo.Ratings.Application	Відповідає за бізнес-правила обробки рейтингу.
Система рекомендацій	Vlogo.Recomendations.Application	Містить логіку для надання користувачам рекомендацій на основі попередніх покупок, бонусів та акцій.
Система рекомендацій	Vlogo.Recommendations.Contracts	Містить контракти для роботи з сервісом рекомендацій.

Продовження таблиці 2.1

Система управління постачальниками	Vlogo.Suppliers.Application	Містить сервіси для обробки запитів постачальників, для їх повноцінних роботи.
Система управління постачальниками	Vlogo.Suppliers.Contracts	Містить контракти для роботи із сервісом постачальників.
Система управління постачальниками	Vlogo.Suppliers.Host	Містить веб-хост, його конфігурацію, контролери для обробки вхідних запитів постачальників, представлення, що відображаються на веб-сайті та налаштування модуля.

2.2.4 Опис інструментів розробки

Для розробки системи використовується інтегрована середа розробки VisualStudio 2019 Community Edition. Для більш зручної роботи був встановлений додаток Resharper Ultimate, приклад зображений на рисунку 2.7.

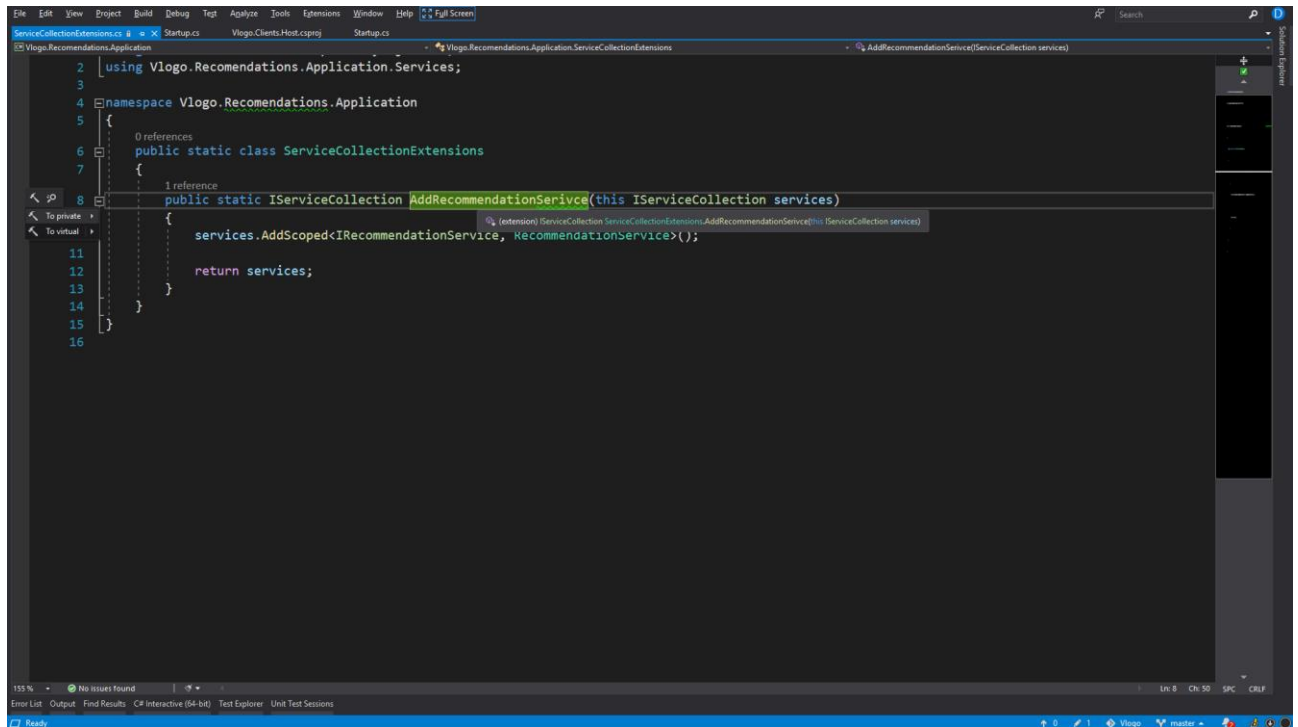


Рисунок 2.7 - Приклад середовища розробки

Для контролю версій використана утиліта git та віддалений сервер GitHub, приклад зображений на рисунку 2.8.

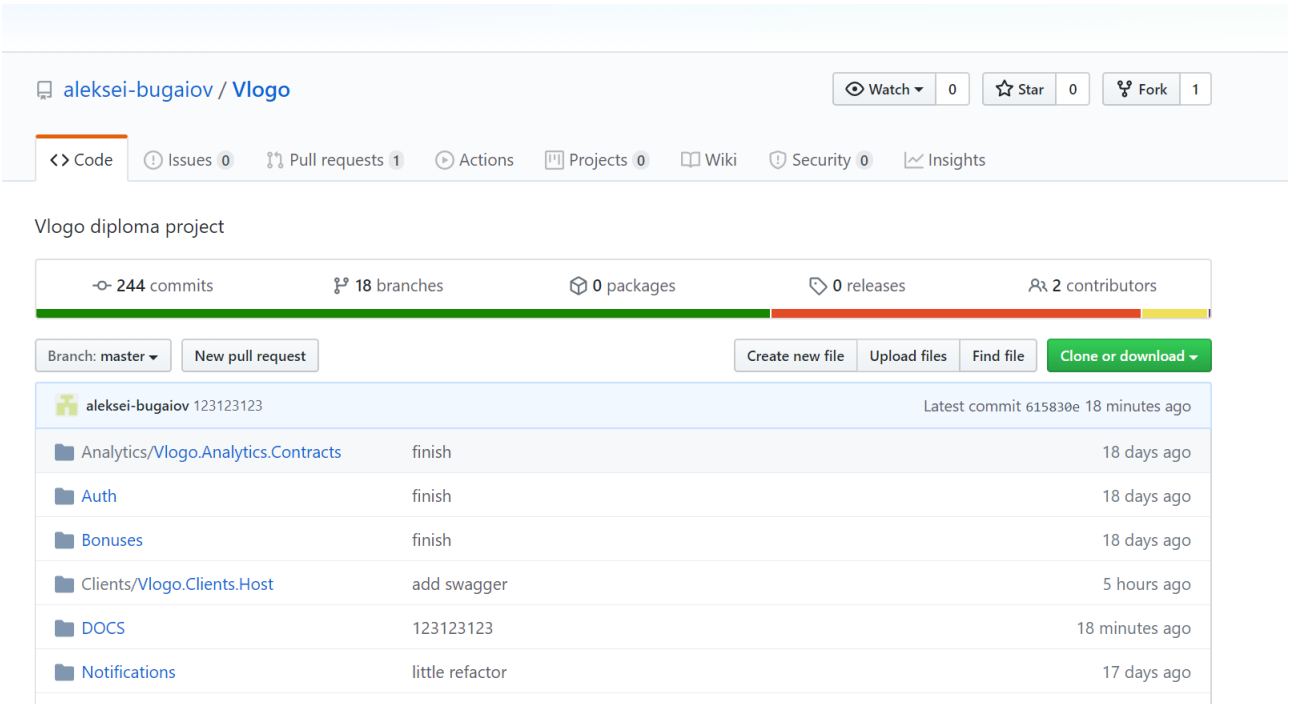


Рисунок 2.8 - Приклад GitHub

Для ведення історії та спрощення процесу, відмови від змін, використовується підхід, коли для кожного нового функціоналу створюється окрема гілка, приклад зображено на рисунку 2.9.

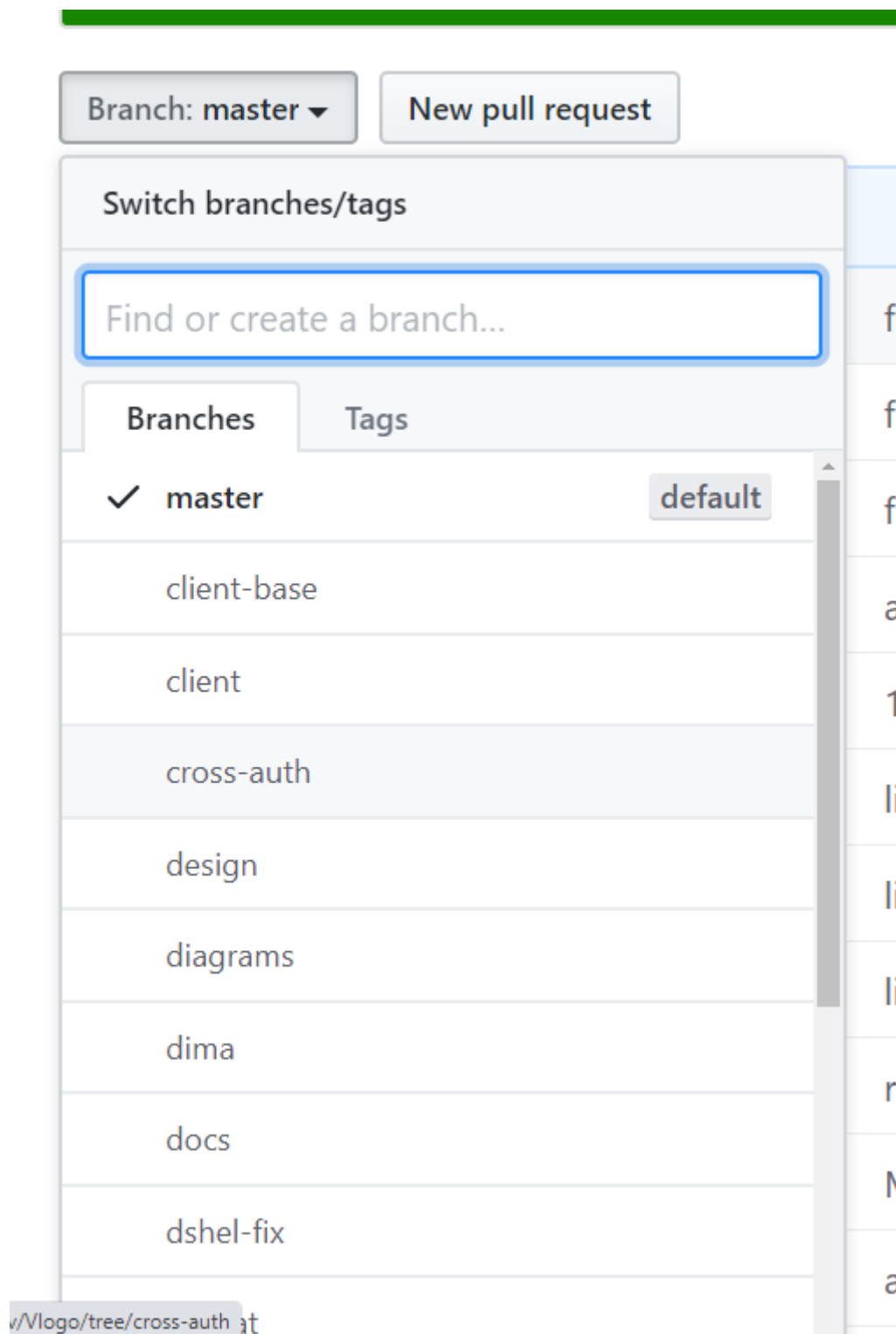


Рисунок 2.9 - Список гілок

Коли робота з новим функціоналом завершена, створюється запит на перенесення змін в головну гілку, як зображено на рисунку 2.10.

Create test #1

Edit Open with ▾

 Open toStars21 wants to merge 1 commit into `aleksei-bugaiov:master` from `toStars21:patch-1`

Conversation 0 Commits 1 Checks 0 Files changed 1

+1 -0

toStars21 commented 26 days ago Collaborator

No description provided.

Create test Verified 03bf78a

Add more commits by pushing to the `patch-1` branch on `toStars21/Vlogo`.

Checking for ability to merge automatically...
Hang in there while we check the branch's status.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Reviewers

No reviews

Still in progress? Convert to draft

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

Рисунок 2.10 - Приклад запиту на зміни

2.3 Висновки по розділу

У даному розділі в загальному вигляді було розглянуто архітектуру даного комплексу програм. За допомогою підходу мікро-сервісної архітектури ми отримали слабко зв'язані програмні модулі, які дуже легко масштабуються та розробляються, адже логіка кожного з даних модулів вийшла проста, проте в цілому дані компоненти формують досить складне програмне забезпечення, яке задовольняє поставленому технічному завданні.

При написання програмного коду для даних модулів широко використовувалися підходи SOLID, які є дуже популярними на сьогоднішній день, за допомогою даних підходів ми отримали код, який дуже просто зрозуміти, та легко модифікувати. Усі сервіси використовують REST підходи для проектування власних API. Що стосується інтерфейсу користувача, то для його реалізації було використано популярну зв'язку з HTML, CSS та JavaScript. Таким чином ми отримали динамічний та надійний веб інтерфейс, який має також можливість правильно відображатися на пристроях із різними діагоналями екранів.

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

3 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розгортання програмного забезпечення

Розгортання комплексу програмного забезпечення потребує наявності утиліт для контейнеризації, оркестрації та керування компонентами системи такі як: docker та docker-compose. В подальшому, ці утиліти будуть використані як для створення утиліти для розгортання додатку незалежно від операційної системи, так і для керування життям, станом та налаштуваннями кожного із компонентів комплексу програм. Також, для повноцінної роботи системи, що буде відповідати нефункціональним вимогам, повинні бути встановлені системи керування БД, такі як: MS SQL Server та MongoDB.

Для встановлення комплексу програм на комп'ютер, необхідно запустити спеціальну програму- інсталятор в залежності від операційної системи, що розгорне та відразу ж запустить систему. Під час встановлення, можна вибрати декілька опцій, що вплинуть на процес розгортання та подальшу швидкість роботи системи. Для тестування, рекомендується вибрати такі опції, автоматично піднімуть необхідні залежності, такі як: MS SQL Server, MongoDB та спростять процес розгортання завдяки зменшенню кількості необхідних залежностей на сторонні утиліти.

Для нормального функціонування програми, потрібно своєчасно запустити системи керування БД та під час встановлення системи, пропустити автоматичне розгортання.

3.2 Робота з програмним забезпеченням

Керівництво користувача описане у «Керівництво користувача».

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

В рамках даної роботи, був пройдений шлях створення готового та функціонально насиченого продукту, починаючи від етапу проектування та закінчуючи перевіркою виконання нефункціональних вимог.

Після формування ідеї дипломної роботи, був проведений детальний аналіз ринку. Це дозволило виділити низку схожих ідей та провести глибокий аналіз їх бізнесу. Головним та найпопулярнішим став такий додаток, як Glovo. Наразі – це одна із найпопулярніших служб онлайн замовлень в Україні. Після аналізу їх ринку, способу ведення бізнесу та ознайомлення із функціоналом, був виділений ряд переваг та недоліків, що в подальшому стало базою для створення функціональних вимог.

Наступним кроком, після появи функціональних вимог, став процес проектування системи, починаючи із створення діаграми варіантів використання продукту та закінчуючи діаграмою відносин сутностей в базі даних.

Маючи в арсеналі готову діаграму варіантів використання, був проведений сеанс проектування системи та вирішено створити комплекс програм, що матиме розподілений характер. Для цього функціонал було поділено на низку змістовно-завершених груп, що і лягли в основу комплексності та модульності системи. Таким чином, ми отримали ряд функціональних вимог до окремо взятих та слабо зв'язаних компонентів модульної системи. Маючи низку функціональних вимог та результати аналізу конкурентів, був поставлений ряд нефункціональних вимог, що дозволить досягти та перевершити продуктивність конкурентів. Кожен із компонентів був детально проаналізований за цільовим призначенням, для виявлення оптимального технологічного стеку, який забезпечить максимальну продуктивність та ефективність відповідно поставленим вимогам.

Після вибору платформи розробки для кожної підсистеми, постала задача проектування глобальної архітектури. Була розроблена та зображена діаграма взаємодії усіх підсистем між собою.

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

Наступним етапом, стало планування деталей реалізації кожної підсистеми. Під час даної фази було спроектовано низку діаграм для відображення внутрішньої роботи підсистеми із врахуванням її цільових задач та нефункціональних вимог.

Одним із найважливіших кроків стала програмна реалізація кожної підсистеми. Маючи в арсеналі низку діаграм, вибраний стек технологій та набір функціональних і нефункціональних вимог, був розпочатий етап розробки. На цьому етапі, було вирішені та узгоджені інфраструктурні проблеми. Для налагодження процесів командної роботи було прийнято рішення про використання ітеративних підходів до розробки програмного забезпечення. Також для поширення знань та підвищення якості створюваного продукту, був введений обов'язковий перегляд змін внесених у програмний код, будь-яким членом команди. Одним із ключових рішень для створення потужного інструменту контролю якості програмного забезпечення, була інтеграція із gitlab-ci. Що пришвидшило процес ідентифікації, локалізації та виключення проблем, спричинених внесенням нових змін.

Невід'ємною частиною реалізації програмного продукту став процес проведення тестування. Для підвищення гарантій збереження коректної роботи системи, було прийняте рішення про обов'язкове покриття кожного методу програмного коду низкою юніт тестів. Проте, процес проведення юніт тестування не гарантує збереження коректної функціональності продукту в цілому. Для вирішення цієї проблеми, був використаний підхід функціонального тестування. На основі функціональних вимог був створений ряд тест-кейсів, що лягли в основу ручного функціонального тестування програмного продукту. Даний процес проводився кожного разу, після внесення змістовно завершених змін хоча б до однієї із підсистем.

Розподілений характер продукту, став причиною низки проблем, що пов'язані із консистентністю та атомарністю розгортання системи. Для їх вирішення був обраний підхід, що давав би необхідні гарантії. Були вибрані

					КПІ.ІП-6304.045440.01.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		64

утиліти docker/docker-compose, що дозволили отримати простий та уніфікований спосіб розгортання комплексу програмного забезпечення, що потребує мінімальну кількість заздалегідь встановлених залежностей.

Результатом роботи, став готовий продукт, що відповідає усім поставленим вимогам, готовим до комерційного використання та є конкурентоспроможним на ринку України.

					КПІ.ІП-6304.045440.01.81	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК ПОСИЛАНЬ

- 1) Software development in Ukraine: 2019-2020 IT market report [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://www.n-ix.com/software-development-in-ukraine-2019-2020-market-report/>.
- 2) Workplace Automation is Everywhere [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://www.businessnewsdaily.com/9835-automation-tech-workforce.html>.
- 3) Исследование украинского Telegram [Електронний ресурс]. – 2019 – Режим доступу до ресурсу: <https://vctr.media/ukrainskiy-telegram-5000-15716/>.
- 4) China, WeChat, and the Origins of Chatbots [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: <https://chatbotsmagazine.com/china-wechat-and-the-origins-of-chatbots-89c481f15a44>.
- 5) Slack Statistics [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://expandedramblings.com/index.php/slack-statistics/>.
- 6) Microsoft Teams now has 75 million daily active users [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://www.businessinsider.com/microsoft-teams-hits-75-million-daily-active-users-2020-4>.
- 7) Project Management Digital Assistant [Електронний ресурс] – Режим доступу до ресурсу: <https://standuply.com/>.
- 8) Task management, Project Management and Calendar for Slack | Kyber [Електронний ресурс] – Режим доступу до ресурсу: <https://kyber.me/>.
- 9) Vacation Tracker | Slack App Directory [Електронний ресурс] – Режим доступу до ресурсу: <https://slack.com/apps/A1L3ZBM7G-vacation-tracker>.
- 10) Hubot is your friendly robot sidekick. [Електронний ресурс] – Режим доступу до ресурсу: <https://hubot.github.com/>.
- 11) Heartbeat (computing) [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Heartbeat_\(computing\)](https://en.wikipedia.org/wiki/Heartbeat_(computing)).

12) Amazon DynamoDB vs. Memcached vs. Redis [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://db-engines.com/en/system/Amazon+DynamoDB%3BMemcached%3BRedis>.

13) Amazon DynamoDB vs. CouchDB vs. MongoDB [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://db-engines.com/en/system/Amazon+DynamoDB%3BCouchDB%3BMongoDB>.

14) Introduction to ASGI: Emergence of an Async Python Web Ecosystem [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://florimond.dev/blog/articles/2019/08/introduction-to-asgi-async-python-web/>.

15) FastAPI [Електронний ресурс] – Режим доступу до ресурсу: <https://fastapi.tiangolo.com/>.

16) vasyukvv42/metabot: Modular Slack bot [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/vasyukvv42/metabot>.

17) Functional Programming HOWTO [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://docs.python.org/3/howto/functional.html>.

18) HMAC. // Practical UNIX and Internet Security, 3rd Edition / , 2011. – С. 187–194.

19) Verifying requests from Slack [Електронний ресурс] – Режим доступу до ресурсу: <https://api.slack.com/authentication/verifying-requests-from-slack>.

20) How to Write a Test Plan with the IEEE 829 Standard [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://reqtest.com/testing-blog/how-to-write-a-test-plan-2/>.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“ ____ ” _____ 2020 р.

КОМПЛЕКС ПРОГРАМ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ
ЗАМОВЛЕНЬ ТОВАРІВ ТА ПОСЛУГ (КОМПЛЕКСНА ТЕМА)

Програма та методика тестування

КП.ІІ-6329.045440.03.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ М.С Храмченко

Нормоконтроль:

_____ К.І. Ліщук

Виконали:

_____ О. О. Бугайов

_____ Д.М. Шелудько

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ.....	3
2 МЕТА ТЕСТУВАННЯ.....	4
3 МЕТОДИ ТЕСТУВАННЯ.....	5
4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

					КПІ.ІП-6329.045440.03.51	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Комплекс програм для управління та моніторингу замовлень товарів та послуг.

					КПІ.ІП-6329.045440.03.51	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

2 МЕТА ТЕСТУВАННЯ

Перевірити такі пункти:

- функціональна відповідність вимогам;
- коректність кожного ізольованого компонента;
- відповідність нефункціональним вимогам;
- безпека даних.

					КПІ.ІП-6329.045440.03.51	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується з використанням підходу Black Box. Проводиться перевірка коректної функціональності продукту. Проводиться наступні види тестів:

- функціональне тестування;
- критичне тестування;
- тестування навантаженням.

					КПІ.ІП-6329.045440.03.51	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію Visual Studio, NUnit, JMeter.

Коректність перевіряється завдяки:

- ручному тестуванню;
- юніт тестуванню;
- інтеграційного тестуванню;
- тестуванню навантаженнями.

					КПІ.ІП-6329.045440.03.51	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“ ____ ” _____ 2020 р.

КОМПЛЕКС ПРОГРАМ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ
ЗАМОВЛЕНЬ ТОВАРІВ ТА ПОСЛУГ (КОМПЛЕКСНА ТЕМА)

Керівництво користувача

КП.ІІ-6329.045440.02.34

“ПОГОДЖЕНО”

Керівник проекту:

Храмченко М.С.

Виконали:

Нормоконтроль:

К.І. Ліщук

Шелудько Д.М

Бугайов О.О.

Київ – 2020 року

Для початку роботи з програмним забезпеченням необхідно перейти по Інтернет посиланню <https://vlogo.com>. Після цього користувачу буде запропоновано пройти етап реєстрації, який можна побачити на рисунку 1.

Рисунок 1 - Реєстрація нового користувача

Зареєстровані користувачі, мають змогу пройти процес автентифікації/авторизації, який можна побачити на рисунку 2.

Рисунок 2 - Логін нового користувача

Після успішної авторизації користувача настає етап вибору версії додатку для подальшої роботи. В даному комплексі програмного забезпечення існує два окремих веб-додатки для роботи. Один призначений для клієнтів які планують робити онлайн замовлення, інший призначений для ведення бізнесу постачальниками.

Розглянемо веб-застосунок, призначений для роботи клієнтів. Першим що бачить клієнт при переході на головну сторінку додатку – це список усіх

					КПІ.ІП-6329.045440.02.34	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

продуктів, дана сторінка зображена на рисунку 3.

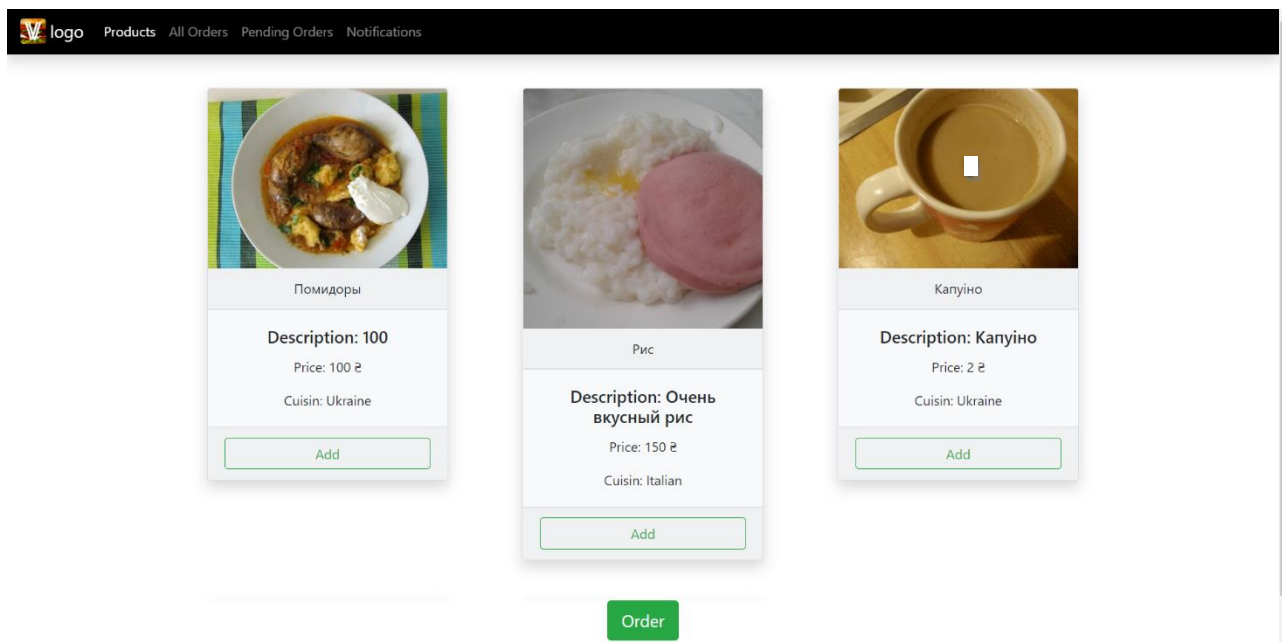


Рисунок 3 - Головна сторінка клієнта

Далі, користувач має можливість додати один або більше товарів, зображених на сторінці до кошику. Після цього для оформлення замовлення потрібно натиснути кнопку «Order» та прослідувати інструкціям з оплати даного замовлення. Після успішної оплати замовлення користувач отримує інформацію щодо даного замовлення на сторінці з повідомленнями, зображеними на рисунку 4.

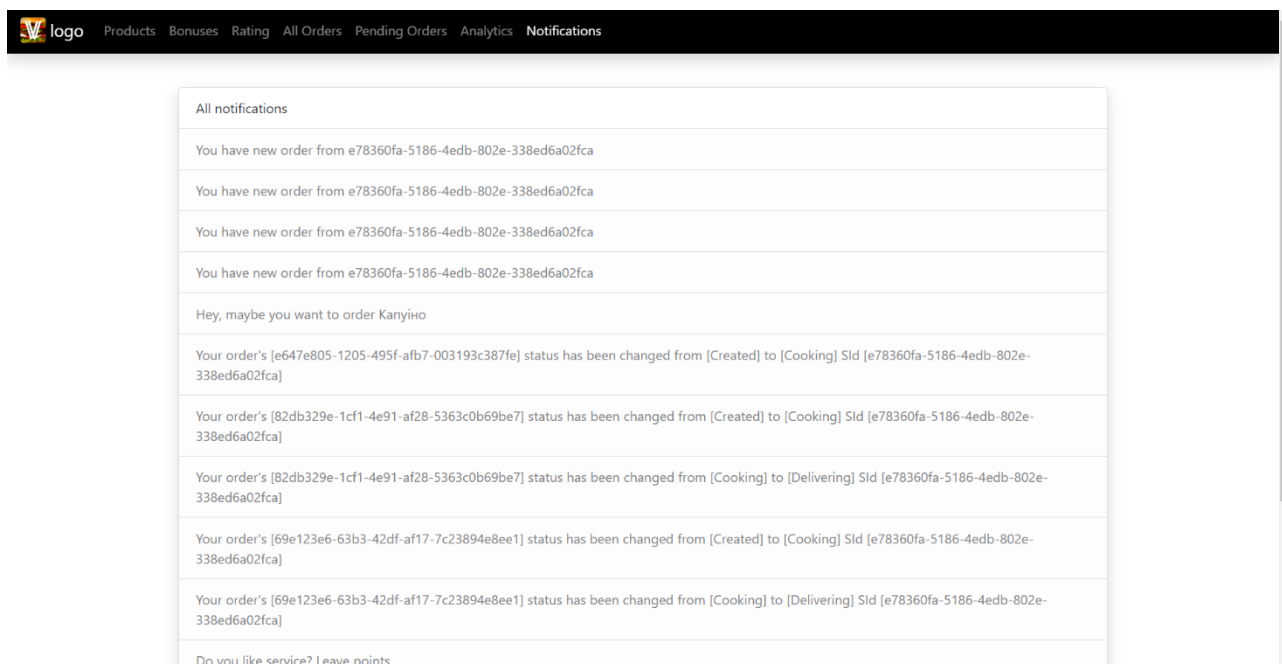


Рисунок 4 - Повідомлення користувача

					КП.ІП-6329.045440.02.34	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

Також користувач може слідкувати за статусом його замовлень у власному кабінеті, зображеного на рисунку 5.

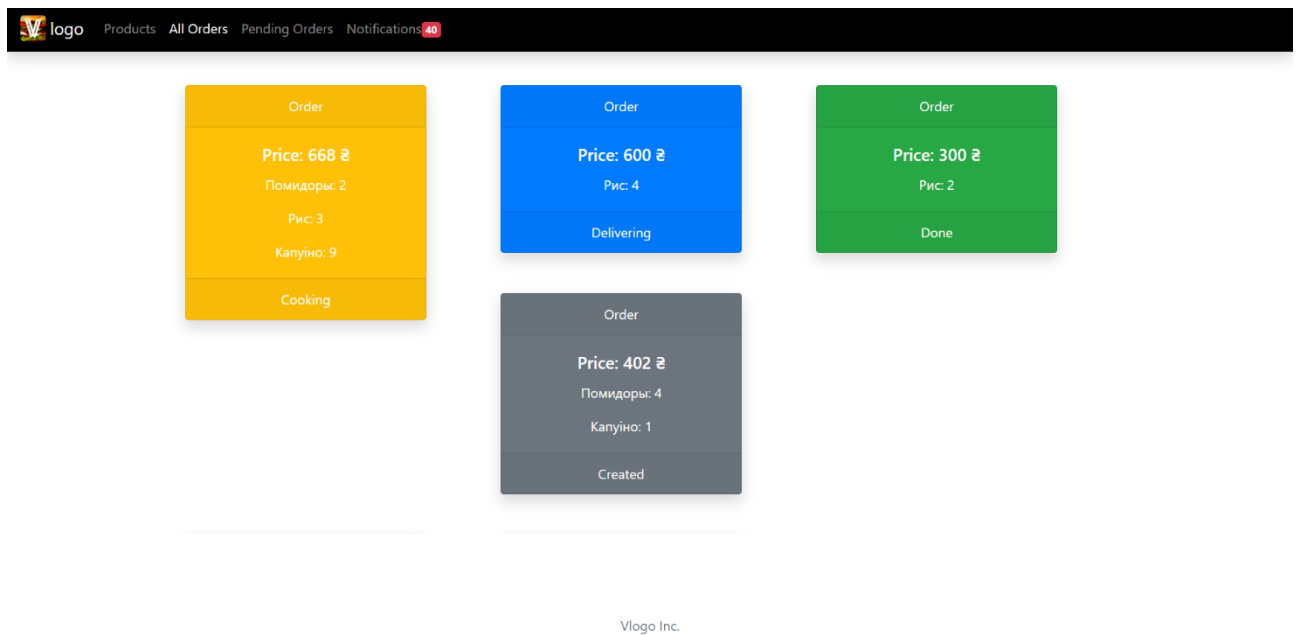


Рисунок 5 - Кабінет статусу замовлень

Тепер розглянемо додаток, призначений для роботи постачальників. При переході на головну сторінку даного додатку, постачальник потрапляє у меню створення власних товарів, зображене на рисунку 6.

The screenshot shows the 'Add ingredient' form in the Vlogo Inc. dashboard. The navigation bar includes 'logo', 'Products', 'Bonuses', 'Rating', 'All Orders', 'Pending Orders', 'Analytics', and 'Notifications'. The form has the following fields:

- Select a picture:** A placeholder for an image.
- Title:** A text input field with 'Salate' entered.
- Description:** A text input field with 'Very tasty and fresh' entered.
- Mass:** A text input field with '243' entered, and a unit selector set to 'gramm'.
- Price:** A text input field with '150' entered, and a unit selector set to 'grn'.
- Cuisin:** A dropdown menu with 'Please select' as the current selection.

At the bottom of the form, there are two buttons: 'Add ingredient' (green) and 'Save' (blue).

Рисунок 6 - Меню керування створенням замовлень

					КПІ.ІП-6329.045440.02.34	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

Після створення продукту, користувач може переглянути увесь асортимент товарів, зображений на рисунку 7.

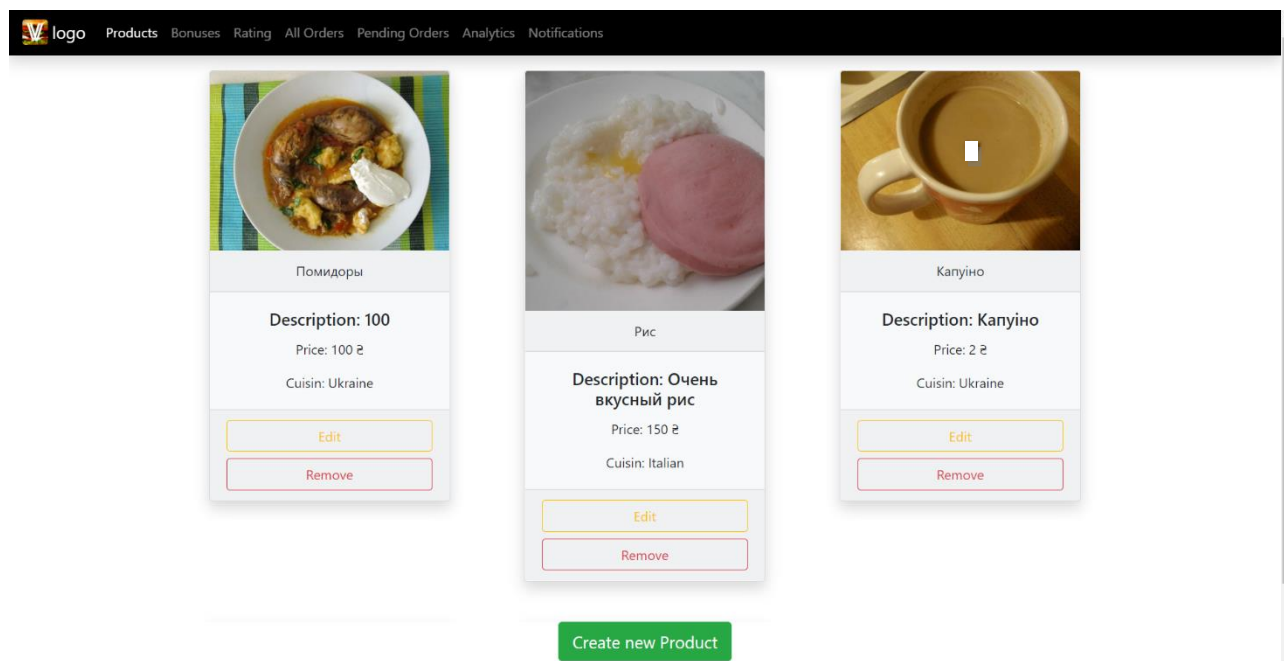


Рисунок 7 - Меню перегляду асортименту

Постачальник отримує повідомлення одразу після оформлення та успішної оплати замовлення клієнтом. Даний процес зображений на рисунку 8.

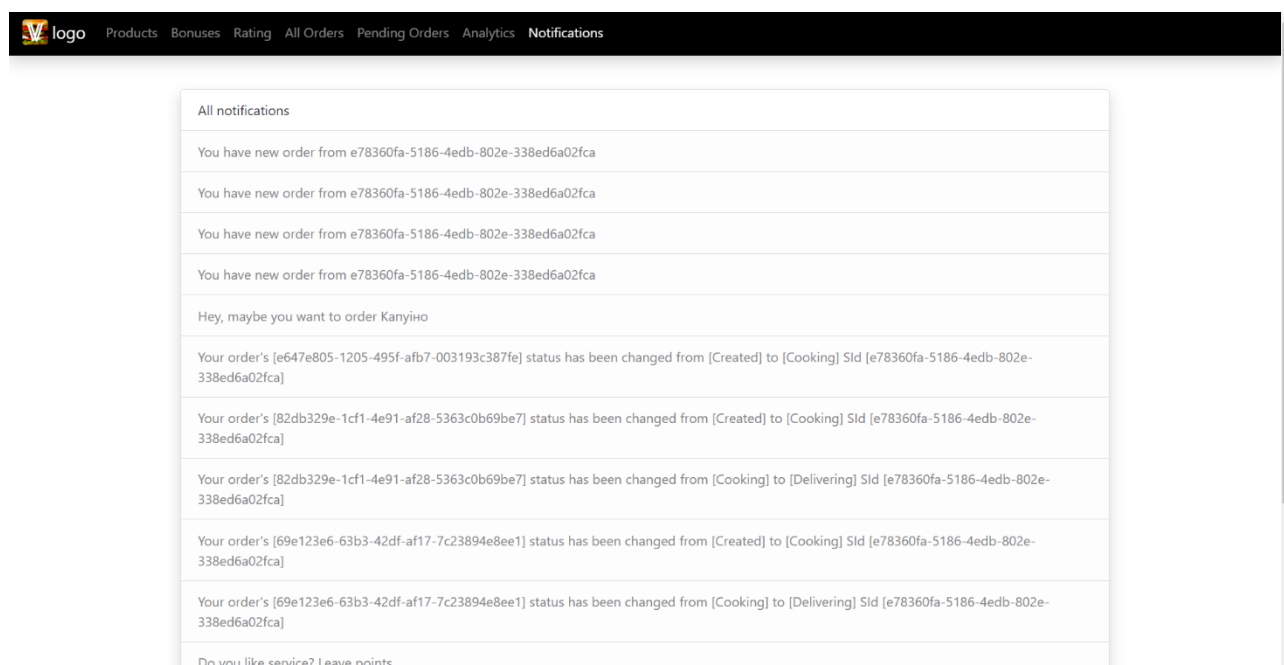


Рисунок 8 - Перегляд повідомлень постачальником

Також існує окреме меню керування замовленнями. В даному меню постачальник має можливість переглядати та змінювати статус активних

					КП.ІП-6329.045440.02.34	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

замовлень. Даний процес зображено на рисунку 9.

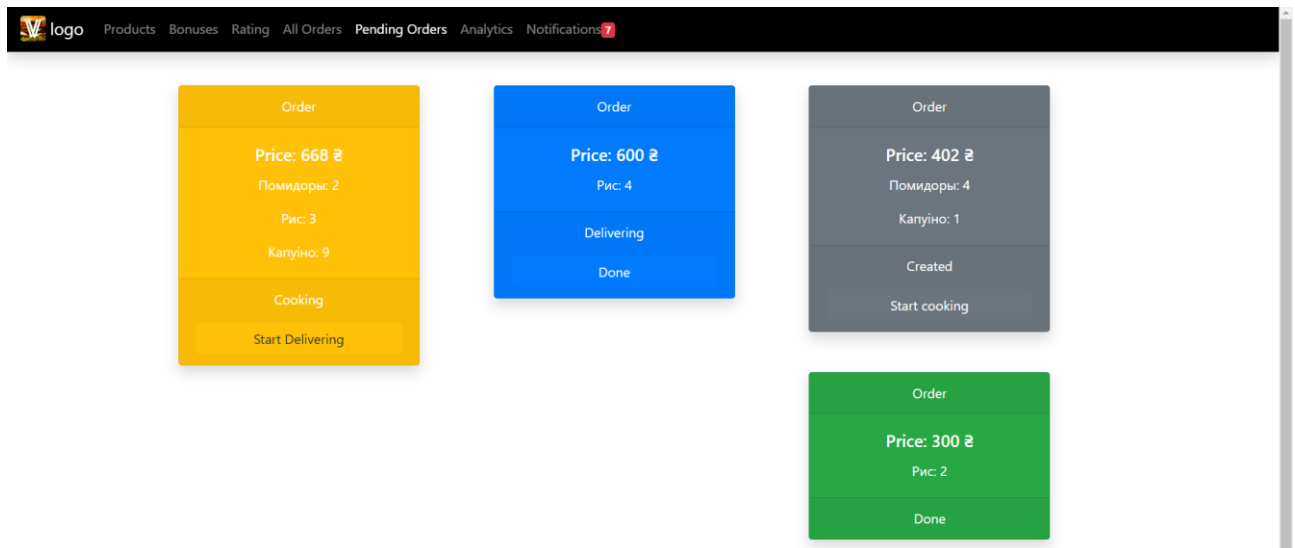


Рисунок 9 - Меню керування замовленнями

У разі наявності у постачальника завершених, або активних замовлень, даний постачальник має можливість переглянути персональну аналітику, зображену на рисунку 10.

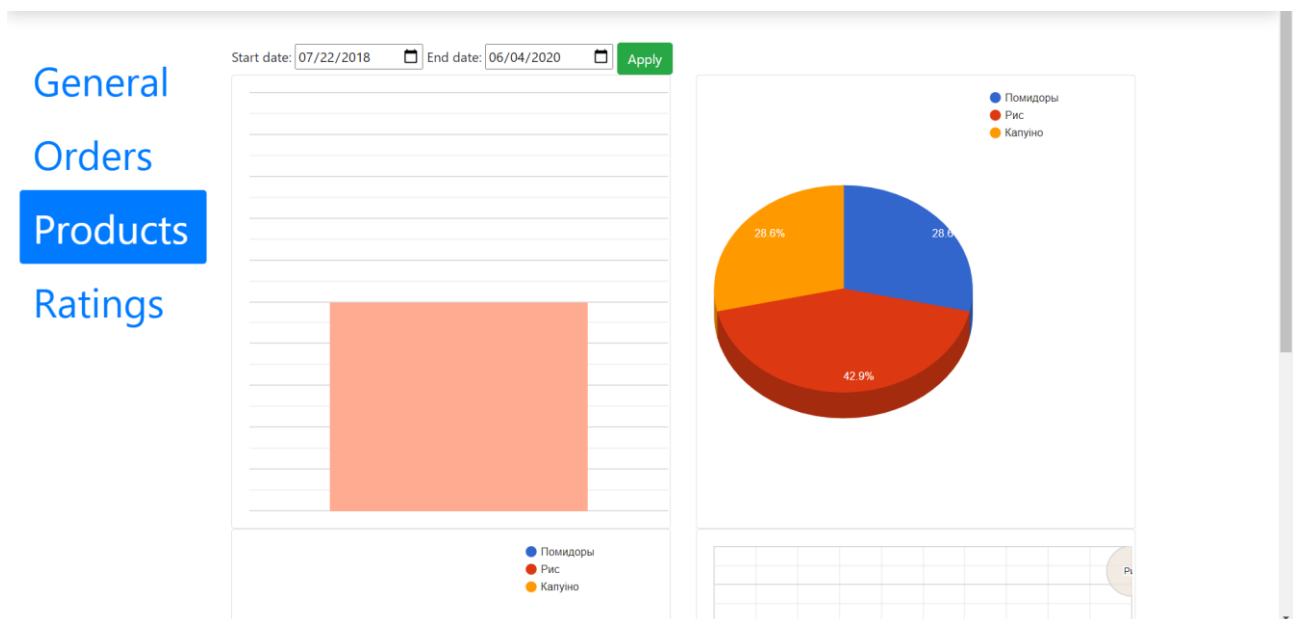


Рисунок 10 - Персональна аналітика

Також, будь-який постачальник може інтегрувати власну бонусну систему, для наявних активних товарів, для цього потрібно перейти на сторінку із бонусами,

та налаштувати їх, згідно своїх потреб. Даний процес зображено на рисунку 11.

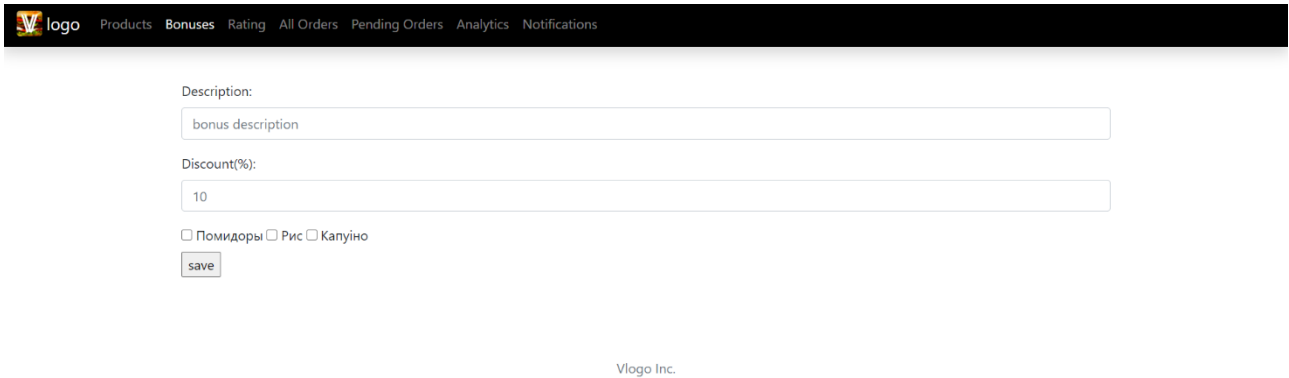


Рисунок 11 - Система бонусів

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“ ____ ” _____ 2020 р.

КОМПЛЕКС ПРОГРАМ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ
ЗАМОВЛЕНЬ ТОВАРІВ ТА ПОСЛУГ (КОМПЛЕКСНА ТЕМА)

Графічні матеріали

“ПОГОДЖЕНО”

Керівник проєкту:

М.С. Храмченко

Нормоконтроль:

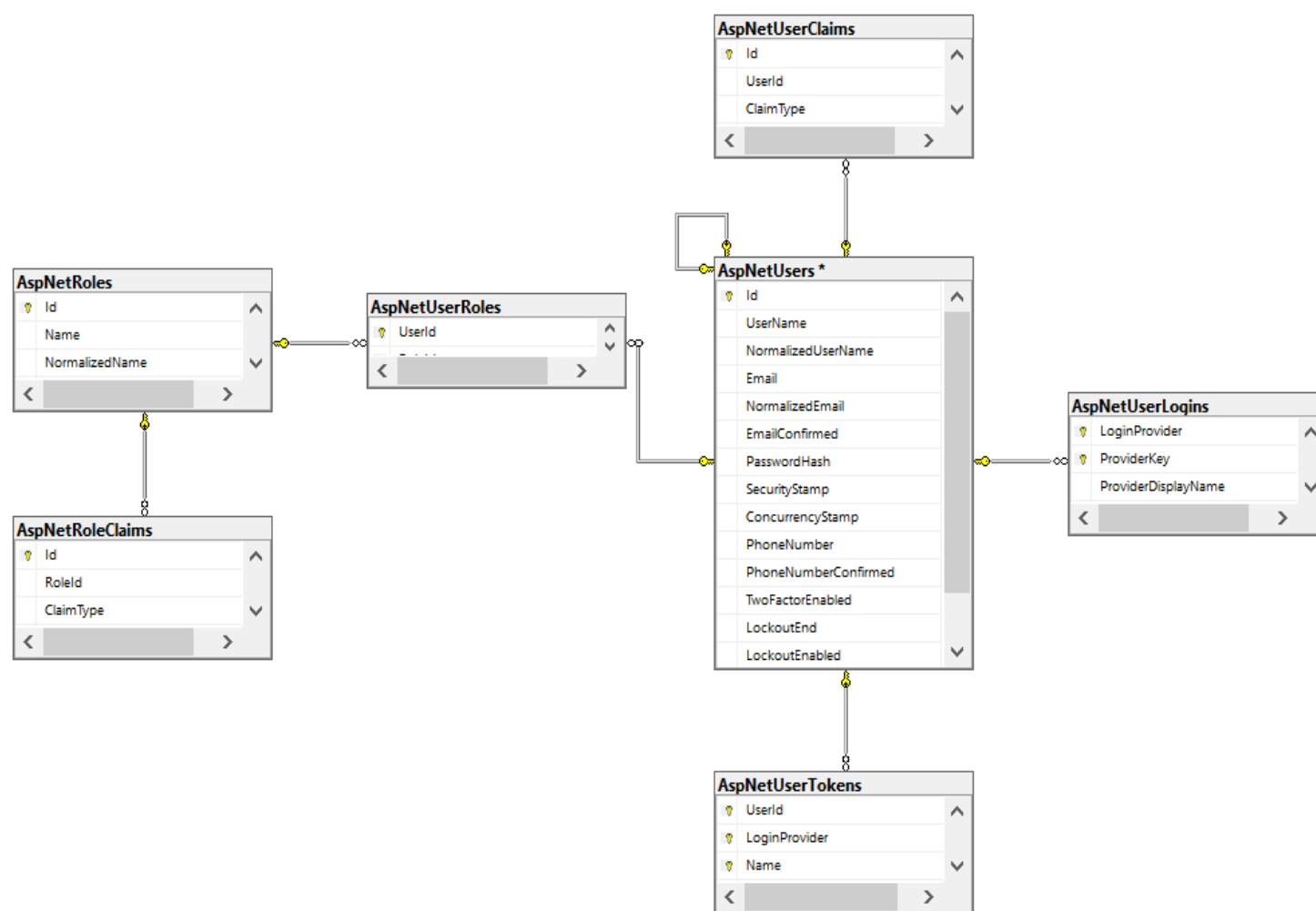
К.І. Лішук

Виконали:

О.О. Бугайов

Д.М. Шелудько

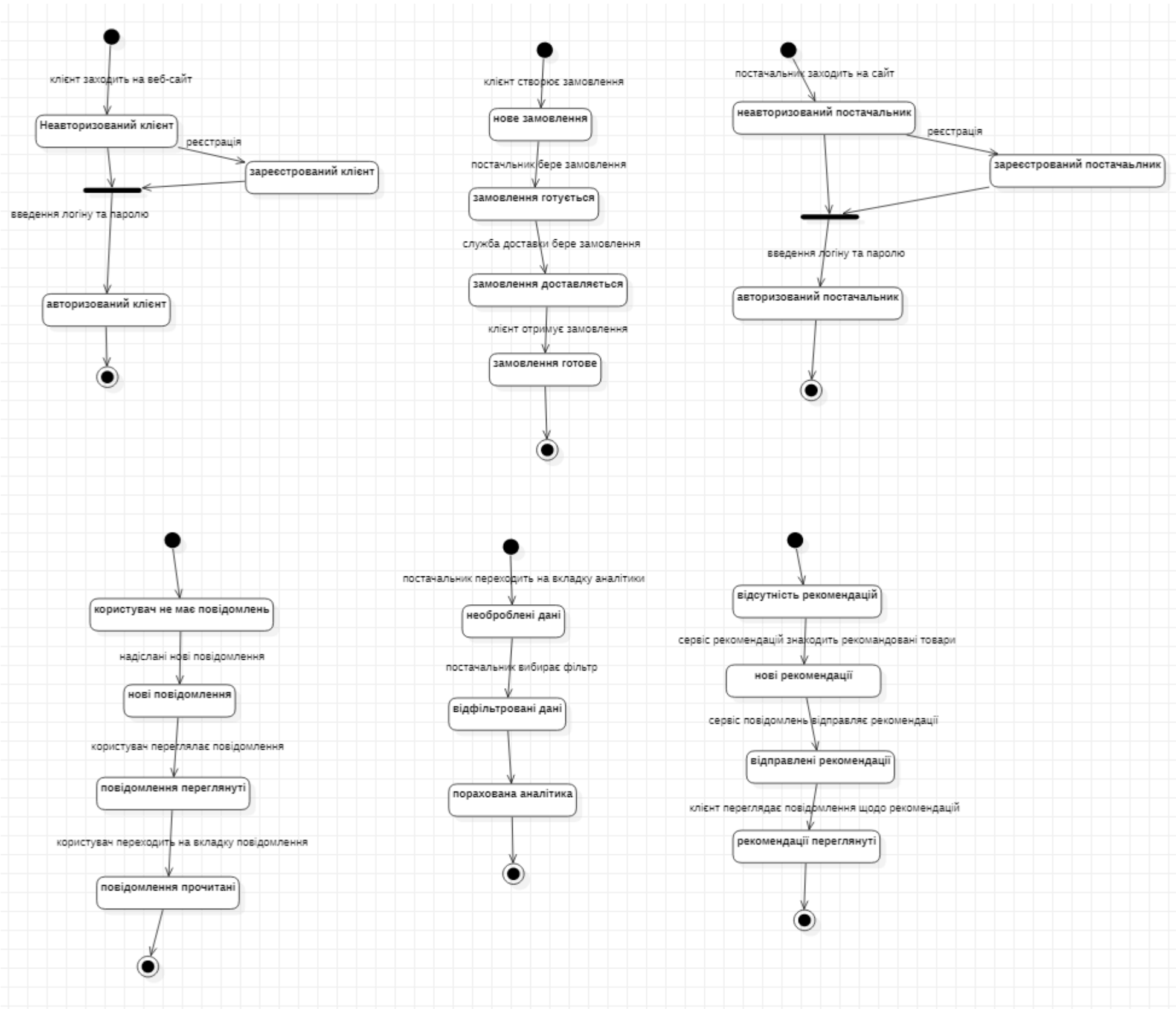
Київ – 2020 року



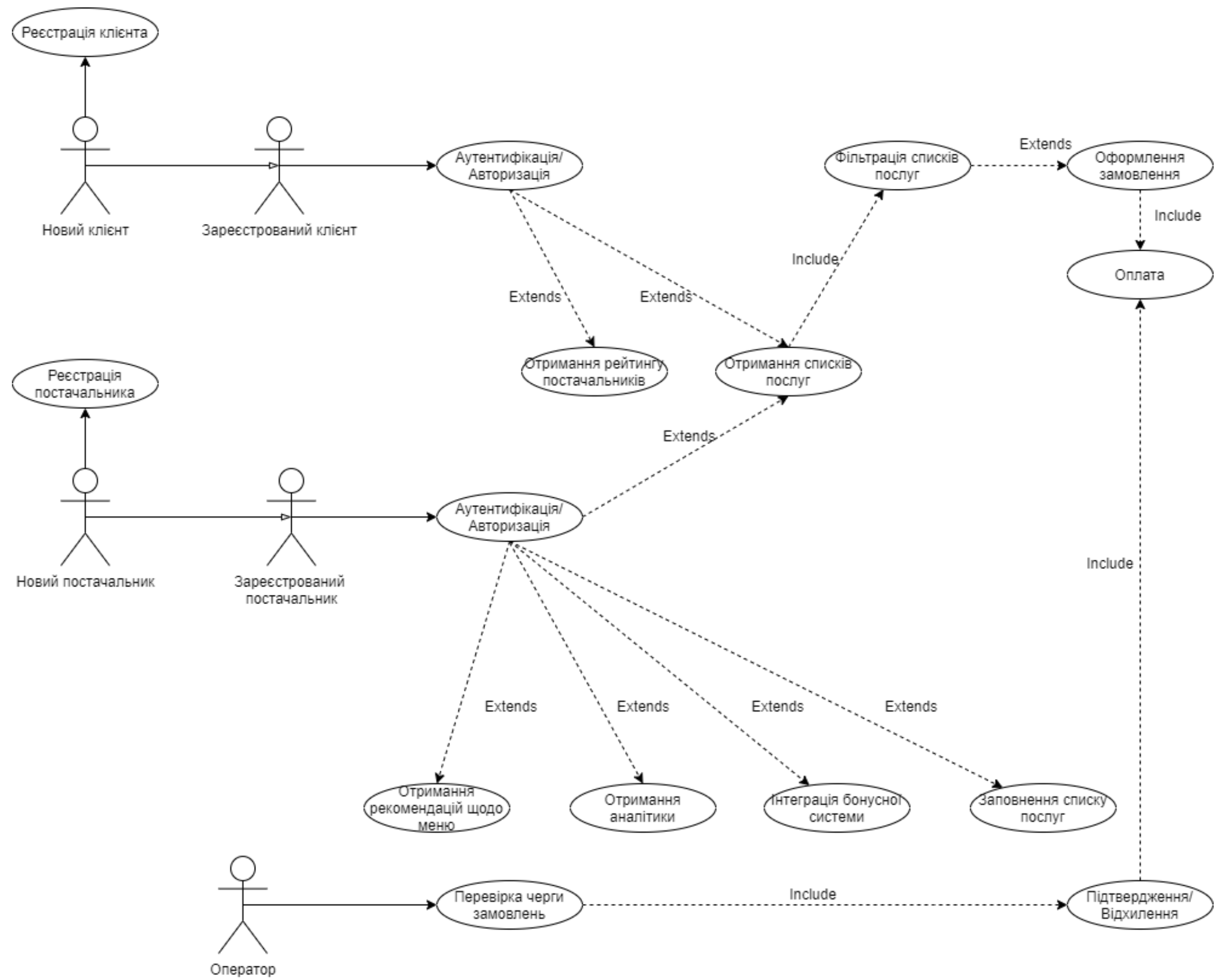
Key	Value	Type
▼ (1) LUUID("1593a353-048f-a54b-bbe2-bc869b758909")	{ 7 fields }	Object
_id	LUUID("1593a353-048f-a54b-bbe2-bc869b758909")	Legacy UUID
ClientId	LUUID("689a79d7-275e-f749-b431-eb2d9582389d")	Legacy UUID
SupplierId	LUUID("689a79d7-275e-f749-b431-eb2d9582389d")	Legacy UUID
Price	300.0	Double
> ProductId2Count	[1 element]	Array
Status	1	Int32
Time	2020-06-09 18:34:50.428Z	Date
Key	Value	Type
▼ (1) LUUID("0364a561-3805-fe40-b786-26adff73267c")	{ 7 fields }	Object
_id	LUUID("0364a561-3805-fe40-b786-26adff73267c")	Legacy UUID
Owner	LUUID("689a79d7-275e-f749-b431-eb2d9582389d")	Legacy UUID
Target	LUUID("689a79d7-275e-f749-b431-eb2d9582389d")	Legacy UUID
NotificationStatus	2	Int32
Message	You have new order from d7799a68-5e27-49f7-b431-eb2d9582389d	String
Uri	null	Null
UriDestinationDescription	null	Null
Key	Value	Type
▼ (1) LUUID("6c6a0b28-23c1-3648-9bbd-5d4b7b73b07c")	{ 4 fields }	Object
_id	LUUID("6c6a0b28-23c1-3648-9bbd-5d4b7b73b07c")	Legacy UUID
> ProductIds	[1 element]	Array
Description	Test	String
DiscountPercentage	0	Int32

Key	Value	Type
▼ (1) LUUID("5c46c2a7-00e1-b244-b289-492934060393")	{ 9 fields }	Object
_id	LUUID("5c46c2a7-00e1-b244-b289-492934060393")	Legacy UUID
SupplierId	LUUID("689a79d7-275e-f749-b431-eb2d9582389d")	Legacy UUID
Title	Test	String
Cuisin	0	Int32
Description	Test	String
> Ingredients	[0 elements]	Array
Mass	100.0	Double
Price	100.0	Double
PicBase64	null	Null

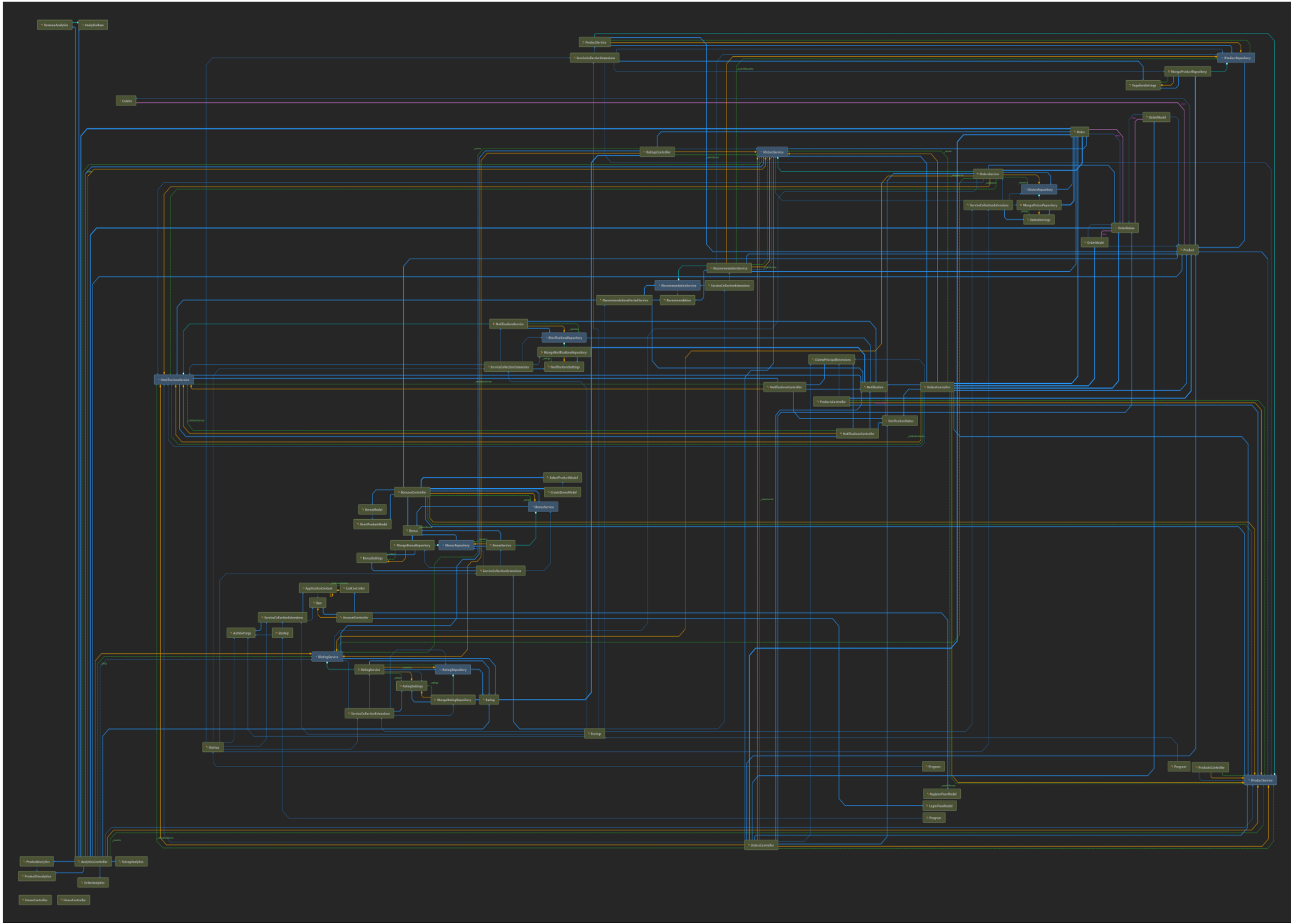
					КПІ.ІП-6329.045440.04.99.СБД			
					Схема бази даних	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Бугайов О.О., Шелудько Д.М.						
Перевірів		Храмченко М.С.						
Т. кон.					Комплекс програм для управління та моніторингу замовлень товарів та послуг (комплексна тема)	Аркуш		Аркушів
Н. кон.		Ліщук К.І.				КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Затвердив		Храмченко М.С.						



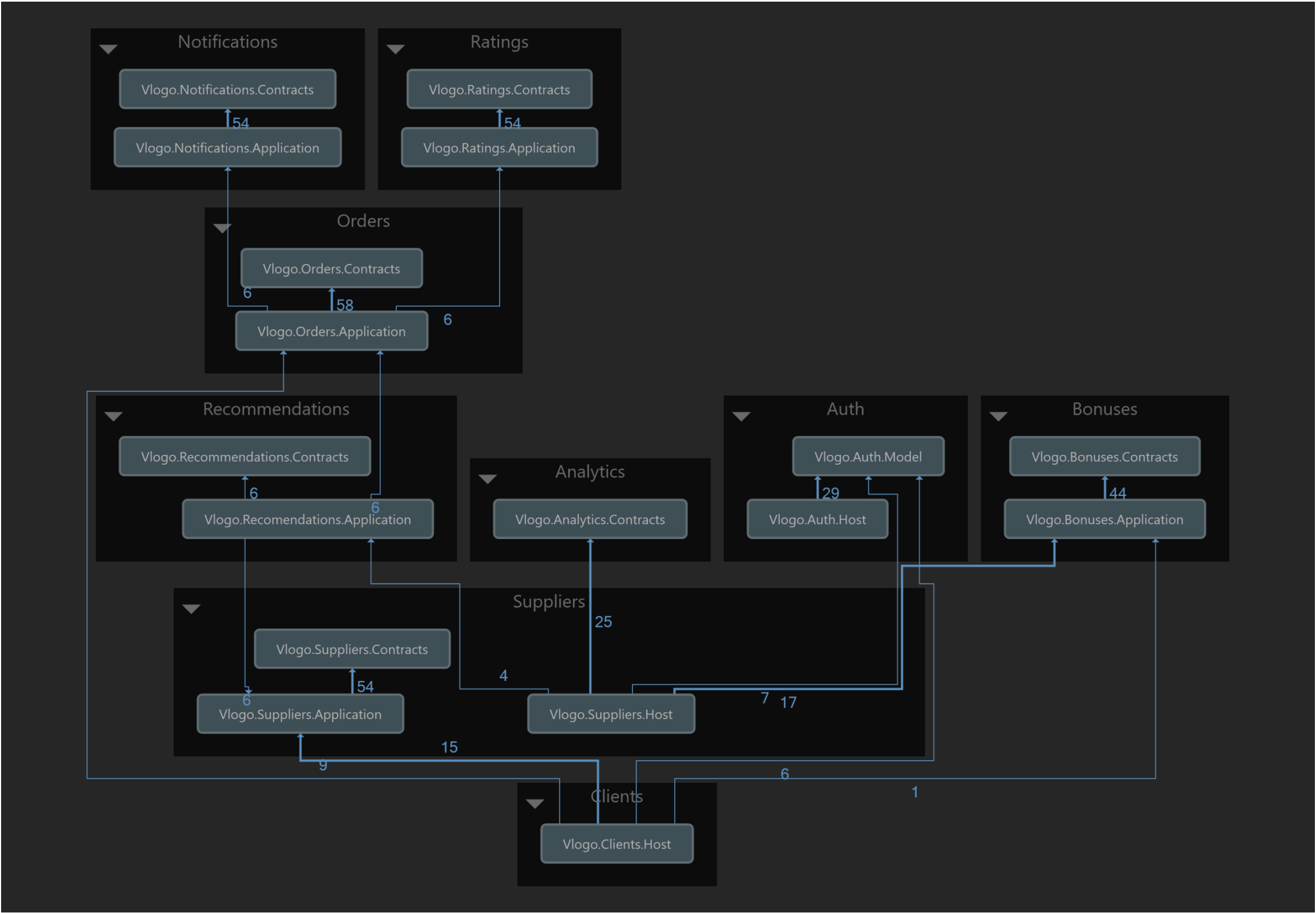
					КПІ.ІП-6329.045440.07.99.СС			
					Схема структурна станів системи			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Бугайов О.О. Шелудько Д.М						
Перевірів		Храмченко М.С.						
Т. кон.								
						Літера	Маса	Масштаб
						Аркуш		Аркушів
					Комплекс програм для управління та моніторингу замовлень товарів та послуг (комплексна тема)	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Н. кон.		Ліщук К.І.						
Затвердив		Храмченко М.С.						



					КПІ.ІП-6329.045440.05.99.СС					
					Схема структурна варіантів використань			Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата						
Розробив		Бугайов О.О, Шелудько Д.М								
Перевірів		Храмченко М.С.								
Т. кон.					Комплекс програм для управління та моніторингу замовлень товарів та послуг (комплексна тема)			Аркуш		Аркушів
Н. кон.		Ліщук К.І.								
Затвердив		Храмченко М.С.								
								КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		



					КПІ.ІП-6329.045440.06.99.СС					
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера			Маса	Масштаб
Розробив		Бугайов О.О. Шелудько Д.М								
Перевірив		Храмченко М.С.								
Т. кон.										
					Комплекс програм для управління та моніторингу замовлень товарів та послуг (комплексна тема)	Аркуш			Аркушів	
Н. кон.		Ліщук К.І.				КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63				
Затвердив		Храмченко М.С.								



					КПІ.ІП-6329.045440.08.99.СС						
						Схема структурна компонентів програмного забезпечення	Літера			Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Комплекс програм для управління та моніторингу замовлень товарів та послуг (комплексна тема)		Аркуш			Аркушів	
Розробив		Бугайов О.О, Шелудько Д.М									
Перевірів		Храмченко М.С.									
Т. кон.											
Н. кон.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63						
Затвердив		Храмченко М.С.									

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)
Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«До захисту допущено»

В.о. завідувача кафедри

Олександр ПАВЛОВ
(підпис) (ініціали, прізвище)

“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних
управляючих систем та технологій»
спеціальності «121 Інженерія програмного забезпечення»

*Комплекс програм для управління та моніторингу замовлень
товарів та послуг (комплексна тема). Частина для роботи з
постачальниками.*

на тему

Індивідуальна частина № 1

Виконав: студент IV курсу, групи ІП-63 Шелудько Дмитро
Максимович
(прізвище, ім'я, по батькові) (підпис)

Керівник ас. Каф. АСОІУ Храмченко М.С.
посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові (підпис)

Консультант з графічної документації
доц., к.т.н., Лішук К.І.
посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові (підпис)

Рецензент:
к.т.н. Доц каф. ТК, Лісовиченко Олег Іванович
посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові (підпис)

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

Студент _____
(підпис)

Київ – 2020 року

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1004053773

Дата перевірки:
15.06.2020 16:58:32 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2020 17:14:18 EEST

ID користувача:
77149

Назва документу: Scheludko_ind_ip63

ID файлу: 1004066681 Кількість сторінок: 39 Кількість слів: 5571 Кількість символів: 43059 Розмір файлу: 112.76 KB

1.88% Схожість

Найбільша схожість: 1.88% з джерело бібліотеки. ID файлу: 1004066692

0.88% Схожість з Інтернет джерелами

21

Page 41

1.88% Текстові збіги по Бібліотеці акаунту

46

Page 41

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів

11

АНОТАЦІЯ

Пояснювальна записка індивідуальної частини дипломного проекту складається з трьох розділів, містить 3 таблиці, 2 додатки та 17 джерел – загалом 101 сторінка.

Об’єкт дослідження: застосування, що можуть використовуватися для забезпечення онлайн замовлення та доставки товарів і послуг.

Мета дипломного проекту: спростити процес замовлення товарів та послуг в мережі Інтернет, поліпшити процес моніторингу за станом замовлення, розробити зручний спосіб доставки та отримання товарів та послуг, пришвидшити процес інтеграції постачальників товарів і послуг, розробити розумну бонусну систему для клієнтів, створити зручну систему аналітики для постачальників, створити рейтингову систему, розробити розумну та зручну систему повідомлення клієнтів

КЛЮЧОВІ СЛОВА: ДОСТАВКА, ТОВАРИ ТА ПОСЛУГИ, ІНТЕРНЕТ ЗАМОВЛЕННЯ, АНАЛІТИКА ЗАМОВЛЕНЬ

					КПІ.ІП-6329.045440.09.81	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ABSTRACT

Explanation note of the individual part of the diploma project is folded with three parts, include 3 tables, 2 attachments and 17 sources – in summary 101 pages.

The project is completed, and can be used for online procurement and delivery of goods and services.

The subject of the study is to be researched beforehand: to ensure that the online procurement and delivery of goods and services can be avoided.

Diploma`s purpose: Ask the process of conceiving goods and services in the measure of the Internet, improve process of monitoring behind the mill conceiving, disassemble a detailed description of the delivery and decommissioning of goods and services, see the process of integration suppliers goods and services, dissipate a reasonable bonus system for customers, create a strong analyst system for senior customers, create a rating system, dissipate a reasonable and strong notification system for customers.

KEYWORDS: DELIVERY, COMRADES AND SERVICES, INTERNET CONNECTION, ANALITICS CONNECTION

					КПІ.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОГО ПРОЕКТУ

на тему: Комплекс програм для управління та моніторингу замовлень товарів та послуг (комплексна тема). Частина для роботи з постачальниками

ЗМІСТ

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,
СКОРОЧЕНЬ І ТЕРМІНІВ 7**

**1 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ..... 9**

1.1	АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	9
1.1.1	Управління постачальниками	10
1.1.2	Інтеграція бонусної системи.....	15
1.1.3	Аналітика онлайн замовлень.....	17
1.1.4	Рейтингова система	18
1.1.5	Система керування замовленнями	19
1.1.6	Опис класів.....	20
1.2	АРХІТЕКТУРА БД	41
1.3	АНАЛІЗ БЕЗПЕКИ ДАНИХ.....	41
1.4	Висновки до розділу	42

**2 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО
ЗАБЕЗПЕЧЕННЯ..... 43**

2.1	ОПИС ПРОЦЕСІВ ТЕСТУВАННЯ	43
2.1.1	Юніт тестування	43
2.1.2	Функціональне тестування.....	44
2.2	ОПИС КОНТРОЛЬНОГО ПРИКЛАДУ	44
2.3	Висновки до розділу	50

**3 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО
ЗАБЕЗПЕЧЕННЯ..... 58**

3.1	РОЗГОРТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	58
3.2	РОБОТА З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	59
3.3	Висновки до розділу	59

ВИСНОВКИ..... 61

ПЕРЕЛІК ПОСИЛАНЬ..... 63

					КПІ.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface – набір чітко визначених методів для взаємодії різних компонентів ПЗ.

JSON – JavaScript Object Notation – текстовий формат серіалізації даних.

REST – Representational State Transfer – підхід до архітектури мережеских додатків та API.

HTTP – Hypertext Transfer Protocol – протокол прикладного рівня для передачі веб-сторінок та інших даних через мережу.

MIT (ліцензія) – група ліцензій для розповсюдження програмного забезпечення з відкритим програмним кодом.

NoSQL – тип баз даних, що відрізняється від SQL-баз даних переважно відсутністю реляційних зв'язків.

BSON – Binary JSON – бінарний формат серіалізації даних, що використовується в MongoDB для збереження документів.

TDD – Test-Driven Development – підхід до розробки програмного забезпечення, при якому розробка починається з попереднього написання тестів, після чого розробляється код, який має пройти ці тести.

Розгортання – процес встановлення програми на комп'ютер

MITM – Man in the middle - термін в криптографії, що позначає ситуацію, коли атакуючий здатний читати та видозмінювати за своєю волею повідомлення, якими обмінюються кореспонденти, причому жоден з останніх не може здогадатися про його присутність в каналі.

Горизонтальне масштабування – процес зміни кількості паралельно запущених програм

DI – Dependency injection - проектування програмного забезпечення, при якому підключення зовнішніх залежностей відбувається автоматично

Скрипт – частина програмного коду, зазвичай малого розміру

					КП.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

IDE – Interactive development environment - інтегроване середовище розробки

NUGET – засіб пошуку та інсталяції програмних залежностей в середовищі .NET

Framework – поняття, що описує готовий програмний продукт, на базі якого можна вести розробку

SWAGGER – програмний засіб, за допомогою якого можна автоматично згенерувати опис усіх точок доступу

GIT – утиліта за допомогою якої відбувається побудова системи версій для програмного коду

					КПІ.ІП-6329.045440.09.81	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

1 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Архітектура програмного забезпечення

В рамках даної роботи була розроблена розподілена багатокomпонентна система, що складається із дев'яти модулів. Кожен із модулів є незалежним від інших та може їх використовувати за допомогою інтерфейсу взаємодії, що надає сервіс. У цілях збереження однорідності та схожості модулів, кожен із них було поділено на декілька обов'язкових частин(проєктів), таких, як:

- контракти – це проєкт, що містить вхідні та вихідні контракти сервісу. Вони використовуються для взаємодії інших компонентів із сервісом;
- хостингу – це проєкт, що по своїй суті є точкою входу в програму. Він призначений для налаштування та запуску веб-сервісу;
- рівень додатків – це проєкт, що інкапсулює всю бізнес логіку та використовується хостингом для опрацювання запитів користувачів.

Кожен компонент системи має можливість комунікувати з іншими компонентами. Для цього йому достатньо використовувати контракти сервісу, з яким він хоче взаємодіяти. На рисунку 1.1 зображена схема структурна взаємодії компонентів системи між собою.

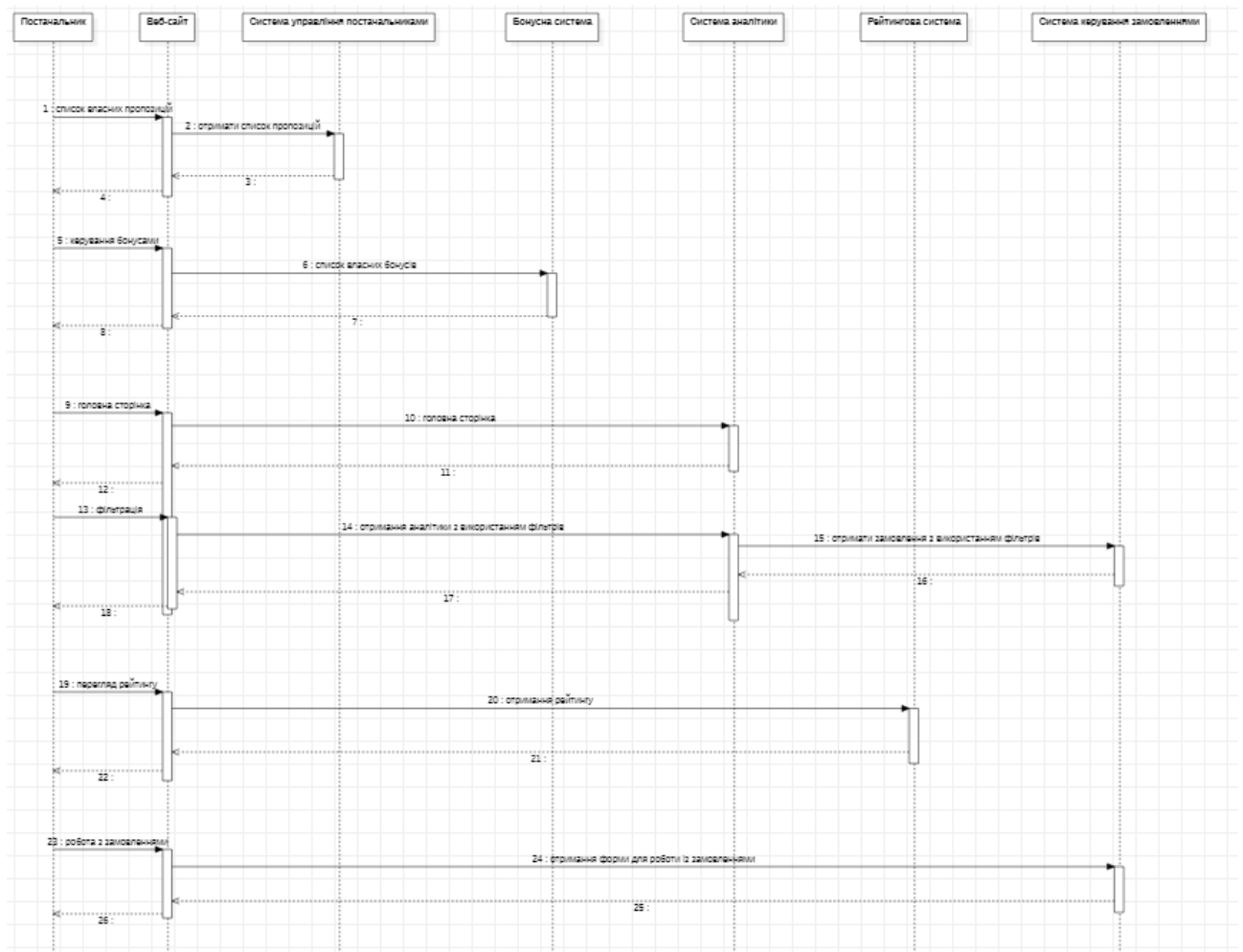


Рисунок 1.1 – Схема структурна взаємодії компонентів системи

1.1.1 Управління постачальниками

Даний компонент надає базовий функціонал по роботі з постачальниками та їх послугами. Він дає можливість створювати, редагувати та видаляти товари та послуги.

Підсистема управління постачальниками складається з двох компонентів: клієнтська частина(веб-сайт) та серверна частина.

Серверна частина реалізована з використанням REST-підходів до архітектурного стилю. Завдяки своїй REST-натурі, сервіс не зберігає жодної інформації про стан клієнта, що забезпечую високу надійність. Також завдяки цьому забезпечується високий рівень горизонтального та вертикального масштабування серверної частини.

Змн.	Арк.	№ докум.	Підпис	Дата

Для реалізації було обрано платформу Asp.Net Core, та одна із її варіацій Asp.Net Core MVC, що дозволило в зручному форматі розділити застосування на 3 частини: моделі, представлення та контролера. Також були виділені спеціальні контракти, що використовуються іншими компонентами для взаємодії з підсистемою управління постачальниками. Завдяки незалежності від операційної системи, компонент можна запускати як на Unix-подібних системах так і на Windows. Для зберігання даних була вибрана система керування базами даних MongoDB. Це дозволить в майбутньому скористатися легкістю горизонтального масштабування БД для забезпечення стабільної роботи та пришвидшення відповіді серверної частини.

Для взаємодії інших компонентів із серверною частиною, використовується формат даних JSON та протокол взаємодії HTTP, а для взаємодії веб-сайта(клієнтської частини) із сервером, дані відправляються у вигляді спеціальних форм, що серіалізуються в відповідні об'єкти автоматично.

Було прийняте рішення, щодо підключення бібліотеки Swagger до проекту управління постачальниками. Задяки цьоу, з'явилося описання підсистеми, у вигляді OpenAPI специфікації. Кожен клієнт, що має намір взаємодіяти із даним компонентом, має можливість переглянути інтерфейс взаємодії та випробувати його власноруч.

Список всіх доступних ендпоінтів веб-інтерфейсу системи керування постачальниками зображена на рисунку 1.2.

Products		▼
GET	/Products	
GET	/Products/create	
POST	/Products/create	
GET	/Products/edit/{id}	
POST	/Products/edit/{id}	
POST	/Products/addIngredient	
POST	/Products/removeIngredient/{index}	
GET	/Products/delete/{id}	

Рисунок 1.2 – Список доступних ендпоінтів

Розглянемо декілька ендпоінтів більш детально.

На рисунку 1.3 зображений веб-інтерфейс взаємодії для редагування продуктів.

					КПІ.ІП-6329.045440.09.81	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

POST `/Products/edit/{id}`

Parameters

Name	Description
id * required string (path)	id
Id string (query)	Id
SupplierId string (query)	SupplierId
Title string (query)	Title
Cuisin integer (query)	Available values : 0, 1, 2, 3 --
Description string (query)	Description
Ingredients array[string] (query)	
Mass number (query)	Mass
Price number (query)	Price
PicBase64 string (query)	PicBase64

Рисунок 1.3 - Метод редагування продукту

На рисунку 1.4 зображений метод видалення продуктів. На рисунку можна побачити, що Swagger дозволяє виконати тестовий запит із сайту, це дуже зручно

для тестування. Також він описує процес роботи із сервісом у зручній та стандартизованій формі, це дозволяє зменшити кількість документації. Також видно результат та статус коди, який повертає сервіс. Це дозволяє використовувати модуль максимально безпечно та захищено.

GET /Products/delete/{id}	
Parameters	
Name	Description
id * required string (path)	id
Responses	
Code	Description
200	Success

Рисунок 1.4 - Метод видалення продуктів

Модуль керування постачальниками складається із трьох основних частин, що взаємодіють один з одним. Схемструктурну взаємодії зображено на рисунку 1.5.

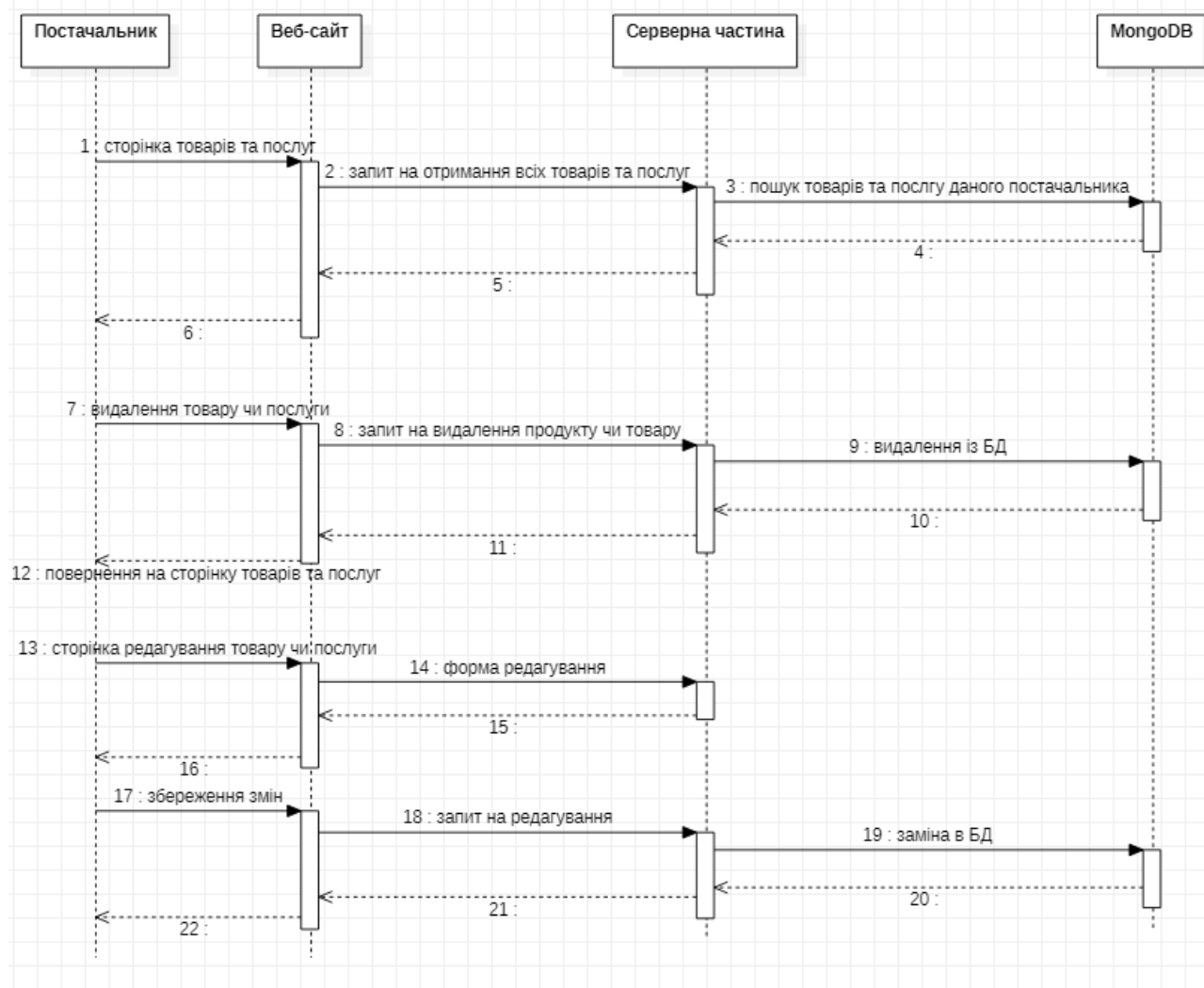


Рисунок 1.5 – Схема структурна взаємодії постачальника з компонентом керування постачальниками

1.1.2 Інтеграція бонусної системи

Система для інтеграції бонусів постачальників, дає можливість додавати, оновлювати існуючі та видаляти бонусні програми. Дана підсистема складається з двох компонентів: серверної та клієнтської.

Для реалізації даної підсистеми, була вибрана платформа Asp.Net Core MVC. Завдяки цьому сервіс поділено на 3 частини: моделі, представлення та контролери.

Для розробки клієнтської частини(веб-сайту), були використані Razor views, що автоматично підтримуються платформою. Це дозволило спростити передачу даних та взаємодію між контролерами та представленнями.

Як сховище даних була використана MongoDB та бібліотека MongoDB Driver, що дозволило спростити роботу та комунікації зі сховищем. Такий вибір дозволить зробити систему стійкою до стихійних лих та неочікуваних збоїв на серверах, де запущена підсистема.

Компонент для роботи із бонусами складається із трьох частин, а також використовує модуль керування постачальниками під час своєї роботи. Схема структурна взаємодії елементів модуля зображено на рисунку 1.6.

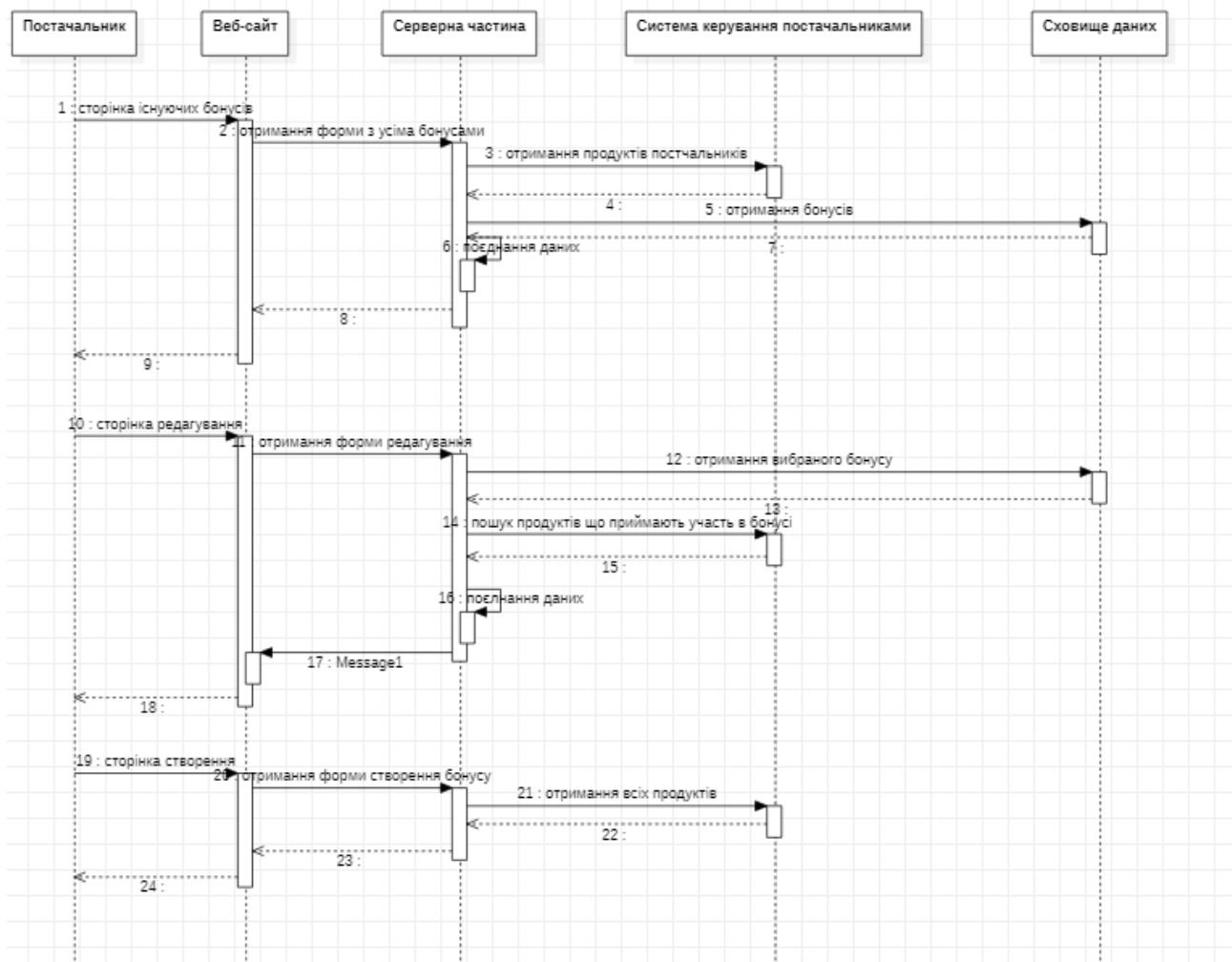


Рисунок 1.6 – Схема структурна взаємодії постачальників з компонентом керування бонусами

Модуль для інтеграції бонусної системи використовує бібліотеку Swagger та стандарт OpenAPI для документації своїх ендпоінтів. Список всіх ендпоінтів зображено на рисунку 1.7.

Bonuses	
GET	/Bonuses
GET	/Bonuses/create
POST	/Bonuses/create
GET	/Bonuses/edit/{id}
POST	/Bonuses/edit/{id}
POST	/Bonuses/delete/{id}
POST	/Bonuses/shortProducts/all

Рисунок 1.7 - Список ендпоінтів модуля бонусів

1.1.3 Аналітика онлайн замовлень

Сервіс аналітики є дуже важливим компонентом. Його основна задача – це отримання, обробка та повернення на клієнтську частину великої кількості даних в зручній для відображення формі. Оскільки найважливіше для підприємців – це отримати дані якомога швидше та отримати їх навіть у випадку непрацездатності серверних компонентів системи, він має легко масштабуватися для підтримки підвищеної активності та буде толерантним до збоїв.

Для реалізації серверної частини була вибрана платформа Asp.Net Core MVC. Вона задовольняє всім вимогам, бо побудована таким чином, що легко масштабується як горизонтально, так і вертикально. Завдяки масштабуванню,

					КПІ.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

сервіс є толерантним до збоїв, оскільки завжди можна створити 2 та більше копій підсистеми. Також завдяки архітектурі системи, компонент аналітики легко може отримати дані з них та у випадку мережевих затримок, виставити час очікування відповіді, після вичерпання якого, модуль буде вважатися непрацездатним.

Клієнтська частина створена на основі Razor View, JavaScript, Bootstrap, HTML, CSS, що дозволяє виводити аналітичні дані у зручному для перегляду форматі.

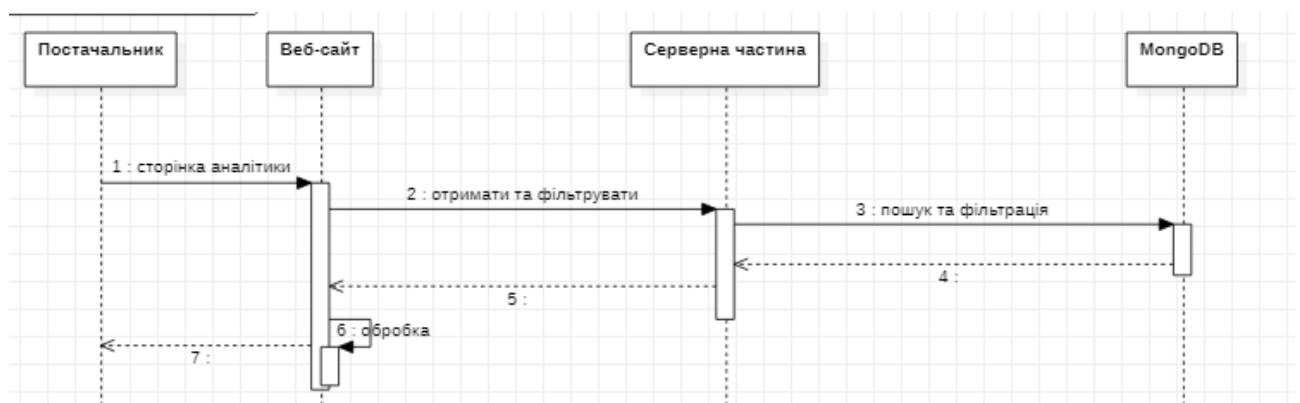


Рисунок 1.8 – Схема структурна взаємодії постачальника з компонентом аналітики

1.1.4 Рейтингова система

Основна мета даного компонента – це забезпечити дуплексний зв’язок між клієнтом та постачальником, щоб клієнти могли залишати свої відгуки та ставити оцінку сервісу, а постачальники могли переглядати відгуки, оцінки та виправляти свої помилки і недоліки в обслуговуванні.

Цей модуль реалізовано за допомогою платформи Asp.Net MVC та нереляційної БД MongoDB. Клієнти залишають відгуки на веб-сайті, що реалізовано з використанням Bootstrap, Razor Views, JavaScript, HTML, CSS. Дані з веб-сайту відправляються у вигляді спеціальних форм на серверну частину, де серіалізуються в об’єкти мови програмування та за допомогою MongoDB Driver зберігаються в сховище даних.

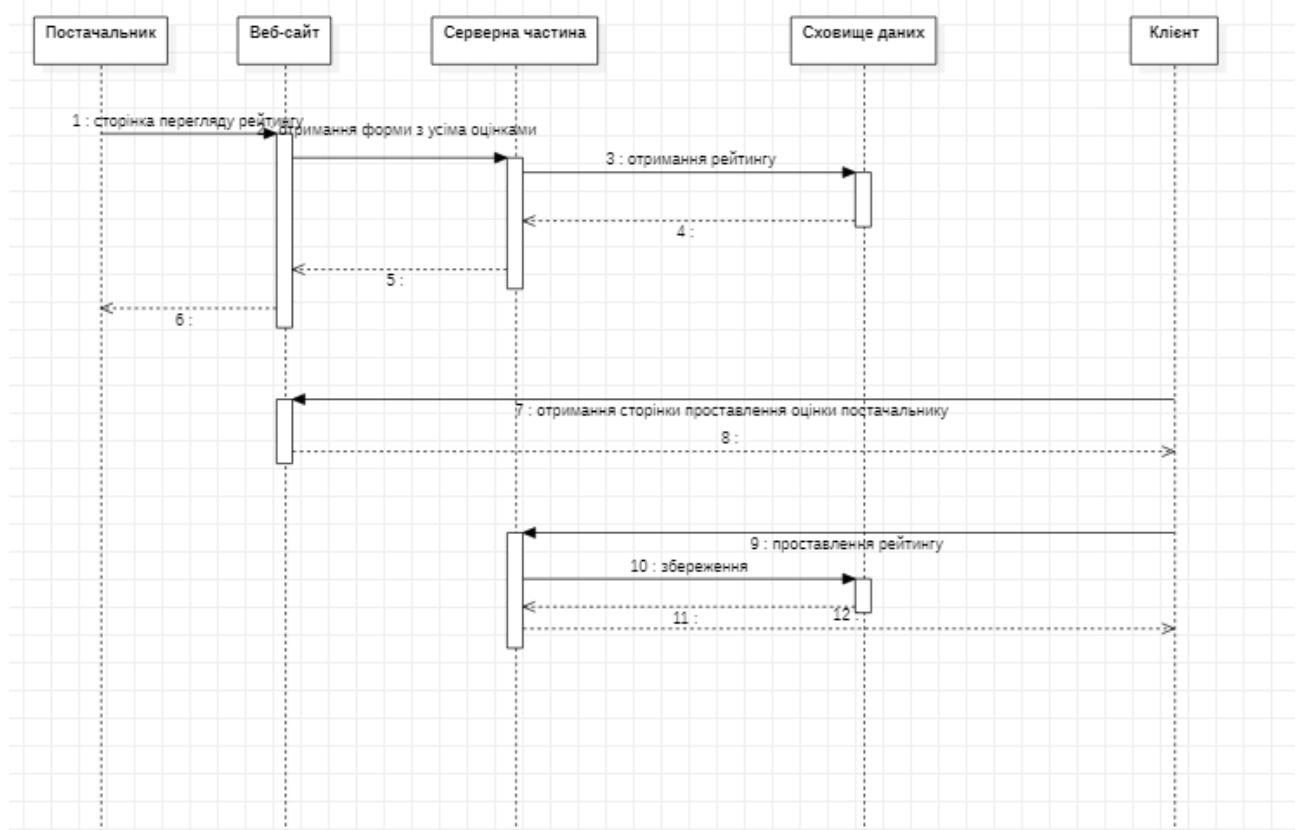


Рисунок 1.9 – Схема структурна взаємодії постачальника та клієнта з рейтинговою системою

1.1.5 Система керування замовленнями

Дана підсистема є зв'язувальною між двома іншими компонентами: системою керування постачальниками та системою керування замовленнями. Головна ціль даного компонента – це створити уніфікований інструмент для керування замовленнями. В залежності від ролі користувача: клієнт чи постачальник, у нього будуть різні права. Наприклад, клієнт має права на створення замовлень, а постачальник, що отримав замовлення, на його подальшу обробку та модифікацію станів.

Для реалізації вибрана платформа Asp.Net Core MVC та використано інший компонент системи, а саме модуль уніфікованої реєстрації та авторизації, що керувати ролями та розрізняти запити постачальників від запитів клієнтів.

Змн.	Арк.	№ докум.	Підпис	Дата

Компонент використовує нереляційну базу даних, MongoDB та MongoDB Driver як ORM-бібліотеку для спрощення комунікацій зі сховищем даних.

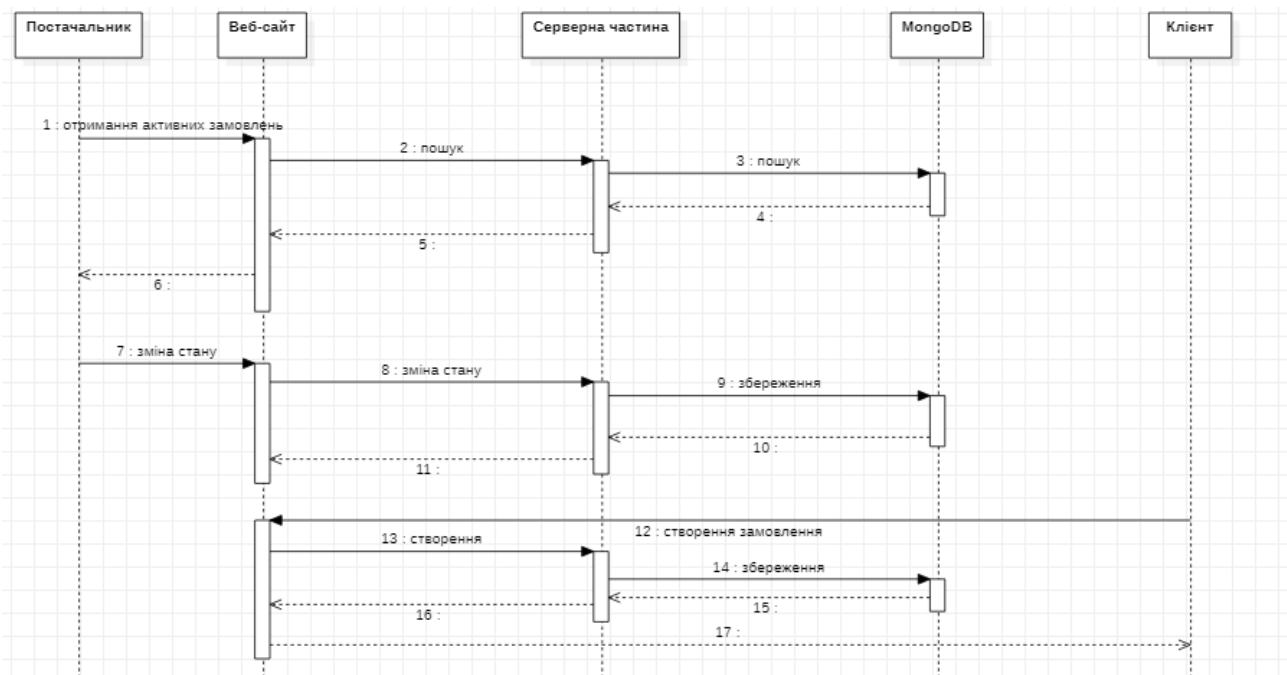


Рисунок 1.10 – Схема структурна взаємодії постачальника та клієнта з системою керування замовленнями

1.1.6 Опис класів

Комплекс програм складається із 9 модулів, що у свою чергу поділені на декілька проєктів із бізнес-логікою, контрактами, сервісами, хостінгом, тощо. Кожен із цих проєктів містить специфічні класи, сутності, сервіси та конфігурації, що призначені для вузьконаправлених цілей та можуть використовуватися іншими модулями.

У кожного сервісу існує обов’язковий проєкт із контрактами, що призначений для використання іншими модулями для взаємодії із сервісом та проєкт із бізнес-логікою, що відповідає за обробку запитів використовуючі бізнес-правила та контракти. У кожному із цих проєктів знаходяться однотипні класи, сутності та сервіси, незалежно від модуля. Розглянемо детальний опис класів, сутностей та їх відносин описано в таблиці 1.1

Таблиця 1.1 - Опис класів у системі

Назва модуля	Назва проєкту	Назва сутності	Опис
Управління постачальниками	Vlogo.Suppliers.C ontracts	Cuisins	Відображає доступні кухні світу. Використовується при описанні продуктів.
Управління постачальниками	Vlogo.Suppliers.C ontracts	Product	Відображає продукт у системі. Використовується компонентами керування замовленнями, управління постачальниками та управління клієнтами.

Продовження таблиці 1.1

Управління постачальника ми	Vlogo.Suppliers.Applicat ion	IProductReposito ry	Інтерфейс, що по своїй суті являється шаблоном проектування репозиторій та надає рівень абстракції над сховищем даних продуктів. Він використовуєть ся сервісом продуктів для роботи з базою даних.
-----------------------------------	---------------------------------	------------------------	--

Продовження таблиці 1.1

Управління постачальника ми	Vlogo.Suppliers.Applicat ion	MongoProductReposit ory	Реалізація шаблону проектуван ня репозиторій , що використов ує для збереження даних нереляційн у базу даних MongoDB. Дозволяє зберегти, Знайти, видалити, змінити продукти.
Управління постачальника ми	Vlogo.Suppliers.Applicat ion	IProductService	Абстракція сервісу для роботи з продуктами ..

Продовження таблиці 1.1

Управління постачальниками	Vlogo.Suppliers.Application	ProductService	Реалізація інтерфейсу IProductService. Використовує репозиторій продуктів для роботи із даними, тобто їх збереження , пошук, зміна та видалення
Управління постачальниками	Vlogo.Suppliers.Application	SuppliersSettings	Клас, що потрібен для маніпулювання налаштуваннями модуля, такими, як підключення до бази даних, назва бази даних, колекцій та таблиць.

Продовження таблиці 1.1

Управління постачальниками	Vlogo.Suppliers.Application	ServiceCollectionExtensions	Статичний клас, що дозволяє налаштувати та під'єднати певні залежності для подальшого їх інжектування системою Asp.Net DI.
Управління постачальниками	Vlogo.Suppliers.Host	ProductsController	Контролер, призначений для роботи із продуктами. Він дозволяє додавати, змінювати, видаляти продукти із компонента.
Управління постачальниками	Vlogo.Suppliers.Host	All	Клас призначений для відображення на сайті списку усіх продуктів
Управління постачальниками	Vlogo.Suppliers.Host	Create	Клас призначений для зображення форми створення нового продукту

Продовження таблиці 1.1

Управління постачальниками	Vlogo.Suppliers.Host	Edit	Клас призначений для відображення форми редагування продукту
Управління постачальниками	Vlogo.Suppliers.Host	Program	Клас, що є точкою входу в програму. Він налаштовує веб- сервер та запускає його.
Управління постачальниками	Vlogo.Suppliers.Host	Startup	Клас, що призначений для налаштування залежностей для роботи веб- серверу, конфігурування самого веб- серверу.
Управління постачальниками	Vlogo.Suppliers.Host	_Layout	Клас призначений для налаштування представлень, що відображаються клієнту на веб- сайті.

Продовження таблиці 1.1

Бонусна система	Vlogo.Bonuses.Contracts	Bonus	Клас, що представляє собою бонус в системі та використовуються в компонентах керування постачальниками, управління користувачами та керування замовленнями.
Бонусна система	Vlogo.Bonuses.Application	IBonusRepository	Абстракція над сховищем даних, де знаходяться бонуси.
Бонусна система	Vlogo.Bonuses.Application	MongoBonusRepository	Клас, що реалізує абстракцію для збереження, пошуку, редагування та видаленню бонусів із сховища даних MongoDB.
Бонусна система	Vlogo.Bonuses.Application	IBonusService	Абстракція для роботи із бонусами.

Продовження таблиці 1.1

Бонусна система	Vlogo.Bonuses.Application	BonusService	Реалізація сервісу для роботи із бонусами. Має залежність на репозиторій бонусів.
Бонусна система	Vlogo.Bonuses.Application	BonusSettings	Клас, що потрібен для маніпулювання налаштуваннями модуля для роботи з бонусами, такими, як підключення до бази даних, назва бази даних, колекцій та таблиць.
Бонусна система	Vlogo.Bonuses.Application	ServiceCollectionExtensions	Статичний клас, що дозволяє підключити всі залежності для роботи модуля бонусів у систему інжектування залежностей.

Продовження таблиці 1.1

Бонусна система	Vlogo.Suppliers.Host	BonusesController	Контролер, призначений для обробки запитів для роботи із бонусами. Дозволяє шукати, створювати, видаляти та редагувати бонуси.
Бонусна система	Vlogo.Suppliers.Host	BonusModel	Клас, що використовується для передачі у представлення та інкапсулює в собі частину даних про бонус.
Бонусна система	Vlogo.Suppliers.Host	CreateBonusModel	Клас, що представляє собою модель для створення бонуса. Його використовують представлення та контролер.

Продовження таблиці 1.1

Бонусна система	Vlogo.Suppliers.Host	SelectProductModel	Клас, що потрібен для можливості динамічного включення та вилучення продуктів із бонуса.
Бонусна система	Vlogo.Suppliers.Host	ShortProductModel	Клас, що представляє собою зменшений опис продукту.
Бонусна система	Vlogo.Suppliers.Host	All	Клас призначений для відображення списку бонусів.
Бонусна система	Vlogo.Suppliers.Host	Create	Клас призначений для зображення форми для створення бонуса на веб-сайті.
Бонусна система	Vlogo.Suppliers.Host	Edit	Клас призначений для зображення форми для редагування бонуса на веб-сайті.

Продовження таблиці 1.1

Система аналітики онлайн замовлень	Vlogo.Analytics.Contracts	AnalyticsBase	Базовий абстрактний клас для всіх аналітичних даних. Призначений для виділення спільного та запобігання дублювання коду.
Система аналітики онлайн замовлень	Vlogo.Analytics.Contracts	OrderAnalytics	Клас, що містить аналітичні дані на основі попередніх замовлень постачальника.
Система аналітики онлайн замовлень	Vlogo.Analytics.Contracts	ProductDescription	Клас, що описує продукт, використовується як допоміжний клас сервісом аналітики.
Система аналітики онлайн замовлень	Vlogo.Analytics.Contracts	ProductAnalytics	Клас, що інкапсулює аналітичні дані на основі існуючих продуктів певного постачальника.

Продовження таблиці 1.1

Система аналітики онлайн замовлень	Vlogo.Analytics.Contracts	RatingAnalytics	Клас, що відображає аналітичні дані щодо популярності, оцінки та рейтингу постачальника на основі його попередніх замовлень.
Система аналітики онлайн замовлень	Vlogo.Analytics.Contracts	RevenueAnalytics	Клас, що інкаспулює в собі аналітичні дані щодо доходу постачальника за певний період.
Система аналітики онлайн замовлень	Vlogo.Suppliers.Host	AnalyticsController	Контролер, що потрібен для отримання аналітичних даних та їх фільтрації.
Система аналітики онлайн замовлень	Vlogo.Suppliers.Host	Get	Клас, що потрібен для візуалізації аналітичних даних на веб- сайті.

Продовження таблиці 1.1

Система аналітики онлайн замовлень	Vlogo.Suppliers.H ost	orders	JavaScript функції для обробки та відображення аналітичних даних щодо замовлень у вигляді графіків.
Система аналітики онлайн замовлень	Vlogo.Suppliers.H ost	products	JavaScript функції для обробки та відображення аналітичних даних щодо продуктів у вигляді графіків.
Система аналітики онлайн замовлень	Vlogo.Suppliers.H ost	ratings	JavaScript функції для обробки та відображення аналітичних даних щодо рейтингу у вигляді графіків.
Система аналітики онлайн замовлень	Vlogo.Suppliers.H ost	revenue	JavaScript функції для обробки та відображення аналітичних даних щодо доходів у вигляді графіків.

Продовження таблиці 1.1

Рейтингова система	Vlogo.Ratings.Contracts	Rating	Клас, що представляє собою контракт рейтингу.
Рейтингова система	Vlogo.Ratings.Application	ServiceCollectionExtensions	Статичний клас, що дозволяє зареєструвати всі залежності модуля рейтингу в системі інжектування залежностей.
Рейтингова система	Vlogo.Ratings.Application	RatingSettings	Клас, потрібний для конфігурування модуля рейтингу. За допомогою нього можна налаштувати базу даних для роботи рейтингової підсистеми.

Продовження таблиці 1.1

Рейтингова система	Vlogo.Ratings.Application	IRatingService	Абстракція у вигляді інтерфейсу для роботи з рейтингом. Дозволяє створювати, видаляти рейтинг, змінювати.
Рейтингова система	Vlogo.Ratings.Application	RatingService	Клас, що реалізовує логіку роботи з рейтингом. Він використовує репозиторій для роботи із сховищем даних.
Рейтингова система	Vlogo.Ratings.Application	IRatingRepository	Абстракція у вигляді інтерфейсу для роботи із сховищем даних рейтингу.

Продовження таблиці 1.1

Рейтингова система	Vlogo.Ratings.Application	MongoRatingRepository	Реалізація шаблону проектування репозиторій, що взаємодіє із нереляційним сховищем даних – MongoDB. Дозволяє отримати, додати, змінити та видалити рейтингові дані.
Рейтингова система	Vlogo.Suppliers.Host	RatingsController	Контролер для роботи із рейтинговими даними. Дозволяє створювати, змінювати та видаляти дані про рейтинг.

Продовження таблиці 1.1

Рейтингова система	Vlogo.Suppliers.Host	Rate	Клас, що представляє собою форму, що надається клієнту для проставлення рейтингу після успішного виконання замовлення постачальником.
Рейтингова система	Vlogo.Suppliers.Host	All	Клас, що потрібен для зображення списку рейтингу.
Система керування замовленнями	Vlogo.Orders.Contracts	OrderStatus	Сутність для відображення статусу замовлення.
Система керування замовленнями	Vlogo.Orders.Contracts	Order	Клас, що відображає замовлення в системі.

Продовження таблиці 1.1

Система керування замовленнями	Vlogo.Orders.Application	ServiceCollectionExtensions	Статичний клас для спрощення реєстрації залежностей, необхідних для роботи модуля системи керування замовленнями.
Система керування замовленнями	Vlogo.Orders.Application	OrdersSettings	Клас, що дозволяє налаштувати підсистему керування замовленнями. Дає можливість наконфігурувати роботу із сховищем даних.
Система керування замовленнями	Vlogo.Orders.Application	IOOrdersService	Абстракція для роботи із замовленнями.

Продовження таблиці 1.1

Система керування замовленнями	Vlogo.Orders.Application	OrdersService	Клас, що є реалізацією абстракції для роботи із замовленнями. Він дозволяє додати, видалити та змінити статус замовлення.
Система керування замовленнями	Vlogo.Orders.Application	IOOrdersRepository	Інтерфейс для роботи із сховищем даних замовлень.
Система керування замовленнями	Vlogo.Orders.Application	MongoOrdersRepository	Реалізація інтерфейсу для роботи із сховищем даних замовлень, що використовує MongoDB.

Продовження таблиці 1.1

Система керування замовленнями	Vlogo.Suppliers.Host	OrdersController	Контролер для роботи з замовленнями. Дозволяє клієнту створювати нові, а постачальнику змінювати стан існуючих замовлень.
Система керування замовленнями	Vlogo.Suppliers.Host	OrderModel	Модель, що використовується для відображення інформації щодо замовлення.
Система керування замовленнями	Vlogo.Suppliers.Host	All	Клас, що необхідний для відображення списку всіх замовлень на веб-сайті.
Система керування замовленнями	Vlogo.Suppliers.Host	Pending	Клас, що необхідний для відображення списку незавершених замовлень.

1.2 Архітектура БД

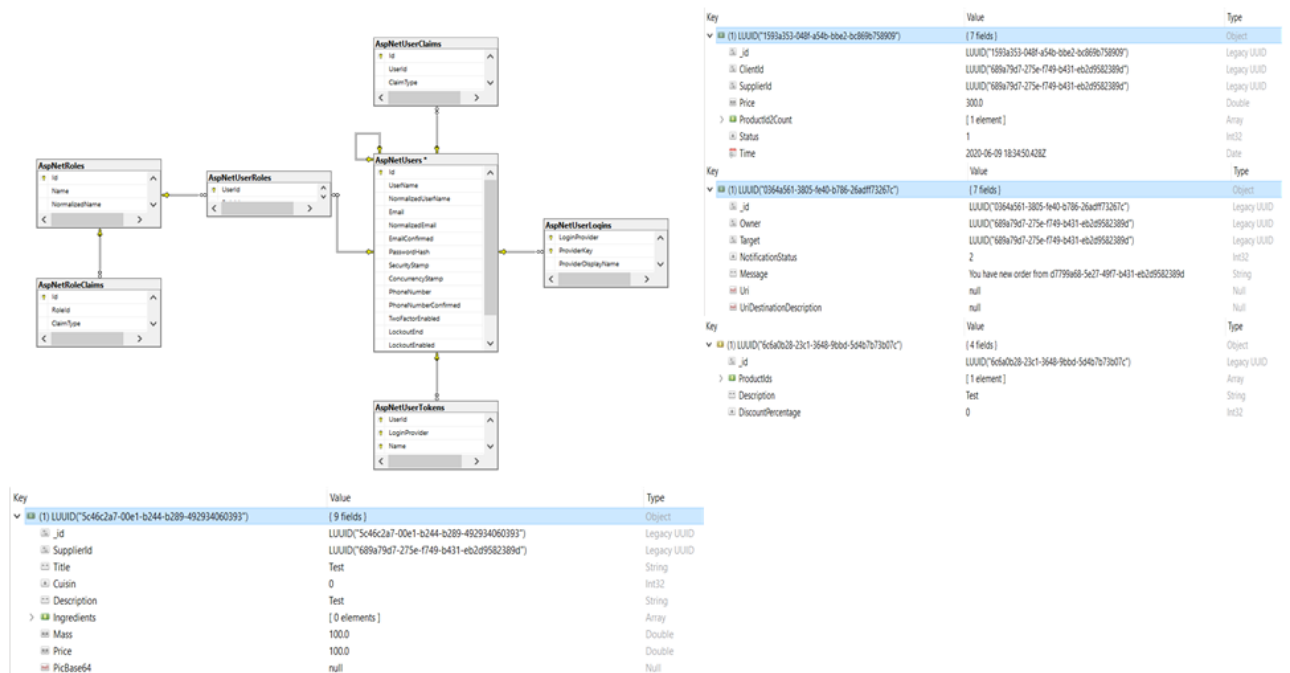


Рисунок 1.11 – Схема бази даних

1.3 Аналіз безпеки даних

Дана система працює з величезною кількістю персональних даних клієнтів, тому питання безпеки виходить на перший план.

Для досягнення максимальної безпеки інформації, було прийнято рішення про використання HTTPS протоколу взаємодії. Також була необхідність для генерації сертифікатів, для цього було використано утиліту CertBot, що дозволило згенерувати SSL сертифікат.

Дані про стан клієнта та сесії зберігаються за допомогою Cookie та з використанням AntiforgeryToken, для забезпечення максимальної безпеки та захисту від XSS атак.

Для збереження даних користувачів у сховищі даних, було використано останню версію бібліотеки Identity. Це абстрагує нас від прямої роботи по шифруванню та безпеці даних. Завдяки використанню нових та надійних

підходів даних, дана бібліотека гарантує безпеку інформації, навіть за умови викрадення бази даних.

1.4 Висновки до розділу

В даному розділі були описані 5 компонентів системи, технології, які були використані та причини їх використання. Також було продемонстровано схему роботи кожного модуля та в деталях, описані можливості та подальші можливі зміни системи.

					КПІ.ІП-6329.045440.09.81	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

2 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Опис процесів тестування

Тестування – це є один із найважливіших етапів у циклі розробки програмного забезпечення, завдяки йому вдається виявити неточності, суперечність із функціональними або нефункціональними вимогами. Існує багато різних видів тестів, кожен з яких призначений для виявлення різного роду проблем.

Для тестування даного комплексу програм, були використані різні види тестів, кожен з яких було уніфіковано для створення базового шаблону тестування, що можливо використовувати повторно під час тестування різних компонентів та їх поєднань.

2.1.1 Юніт тестування

Цей вид тестування є одним із найважливіших, оскільки дозволяє створити максимально прості та ізольовані тести, що самі по собі є легкими в написанні, запуску та в деяких випадках можуть замінити документацію.

Для юніт тестування було використано базовий набір бібліотек та підходів. Як фреймворк був використаний .Net Core та бібліотека Xunit, що дозволяє в простій та зручній формі створити тести. Бонусом використання даних технологій буде можливість запуску тестів паралельно, тобто в один і той самий момент часу, буде запущено декілька різних юніт тестів, що дозволить значно пришвидшити процес проведення юніт тестування. Для створення об'єктів заглушок, що дозволяють створювати більш гранулярні та незалежні тести, була використана бібліотека NSubstitute.

Завдяки використанню платформи .Net Core, процес створення, пошуку помилок та запуску юніт тестів став можливим не тільки спеціальною командою,

					КПІ.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

а і з використанням інтегрованого середовища розробки, який використовується під час написання компонентів.

2.1.2 Функціональне тестування

Всі функціональні вимоги були описані та створені тест-кейси для проведення функціонального тестування. Для проведення тестування, розгортається вся система. Після розгортання, достатньо виконати список тест-кейсів, та перевірити валідність поведінки системи.

2.2 Опис контрольного прикладу

Перед проведенням тестування, система була розгорнута з нуля. Для цього був використаний спеціальний скрипт, що піднімає кожен компонент та системи керування базами даних. Після розгортання, потрібно перевірити відсутність будь-яких даних в системі, та видалити їх за наявності.

Тестування проводилося на комп'ютері з операційною системою Windows 10, 16 Гб оперативної пам'яті та жорстким диском на 500 Гб. Система була розгорнута з використанням утиліти docker-compose, що використовувала 3 віртуальних ядра із 4 фізичних та 4 Гб оперативної пам'яті.

Таблиця 2.1 – Опис тест-кейсів

Ідентифікатор	Компонент	Опис
П-1	Постачальники	Перевірити можливість реєстрації постачальника з правильними вхідними даними

Продовження таблиці 2.1

П-2	Постачальники	Перевірити неможливість реєстрації постачальника з неправильними вхідними даними. Повинна висвітитися помилка з описом не валідних вхідних даних
П-3	Постачальники	Перевірити можливість авторизації постачальника з правильними вхідними даними
П-4	Постачальники	Перевірити неможливість авторизації постачальника з неправильними вхідними даними. Повинна висвітитися помилка з описом не валідних вхідних даних
П-5	Постачальники	Можливість перегляду списку всіх своїх товарів
П-6	Постачальники	Можливість редагувати товари з використанням правильних вхідних даних

Продовження таблиці 2.1

П-7	Постачальники	Неможливість редагувати товари з використанням не валідних вхідних даних
П-8	Постачальники	Редагований продукт повинен з'явитися у списку продуктів із внесеними змінами
П-9	Постачальники	Можливість видалити товари
П-10	Постачальники	Видалений товар повинен зникнути зі списку товарів
П-11	Постачальники	Можливість перегляду історії замовлень
П-12	Постачальники	Можливість перегляду активних замовлень
П-13	Постачальники	Можливість зміни статусу замовлення
П-14	Постачальники	Можливість перегляду аналітики
П-15	Постачальники	Можливість перегляду рейтингу у вигляді списку

Продовження таблиці 2.1

П-16	Постачальники	Перевірити неможливість створення продукту постачальника з неправильними вхідними даними
П-17	Постачальники	Перевірити можливість створення продукту постачальника із правильними вхідними даними
А-1	Аналітика	Можливість фільтрації аналітичних даних по часу
А-2	Аналітика	Відображення валідних аналітичних даних у вигляді графіків за кількістю активних замовлень
А-3	Аналітика	Відображення валідних аналітичних даних у вигляді графіків за кількістю замовлень у різних станах
А-4	Аналітика	Відображення валідних аналітичних даних у вигляді графіків по рейтингу постачальника

Продовження таблиці 2.1

А-5	Аналітика	Відображення валідних аналітичних даних у вигляді графіків за кількістю продуктів
А-6	Аналітика	Відображення валідних аналітичних даних у вигляді графіків за кількістю замовлень певного продукту
А-7	Аналітика	Відображення валідних аналітичних даних у вигляді графіків кореляції ціни та кількості замовлень продуктів
РТ-1	Рейтингова система	Можливість для постачальників переглядати персональний рейтинг
РТ-2	Рейтингова система	Можливість клієнтів залишати відгуки та ставити рейтинг після виконання замовлення
РТ-3	Рейтингова система	Використання системи повідомлень для відправлення форми відгуку про виконане замовлення

Продовження таблиці 2.1

Б-1	Бонусна система	Можливість постачальників створювати бонуси з правильними вхідними даними
Б-2	Бонусна система	Неможливість постачальників створювати бонуси з неправильними вхідними даними
Б-3	Бонусна система	Можливість постачальників редагувати бонуси з правильними вхідними даними
Б-4	Бонусна система	Неможливість постачальників створювати редагувати з неправильними вхідними даними
Б-5	Бонусна система	Можливість видалення бонусів
Б-6	Бонусна система	Можливість клієнтів використовувати бонуси
З-1	Замовлення	Можливість постачальників переглядати замовлення

Продовження таблиці 2.1

3-2	Замовлення	Можливість клієнтів створення замовлень
3-3	Замовлення	Можливість постачальників змінювати статус замовлення

2.3 Висновки до розділу

Під час виконання роботи, чимало часу було приділено проектуванню, створенню та проведенню різного виду тестування. Це позитивно вплинуло на процес створення, еволюції та швидкість розробки програмного забезпечення, оскільки гарантувало працездатність та валідність поведінки програми, навіть під час правок та змінах у ключових компонентах системи.

Завдяки юніт наявності низки юніт тестів, розробка та еволюція кожного окремого компоненту була максимально спрощена, також значно зросла швидкість внесення змін. Але наскільки б багато не було створено юніт тестів, вони не давали гарантії збереження функціонально правильного стану системи та її валідної поведінки. Для цього спеціально були створені тест кейси, що описували валідний функціонал системи та пройшовши які, можна було гарантувати правильність або неправильність поведінки системи.

Таблиця 2.2 – Звіт тестування

Ідентифікатор	Очікуваний результат	Фактичний результат	Статус тесту
---------------	----------------------	---------------------	--------------

Продовження таблиці 2.2

П-1	Постачальник має можливість зареєструватись з правильними вхідними даними	Постачальник має можливість зареєструватись з правильними вхідними даними	Пройдено
П-2	Постачальник не має можливості зареєструватися з неправильними вхідними даними	Постачальник не має можливості зареєструватися з неправильними вхідними даними	Пройдено
П-3	Постачальник має можливість авторизуватися з правильними вхідними даними	Постачальник має можливість авторизуватися з правильними вхідними даними	Пройдено
П-4	Постачальник не має можливість авторизуватися з неправильними вхідними даними	Постачальник не має можливість авторизуватися з неправильними вхідними даними	Пройдено

Продовження таблиці 2.2

П-5	Постачальник має можливість переглянути список товарів	Постачальник має можливість переглянути список товарів	Пройдено
П-6	Постачальник має можливість редагувати продукти із свого списку при введені коректних даних	Постачальник має можливість редагувати продукти із свого списку при введені коректних даних	Пройдено
П-7	Постачальник не має можливості редагувати продукти із свого списку при введені некоректних даних	Постачальник не має можливості редагувати продукти із свого списку при введені некоректних даних	Пройдено
П-8	Редагований продукт повинен з'явитися у списку продуктів із внесеними змінами	Редагований продукт повинен з'явитися у списку продуктів із внесеними змінами	Пройдено
П-9	Постачальник має можливість видаляти товари	Постачальник має можливість видаляти товари	Пройдено
П-10	Видалений товар повинен зникнути зі списку товарів	Видалений товар повинен зникнути зі списку товарів	Пройдено

Продовження таблиці 2.2

П-11	Постачальник має можливість перегляду історії замовлень	Постачальник має можливість перегляду історії замовлень	Пройдено
П-12	Постачальник має можливість перегляду активних замовлень	Постачальник має можливість перегляду активних замовлень	Пройдено
П-13	Постачальник має можливість зміни статусу замовлення	Постачальник має можливість зміни статусу замовлення	Пройдено
П-14	Постачальник має можливість перегляду аналітики	Постачальник має можливість перегляду аналітики	Пройдено
П-15	Постачальник має можливість перегляду рейтингу у вигляді списку	Постачальник має можливість перегляду рейтингу у вигляді списку	Пройдено
П-16	Постачальник не має можливості створення продукту із неправильними вхідними даними	Постачальник не має можливості створення продукту постачальника з неправильними вхідними даними	Пройдено

Продовження таблиці 2.2

П-17	Постачальник має можливість створення продукту з правильними вхідними даними	Постачальник має можливість створення продукту з правильними вхідними даними	Пройдено
А-1	Постачальник має можливість фільтрації аналітичних даних по часу	Постачальник має можливість фільтрації аналітичних даних по часу	Пройдено
А-2	Аналітика містить дані за кількістю активних замовлень	Аналітика містить дані за кількістю активних замовлень	Пройдено
А-3	Аналітика містить дані за кількістю замовлень у різних станах	Аналітика містить дані за кількістю замовлень у різних станах	Пройдено
А-4	Аналітика містить дані по рейтингу постачальника	Аналітика містить дані по рейтингу постачальника	Пройдено
А-5	Аналітика містить дані за кількістю продуктів	Аналітика містить дані за кількістю продуктів	Пройдено

Продовження таблиці 2.2

А-6	Аналітика містить дані за кількістю замовлень певного продукту	Аналітика містить дані за кількістю замовлень певного продукту	Пройдено
А-7	Аналітика містить дані по кореляції ціни та кількості замовлень продуктів	Аналітика містить дані по кореляції ціни та кількості замовлень продуктів	Пройдено
РТ-1	Постачальник має можливість переглядати персональний рейтинг	Постачальник має можливість переглядати персональний рейтинг	Пройдено
РТ-2	Клієнт має можливість залишати відгуки та ставити рейтинг після виконання замовлення	Клієнт має можливість залишати відгуки та ставити рейтинг після виконання замовлення	Пройдено
РТ-3	Клієнт має отримати повідомлення з проханням проставити рейтинг після виконання замовлення	Клієнт має отримати повідомлення з проханням проставити рейтинг після виконання замовлення	Пройдено

Продовження таблиці 2.2

Б-1	Постачальник має можливість створювати бонуси з правильними вхідними даними	Постачальник має можливість створювати бонуси з правильними вхідними даними	Пройдено
Б-2	Постачальник не має можливість створювати бонуси з неправильними вхідними даними	Постачальник не має можливість створювати бонуси з неправильними вхідними даними	Пройдено
Б-3	Постачальник має можливість редагувати бонуси з правильними вхідними даними	Постачальник має можливість редагувати бонуси з правильними вхідними даними	Пройдено
Б-4	Постачальник не має можливість редагувати бонуси з неправильними вхідними даними	Постачальник не має можливість редагувати бонуси з неправильними вхідними даними	Пройдено
Б-5	Постачальник має можливість видалення бонусів	Постачальник має можливість видалення бонусів	Пройдено
Б-6	Клієнт має можливість автоматично використовувати бонуси	Клієнт має можливість автоматично використовувати бонуси	Пройдено

Продовження таблиці 2.2

3-1	Постачальник має можливість переглядати замовлення	Постачальник має можливість переглядати замовлення	Пройдено
3-2	Клієнт має можливість створювати замовлення	Клієнт має можливість створювати замовлення	Пройдено
3-3	Постачальник має можливість змінювати статус замовлення	Постачальник має можливість змінювати статус замовлення	Пройдено

3 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розгортання програмного забезпечення

Під час створення прототипу програмного комплексу, постало питання щодо розгортання кожного компонента окремо та всієї системи в цілому. З одного боку компоненти програмного комплексу є ізольованими один від одного, що робить систему гарно масштабуємою та спрощує процес еволюції, але водночас постає питання інсталювання, оскільки система поділена на багато підсистем, то процес встановлення програмного комплексу ускладнюється. Постає проблема атомарності та консистентності розгортання системи. Для розв'язання цих проблем, були вибрані технології контейнеризації, а саме docker, docker-compose. Використання утиліти docker дозволило декларативним шляхом описати процес розгортання кожного сервісу зі всіма необхідними залежностями для коректної роботи системи. Результатом став файл(docker image) написаний в декларативному стилі, що завдяки утиліті docker можна розповсюджувати через мережу Інтернет та запускати на будь-якому комп'ютері зі встановленою утилітою docker. Також, використання dockery дозволило скористатися утилітою docker-compose, що в свою чергу дуже підходить для розгортання розподілених систем. Для цього було створено спеціальний файл-маніфест, що описує кожен із сервісів розподіленої системи та його налаштування. Для опису сервісів необхідно мати образи кожного сервісу, що беруть участь в розподіленій системі.

Так, як деякі компоненти програмного комплексу є персистентними, то для їх правильного функціонування, необхідно завчасно розгорнути системи керування базами даних, такі як: MongoDB, MS SQL Server. Це можна зробити як на комп'ютері, де буде запущена сама система, чи виділити окремий сервер для цих цілей.

Для полегшення процесу інсталювання системи для її подальшого автоматичного та ручного тестування, був розроблений окремий підхід, що

					КПІ.ІП-6329.045440.09.81	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

використовує утиліту docker-compose для підняття систем керування базами даних разом із всіма компонентами комплексу програм. В такому випадку, для підняття системи не потрібно окремо розгортати системи керування базами даних.

Для розгортання рейтингової системи, системи бонусів, системи керування постачальниками, системи керування замовленнями, аналітичної системи, був створений окремий dockerfile, що містить конфігураційні параметри(їх можна змінювати під час запуску образу). Сам dockerfile, що описує процес побудови та запуску образу складається із етапу компілювання кодової бази та етапу встановлення точки входу в образ.

Для розгортання всієї системи в цілому, було створено docker-compose файл, що описує та конфігурує сервіси управління постачальниками, управління замовленнями, аналітики, рейтингу та бонусів. Для цього він використовує раніше створені dockerfileи, що по своїй суті і виступають у вигляді сервісів та компонентів розподіленої системи. Також було створено окремий docker-compose файл для тестування системи, що окрім вищеперерахованих сервісів містить системи керування базами даних: Mongo та Microsoft SQL Server.

Перед інсталюванням програмного комплексу на сервер, необхідно завчасно встановити 2 утиліти: docker та docker-compose. Також, потрібно розгорнути системи керування базами даних за умови розгортання не в тестовому режимі.

3.2 Робота з програмним забезпеченням

Керівництво користувача описане у «Керівництво користувача».

3.3 Висновки до розділу

У даному розділі були детально описані підходи до розгортання, розглянуті проблеми, що зустрілися під час проведення роботи, а також описані

					КПІ.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		59

вимоги для сервера, що буде використовуватися для інсталювання програмного комплексу.

					КПІ.ІП-6329.045440.09.81	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У даній роботі був розроблений комплекс програмного забезпечення, що є частиною системи для керування замовленнями. Для реалізації задачі була використана низка архітектурних підходів та рішень, що застосовуються в розподілених системах. Був використаний підхід мікросервісів, для відокремлення кожного компоненту та підсистеми як окремого сервісу, що виконує вузькоспеціалізовані задачі та може використовуватися як клієнтської частини напряму, так і іншими мікросервісами. Також, для проєктування певних компонентів був використаний підхід предметно-орієнтованого дизайну, що дозволило максимально точно відобразити сутність та сценарії використання даного сервісу.

Важливим було питання вибору платформи для розробки. Для створення комплексу програм, була вибрана платформа .Net Core, що дозволило досить просто побудувати описану архітектуру та налагодити систему. Кожна підсистема комплексу програм, була розроблена та протестована на платформі .Net.

Результатом роботи став комплекс програм, що являє собою розподілену систему для створення та керування онлайн замовленнями, аналітиці, інтеграції бонусів та рейтингової системи.

Був проведений детальний пошук та аналіз аналогів, що присутні на ринку та мають подібний функціонал. Серед усіх конкурентів, слід виділити основного – це система Glovo. Були розглянуті її позитивні сторони та виявлені певні недоліки. На базі аналізу був розроблений план та функціонал, що повинен повторювати позитивні моменти конкурентів та виправляти знайдені недоліки.

Як сховище даних, була вибрана нереляційна база даних – MongoDB. На вибір вплинуло декілька факторів, таких як: простота роботи, легка можливість горизонтального масштабування та реплікації сховища даних.

Основна задача при реалізації компонентів системи було створити комплекс програм, що дозволить організувати роботу постачальників,

					КПІ.ІП-6329.045440.09.81	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

починаючи із реєстрації та створення товарів та закінчуючи аналітикою бізнесу. Робота з постачальниками є однією із основних в рамках даної роботи, оскільки вони складають немалу частину цільової аудиторії розробленої системи. Також, під час створення системи, наголос був зроблений на зручний спосіб аналітики бізнесу постачальників, що дозволить корегувати його для збільшення прибутку.

У зв'язку з високою складністю та розподіленим характером системи, була розроблена низка різного виду тестів, для зіставлення готового продукту із висунутими до нього вимогами. В результаті тестування було знайдено чимало розбіжностей, але усі вони були успішно виправлені та заново протестовані.

					КПІ.ІП-6329.045440.09.81	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК ПОСИЛАНЬ

- 1) Software development in Ukraine: 2019-2020 IT market report [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://www.n-ix.com/software-development-in-ukraine-2019-2020-market-report/>.
- 2) Workplace Automation is Everywhere [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://www.businessnewsdaily.com/9835-automation-tech-workforce.html>.
- 3) Исследование украинского Telegram [Електронний ресурс]. – 2019 – Режим доступу до ресурсу: <https://vctr.media/ukrainskiy-telegram-5000-15716/>.
- 4) China, WeChat, and the Origins of Chatbots [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: <https://chatbotsmagazine.com/china-wechat-and-the-origins-of-chatbots-89c481f15a44>.
- 5) Slack Statistics [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://expandedramblings.com/index.php/slack-statistics/>.
- 6) Microsoft Teams now has 75 million daily active users [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://www.businessinsider.com/microsoft-teams-hits-75-million-daily-active-users-2020-4>.
- 7) Project Management Digital Assistant [Електронний ресурс] – Режим доступу до ресурсу: <https://standuply.com/>.
- 8) Task management, Project Management and Calendar for Slack | Kyber [Електронний ресурс] – Режим доступу до ресурсу: <https://kyber.me/>.
- 9) Vacation Tracker | Slack App Directory [Електронний ресурс] – Режим доступу до ресурсу: <https://slack.com/apps/A1L3ZBM7G-vacation-tracker>.
- 10) Hubot is your friendly robot sidekick. [Електронний ресурс] – Режим доступу до ресурсу: <https://hubot.github.com/>.
- 11) Heartbeat (computing) [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Heartbeat_\(computing\)](https://en.wikipedia.org/wiki/Heartbeat_(computing)).

12) Amazon DynamoDB vs. Memcached vs. Redis [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://db-engines.com/en/system/Amazon+DynamoDB%3BMemcached%3BRedis>.

13) Amazon DynamoDB vs. CouchDB vs. MongoDB [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://db-engines.com/en/system/Amazon+DynamoDB%3BCouchDB%3BMongoDB>.

14) Introduction to ASGI: Emergence of an Async Python Web Ecosystem [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://florimond.dev/blog/articles/2019/08/introduction-to-asgi-async-python-web/>.

15) HMAC. // Practical UNIX and Internet Security, 3rd Edition / , 2011. – С. 187–194.

16) Verifying requests from Slack [Електронний ресурс] – Режим доступу до ресурсу: <https://api.slack.com/authentication/verifying-requests-from-slack>.

17) How to Write a Test Plan with the IEEE 829 Standard [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://reqtest.com/testing-blog/how-to-write-a-test-plan-2/>.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“ ____ ” _____ 2020 р.

КОМПЛЕКС ПРОГРАМ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ
ЗАМОВЛЕНЬ ТОВАРІВ ТА ПОСЛУГ (КОМПЛЕКСНА ТЕМА)

Опис програми

КПШ.ІІІ-6329.045440.11.13

“ПОГОДЖЕНО”

Керівник проєкту:

М.С. Храмченко

Нормоконтроль:

К.І. Ліщук

Виконавець:

Д.М.Шелудько

Київ - 2020 року

1 КОНТРОЛЕРИ

```

AnalyticsController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Vlogo.Analytics.Contracts;
using Vlogo.Orders.Application.Services;
using Vlogo.Orders.Contracts;
using Vlogo.Ratings.Application.Services;
using Vlogo.Suppliers.Application.Services;

namespace Vlogo.Suppliers.Host.Controllers
{
    [Route("analytics")]
    public class AnalyticsController : Controller
    {
        private readonly IOrdersService _orders;
        private readonly IProductService _products;
        private readonly IRatingService _rating;

        public AnalyticsController(IOrdersService orders, IProductService products, IRatingService
rating)
        {
            _orders = orders;
            _products = products;
            _rating = rating;
        }

        [HttpGet("get")]
        public IActionResult Get()
        {
            return View();
        }

        [HttpGet("revenue")]
        public async Task<JsonResult> GetRevenue(DateTime? start, DateTime? end)
        {
            var all = await _orders.GetAll();
            var revenue2Date = all
                .GroupBy(order => order.Time.Date)
                .Select(g => new KeyValuePair<DateTime, double>(g.Key, g.Sum(order =>
order.Price)))
                .ToList();

            var analytics = new RevenueAnalytics
            {

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

```

        Description = "Revenue per day",
        Date2Value = revenue2Date,
        Title = "Revenue"
    };

    return Json(analytics);
}

[HttpGet("orders")]
public async Task<JsonResult> GetOrders(DateTime? start, DateTime? end)
{
    var all = await _orders.GetAll();

    var anal = new OrderAnalytics
    {
        PendingOrders = all.Count(order => order.Status != OrderStatus.Done),
        TotalOrders = all.Count,
        Status2Count = new List<KeyValuePair<string, int>>
        {
            new KeyValuePair<string, int>(OrderStatus.Cooking.ToString(),
                all.Count(order => order.Status == OrderStatus.Cooking)),
            new KeyValuePair<string, int>(OrderStatus.Created.ToString(),
                all.Count(order => order.Status == OrderStatus.Created)),
            new KeyValuePair<string, int>(OrderStatus.Delivering.ToString(),
                all.Count(order => order.Status == OrderStatus.Delivering)),
            new KeyValuePair<string, int>(OrderStatus.Done.ToString(),
                all.Count(order => order.Status == OrderStatus.Done))
        }
    };

    return Json(anal);
}

[HttpGet("products")]
public async Task<JsonResult> GetProducts(DateTime? start, DateTime? end)
{
    var all = await _products.GetAll();

    var orders = await _orders.GetAll();

    var descr = all.Select(product => new KeyValuePair<string,
    ProductDescription>(product.Title,
        new ProductDescription
        {
            Mass = product.Mass,
            PricePerOne = product.Price,
            Ordered = orders.Where(order =>
order.ProductId2Count.Contains(product.Id)).Count(),
            TotalRevenue = orders
                .Where(order => order.ProductId2Count.Contains(product.Id))

```

					КП.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

```
.Sum(order => order.Price)
    )))
.ToList();

var anal = new ProductAnalytics
{
    TotalProductsCount = all.Count,
    ProductsDescription = descr
};

return Json(anal);
}

[HttpGet("rating")]
public async Task<JsonResult> GetRating(DateTime? start, DateTime? end)
{
    var all = await _rating.GetAll();

    var points2Count = all
        .GroupBy(rating => rating.Points)
        .Select(g => new KeyValuePair<int, int>(g.Key, g.Sum(r => r.Points)))
        .ToList();

    var anal = new RatingAnalytics
    {
        Points2Count = points2Count
    };

    return Json(anal);
}
}

BonusesController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Vlogo.Bonuses.Application.Services;
using Vlogo.Bonuses.Contracts;
using Vlogo.Suppliers.Application.Services;
using Vlogo.Suppliers.Host.Models.Bonuses;

namespace Vlogo.Suppliers.Host.Controllers
{
    [Route("[controller]")]
    public class BonusesController : Controller
    {
        private readonly IProductService _productService;
        private readonly IBonusService _service;
```

```

public BonusesController(IBonusService service, IProductService productService)
{
    _service = service;
    _productService = productService;
}

[HttpGet]
public async Task<IActionResult> All()
{
    var all = await _service.GetAll();
    var productIds = await _productService.Get(all.SelectMany(b => b.ProductIds));
    var models = all.Select(bonus => new BonusModel
    {
        Id = bonus.Id,
        Products = productIds?.Select(p => new ShortProductModel
        {
            Id = p.Id,
            Name = p.Title
        }).ToList(),
        DiscountPercentage = bonus.DiscountPercentage,
        Description = bonus.Description
    }).ToList();
    return View(models);
}

[HttpGet("create")]
public ActionResult Create()
{
    return View(new CreateBonusModel
    {
        Products = new List<SelectProductModel>()
    });
}

[HttpPost("create")]
public async Task<IActionResult> Create(CreateBonusModel model)
{
    var bonus = new Bonus
    {
        Id = model.Id == Guid.Empty ? Guid.NewGuid() : model.Id,
        DiscountPercentage = model.DiscountPercentage,
        ProductIds = model.Products.Select(p => p.Id).ToList(),
        Description = model.Description
    };

    await _service.CreateUpdate(bonus);

    return RedirectToAction(nameof(All));
}

[HttpGet("edit/{id:guid}")]

```

```

public async Task<IActionResult> Edit(Guid id)
{
    var bonus = await _service.Get(id);
    var products = await _productService.GetAll();
    var model = new CreateBonusModel
    {
        Id = bonus.Id,
        Products = products.Select(p => new SelectProductModel
        {
            Id = p.Id,
            Name = p.Title,
            IsSelected = bonus.ProductIds.Contains(p.Id)
        }).ToList(),
        DiscountPercentage = bonus.DiscountPercentage,
        Description = bonus.Description
    };
    return View(model);
}

[HttpPost("edit/{id:guid}")]
public async Task<IActionResult> Edit(Guid id, CreateBonusModel model)
{
    var bonus = new Bonus
    {
        Id = model.Id == Guid.Empty ? Guid.NewGuid() : model.Id,
        DiscountPercentage = model.DiscountPercentage,
        ProductIds = model.Products.Select(p => p.Id).ToList(),
        Description = model.Description
    };

    await _service.CreateUpdate(bonus);

    return RedirectToAction(nameof(All));
}

[HttpPost("delete/{id:guid}")]
public async Task<IActionResult> Delete(Guid id)
{
    await _service.Delete(id);

    return RedirectToAction(nameof(All));
}

[HttpPost("shortProducts/all")]
public async Task<IActionResult> ShortProductDescription(CreateBonusModel model)
{
    var allProducts = await _productService.GetAll();

    var all = allProducts.Select(p => new SelectProductModel
    {
        Id = p.Id,

```

```

        Name = p.Title,
        IsSelected = false
    }).ToList();

    model.Products = all;
    return PartialView("_SelectProducts", model);
}
}
}
HomeController.cs
using Microsoft.AspNetCore.Mvc;

namespace Vlogo.Suppliers.Host.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public IActionResult Index()
        {
            return View();
        }
    }
}
OrdersController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Vlogo.Auth.Model.Extensions;
using Vlogo.Notifications.Application.Services;
using Vlogo.Notifications.Contracts;
using Vlogo.Orders.Application.Services;
using Vlogo.Orders.Contracts;
using Vlogo.Suppliers.Application.Services;
using Vlogo.Suppliers.Host.Models.Orders;

namespace Vlogo.Suppliers.Host.Controllers
{
    public class OrdersController : Controller
    {
        private readonly INotificationsService _notificationService;
        private readonly IProductService _productService;
        private readonly IOOrdersService _service;

        public OrdersController(IOOrdersService service, IProductService productService,
            INotificationsService notificationsService)
        {
            _service = service;
            _productService = productService;
            _notificationService = notificationsService;

```

```

    }

    public async Task<IActionResult> All()
    {
        var all = await GetAll();
        return View(all);
    }

    public async Task<IActionResult> Pending()
    {
        var all = await GetAll();
        var pending = all.Where(o => o.Status != OrderStatus.Done).ToList();
        return View(pending);
    }

    [HttpPost("moveNext/{id:guid}")]
    public async Task<IActionResult> MoveNext(Guid id)
    {
        var userId = HttpContext.User.GetGuid();
        var order = await _service.Get(id);
        var previousStatus = order.Status;

        order.Status = order.Status switch
        {
            OrderStatus.Created => OrderStatus.Cooking,
            OrderStatus.Cooking => OrderStatus.Delivering,
            OrderStatus.Delivering => OrderStatus.Done,
            _ => order.Status
        };

        await _service.CreateUpdate(order);
        var userNotification = new Notification
        {
            Message =
                $"Your order's [{order.Id}] status has been changed from [{previousStatus}] to [{order.Status}] SId [{userId}]",
            NotificationStatus = NotificationStatus.New,
            Owner = userId,
            Target = order.ClientId
        };
        await _notificationService.CreateUpdate(userNotification);

        var products = await _productService.Get(order.ProductId2Count.Keys);
        var product2Count = products
            .Select(p => new { Product = p, Count = order.ProductId2Count[p.Id] })
            .ToDictionary(arg => arg.Product, arg => arg.Count);
        var model = new OrderModel
        {
            Id = order.Id,
            Status = order.Status,
            ClientId = order.ClientId,

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

```

        Price = order.Price,
        Product2Count = product2Count
    };

    return PartialView("Partial/_Order", model);
}

private async Task<IList<OrderModel>> GetAll()
{
    var all = await _service.GetAll();
    var productIds = all.SelectMany(o => o.ProductId2Count.Keys);
    var products = await _productService.Get(productIds.ToList());
    var model = all.Select(o =>
    {
        var product2Count = products
            .Where(p => o.ProductId2Count.ContainsKey(p.Id))
            .Select(p => new { Product = p, Count = o.ProductId2Count[p.Id] })
            .ToDictionary(arg => arg.Product, arg => arg.Count);

        return new OrderModel
        {
            Id = o.Id,
            Status = o.Status,
            Price = o.Price,
            ClientId = o.ClientId,
            Product2Count = product2Count
        };
    });

    return model.ToList();
}
}
}

ProductsController.cs
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Vlogo.Auth.Model.Extensions;
using Vlogo.Suppliers.Application.Services;
using Vlogo.Suppliers.Contracts.Products;

namespace Vlogo.Suppliers.Host.Controllers
{
    [Route("[controller]")]
    public class ProductsController : Controller
    {
        private readonly IWebHostEnvironment _env;

```

					КП.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

```

private readonly IProductService _services;

public ProductsController(IProductService services, IWebHostEnvironment env)
{
    _services = services;
    _env = env;
}

[HttpGet]
public async Task<IActionResult> All()
{
    var all = await _services.GetAll();

    return View(all);
}

[HttpGet("create")]
public ActionResult Create()
{
    return View(new Product
    {
        Ingredients = new List<string>()
    });
}

[HttpPost("create")]
public async Task<IActionResult> Create(Product product, IFormFile pic)
{
    if (pic != null)
        using (var ms = new MemoryStream())
        {
            pic.CopyTo(ms);
            var fileBytes = ms.ToArray();
            product.PicBase64 = Convert.ToBase64String(fileBytes);
        }

    product.SupplierId = HttpContext.User.GetGuid();
    await _services.CreateUpdate(product);

    return RedirectToAction(nameof(All));
}

[HttpGet("edit/{id:guid}")]
public async Task<ActionResult> Edit(Guid id)
{
    var product = await _services.Get(id);

    return View(product);
}

[HttpPost("addIngredient")]

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

```

public IActionResult AddIngredient(Product product)
{
    product.Ingredients.Add(string.Empty);
    return PartialView("_Ingredient", product);
}

[HttpPost("removeIngredient/{index:int}")]
public IActionResult RemoveIngredient(int index, Product product)
{
    product.Ingredients.RemoveAt(index);
    return PartialView("_Ingredient", product);
}

[HttpPost("edit/{id:guid}")]
public async Task<IActionResult> Edit(Guid id, Product product)
{
    await _services.CreateUpdate(product);

    return RedirectToAction(nameof(All));
}

[HttpGet("delete/{id:guid}")]
public async Task<IActionResult> Delete(Guid id)
{
    await _services.Delete(id);

    return RedirectToAction(nameof(All));
}
}

```

RatingsController.cs

```

using System;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Vlogo.Orders.Application.Services;
using Vlogo.Ratings.Application.Services;
using Vlogo.Ratings.Contracts;

```

namespace Vlogo.Suppliers.Host.Controllers

```

{
    public class RatingsController : Controller
    {
        private readonly IRatingService _service;
        private readonly IOOrdersService _ordersService;

        public RatingsController(IRatingService service, IOOrdersService ordersService)
        {
            _service = service;
            _ordersService = ordersService;
        }
    }
}

```

					КП.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

```
public async Task<IActionResult> All()
{
    var all = await _service.GetAll();

    return View(all.ToList());
}

[HttpGet("ratings/rate/{orderId:guid}")]
public async Task<IActionResult> Rate(Guid orderId)
{
    var order = await _ordersService.Get(orderId);

    var rating = new Rating
    {
        Id = Guid.NewGuid(),
        ClientId = order.ClientId,
        SupplierId = order.SupplierId
    };

    return View(rating);
}

[HttpPost("ratings/rate/{orderId:guid}")]
public async Task<IActionResult> Rate(Guid orderId, Rating rating)
{
    await _service.CreateUpdate(new Rating
    {
        Id = Guid.NewGuid(),
        Points = rating.Points,
        Comment = rating.Comment,
        SupplierId = rating.SupplierId,
        ClientId = rating.ClientId,
        OrderId = orderId
    });

    return Redirect("https://localhost:4001");
}
}
```

2 МОДЕЛІ

```

BonusModel.cs
using System;
using System.Collections.Generic;

namespace Vlogo.Suppliers.Host.Models.Bonuses
{
    public class BonusModel
    {
        public Guid Id { get; set; }
        public string Description { get; set; }
        public IList<ShortProductModel> Products { get; set; }
        public int DiscountPercentage { get; set; }
    }
}
CreateBonusModel.cs
using System;
using System.Collections.Generic;

namespace Vlogo.Suppliers.Host.Models.Bonuses
{
    public class CreateBonusModel
    {
        public Guid Id { get; set; }
        public string Description { get; set; }
        public int DiscountPercentage { get; set; }
        public IList<SelectProductModel> Products { get; set; }
    }
}
SelectProductModel.cs
using System;

namespace Vlogo.Suppliers.Host.Models.Bonuses
{
    public class SelectProductModel
    {
        public Guid Id { get; set; }
        public string Name { get; set; }
        public bool IsSelected { get; set; }
    }
}
ShortProductModel.cs
using System;

namespace Vlogo.Suppliers.Host.Models.Bonuses
{
    public class ShortProductModel
    {
        public Guid Id { get; set; }
        public string Name { get; set; }
    }
}
OrderModel.cs
using System;
using System.Collections.Generic;
using Vlogo.Orders.Contracts;
using Vlogo.Suppliers.Contracts.Products;

namespace Vlogo.Suppliers.Host.Models.Orders
{
    public class OrderModel

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

```

    }
    public Guid Id { get; set; }
    public Guid ClientId { get; set; }
    public IDictionary<Product, int> Product2Count { get; set; }
    public double Price { get; set; }
    public OrderStatus Status { get; set; }
}

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

3 ЦЕРБІСН

```

BonusService.cs
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Vlogo.Bonuses.Application.Repositories;
using Vlogo.Bonuses.Contracts;

namespace Vlogo.Bonuses.Application.Services
{
    internal class BonusService : IBonusService
    {
        private readonly IBonusRepository _repository;

        public BonusService(IBonusRepository repository)
        {
            _repository = repository;
        }

        public Task<Bonus> Get(Guid id)
        {
            return _repository.Get(id);
        }

        public Task<IReadOnlyCollection<Bonus>> GetAll()
        {
            return _repository.GetAll();
        }

        public Task CreateUpdate(Bonus product)
        {
            if (product.Id == Guid.Empty) product.Id = Guid.NewGuid();

            return _repository.CreateUpdate(product);
        }

        public Task Delete(Guid id)
        {
            return _repository.Delete(id);
        }
    }
}

IBonusService.cs
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Vlogo.Bonuses.Contracts;

namespace Vlogo.Bonuses.Application.Services

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

```

{
    public interface IBonusService
    {
        Task<Bonus> Get(Guid id);
        Task<IReadOnlyCollection<Bonus>> GetAll();
        Task CreateUpdate(Bonus product);
        Task Delete(Guid id);
    }
}

ServiceCollectionExtensions.cs
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using MongoDB.Driver;
using Vlogo.Bonuses.Application.Repositories;
using Vlogo.Bonuses.Application.Repositories.Mongo;
using Vlogo.Bonuses.Application.Services;
using Vlogo.Bonuses.Application.Settings;

namespace Vlogo.Bonuses.Application
{
    public static class ServiceCollectionExtensions
    {
        public static IServiceCollection AddBonusesApplication(this IServiceCollection services)
        {
            return services
                .AddBonusSettings()
                .AddMongo()
                .AddBonusMongoRepositories()
                .AddBonusServices();
        }

        public static IServiceCollection AddBonusServices(this IServiceCollection services)
        {
            services.AddScoped<IBonusService, BonusService>();

            return services;
        }

        public static IServiceCollection AddMongo(this IServiceCollection services)
        {
            services.AddSingleton<IMongoClient>(provider =>
            {
                var settings = provider.GetRequiredService<BonusSettings>();
                return new MongoClient(settings.MongoConnectionString);
            });

            return services;
        }

        public static IServiceCollection AddBonusMongoRepositories(this IServiceCollection
services)

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

```

    {
        services.AddScoped<IBonusRepository, MongoBonusRepository>();

        return services;
    }

    public static IServiceCollection AddBonusSettings(this IServiceCollection services)
    {
        services.AddSingleton(provider =>
        {
            var config = provider.GetRequiredService<IConfiguration>();

            return config.GetSection("Bonus").Get<BonusSettings>();
        });

        return services;
    }
}

```

IOrdersService.cs

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Vlogo.Orders.Contracts;

namespace Vlogo.Orders.Application.Services
{
    public interface IOrdersService
    {
        Task<Order> Get(Guid id);
        Task<IReadOnlyCollection<Order>> GetAll();
        Task CreateUpdate(Order order);
        Task Delete(Guid id);
    }
}

```

OrdersService.cs

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Vlogo.Notifications.Application.Services;
using Vlogo.Notifications.Contracts;
using Vlogo.Orders.Application.Repositories;
using Vlogo.Orders.Contracts;
using Vlogo.Ratings.Application.Services;

namespace Vlogo.Orders.Application.Services
{
    internal class OrdersService : IOrdersService
    {
        private readonly IOrdersRepository _repository;
        private readonly INotificationsService _notifications;
    }
}

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

```

private readonly IRatingService _ratingService;

public OrdersService(IOrdersRepository repository, INotificationsService notifications,
IRatingService ratingService)
{
    _repository = repository;
    _notifications = notifications;
    _ratingService = ratingService;
}

public Task<Order> Get(Guid id)
{
    return _repository.Get(id);
}

public Task<IReadOnlyCollection<Order>> GetAll()
{
    return _repository.GetAll();
}

public async Task CreateUpdate(Order order)
{
    if (order.Id == Guid.Empty) order.Id = Guid.NewGuid();

    if (order.Status == OrderStatus.Done) await PushRatings(order);

    order.Time = DateTime.Now;

    await _repository.CreateUpdate(order);
}

public Task Delete(Guid id)
{
    return _repository.Delete(id);
}

private Task PushRatings(Order order)
{
    return _notifications.CreateUpdate(new Notification
    {
        Id = Guid.NewGuid(),
        Message = $"Do you like service? Leave points",
        Target = order.ClientId,
        Uri = _ratingService.GetRatingFormUri(order.Id),
        UriDestinationDescription = "Rate us"
    });
}
}
}
ServiceCollectionExtensions.cs
using Microsoft.Extensions.Configuration;

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

```
using Microsoft.Extensions.DependencyInjection;
using MongoDB.Driver;
using Vlogo.Orders.Application.Repositories;
using Vlogo.Orders.Application.Repositories.Mongo;
using Vlogo.Orders.Application.Services;
using Vlogo.Orders.Application.Settings;

namespace Vlogo.Orders.Application
{
    public static class ServiceCollectionExtensions
    {
        public static IServiceCollection AddOrdersApplication(this IServiceCollection services)
        {
            return services
                .AddOrdersSettings()
                .AddMongo()
                .AddOrderMongoRepositories()
                .AddOrderServices();
        }

        private static IServiceCollection AddOrderServices(this IServiceCollection services)
        {
            services.AddScoped<IOrdersService, OrdersService>();

            return services;
        }

        private static IServiceCollection AddMongo(this IServiceCollection services)
        {
            services.AddSingleton<IMongoClient>(provider =>
            {
                var settings = provider.GetRequiredService<OrdersSettings>();
                return new MongoClient(settings.MongoConnectionString);
            });

            return services;
        }

        private static IServiceCollection AddOrderMongoRepositories(this IServiceCollection
services)
        {
            services.AddScoped<IOrdersRepository, MongoOrdersRepository>();

            return services;
        }

        private static IServiceCollection AddOrdersSettings(this IServiceCollection services)
        {
            services.AddSingleton(provider =>
            {
                var config = provider.GetRequiredService<IConfiguration>();
```

```

        return config.GetSection("Orders").Get<OrdersSettings>();
    });

    return services;
}
}
}
IRatingService.cs
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Vlogo.Ratings.Contracts;

namespace Vlogo.Ratings.Application.Services
{
    public interface IRatingService
    {
        Task<Rating> Get(Guid id);
        Task<IList<Rating>> Get(IEnumerable<Guid> ids);
        Task<IReadOnlyCollection<Rating>> GetAll();
        Task CreateUpdate(Rating rating);
        Task Delete(Guid id);
        Uri GetRatingFormUri(Guid orderId);
    }
}
RatingService.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Vlogo.Ratings.Application.Repositories;
using Vlogo.Ratings.Application.Settings;
using Vlogo.Ratings.Contracts;

namespace Vlogo.Ratings.Application.Services
{
    internal class RatingService : IRatingService
    {
        private readonly IRatingRepository _repository;
        private readonly RatingSettings _settings;

        public RatingService(IRatingRepository repository, RatingSettings settings)
        {
            _repository = repository;
            _settings = settings;
        }

        public Task<Rating> Get(Guid id)
        {
            return _repository.Get(id);
        }
    }
}

```

```

    }

    public async Task<IList<Rating>> Get(IEnumerable<Guid> ids)
    {
        var uniqueIds = new HashSet<Guid>(ids);

        var tasks = new List<Task<Rating>>();
        foreach (var id in uniqueIds) tasks.Add(Get(id));

        var result = await Task.WhenAll(tasks);

        return result.Where(x => x != null).ToList() ?? new List<Rating>();
    }

    public async Task<IReadOnlyCollection<Rating>> GetAll()
    {
        return await _repository.GetAll() ?? new List<Rating>();
    }

    public Task CreateUpdate(Rating rating)
    {
        if (rating.Id == Guid.Empty) rating.Id = Guid.NewGuid();

        return _repository.CreateUpdate(rating);
    }

    public Task Delete(Guid id)
    {
        return _repository.Delete(id);
    }

    public Uri GetRatingFormUri(Guid orderId)
    {
        return new Uri(_settings.RatingFormUri.TrimEnd('/') + $"/{orderId}");
    }
}

ServiceCollectionExtensions.cs
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using MongoDB.Driver;
using Vlogo.Ratings.Application.Repositories;
using Vlogo.Ratings.Application.Repositories.Mongo;
using Vlogo.Ratings.Application.Services;
using Vlogo.Ratings.Application.Settings;

namespace Vlogo.Ratings.Application
{
    public static class ServiceCollectionExtensions
    {
        public static IServiceCollection AddRatingsApplication(this IServiceCollection services)

```

					КП.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

```

    {
        return services
            .AddSuppliersSettings()
            .AddMongo()
            .AddMongoRepositories()
            .AddServices();
    }

    public static IServiceCollection AddServices(this IServiceCollection services)
    {
        services.AddScoped<IRatingService, RatingService>();

        return services;
    }

    public static IServiceCollection AddMongo(this IServiceCollection services)
    {
        services.AddSingleton<IMongoClient>(provider =>
        {
            var settings = provider.GetRequiredService<RatingSettings>();
            return new MongoClient(settings.MongoConnectionString);
        });

        return services;
    }

    public static IServiceCollection AddMongoRepositories(this IServiceCollection services)
    {
        services.AddScoped<IRatingRepository, MongoRatingRepository>();

        return services;
    }

    public static IServiceCollection AddSuppliersSettings(this IServiceCollection services)
    {
        services.AddSingleton(provider =>
        {
            var config = provider.GetRequiredService<IConfiguration>();

            return config.GetSection("Rating").Get<RatingSettings>();
        });

        return services;
    }
}
}
}
IProductService.cs
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Vlogo.Suppliers.Contracts.Products;

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

```

namespace Vlogo.Suppliers.Application.Services
{
    public interface IProductService
    {
        Task<Product> Get(Guid id);
        Task<IList<Product>> Get(IEnumerable<Guid> ids);
        Task<IReadOnlyCollection<Product>> GetAll();
        Task CreateUpdate(Product product);
        Task Delete(Guid id);
    }
}

```

ProductService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Vlogo.Suppliers.Application.Repositories;
using Vlogo.Suppliers.Contracts.Products;

```

```

namespace Vlogo.Suppliers.Application.Services
{
    internal class ProductService : IProductService
    {
        private readonly IProductRepository _repository;

        public ProductService(IProductRepository repository)
        {
            _repository = repository;
        }

        public Task<Product> Get(Guid id)
        {
            return _repository.Get(id);
        }

        public async Task<IList<Product>> Get(IEnumerable<Guid> ids)
        {
            var uniqueIds = new HashSet<Guid>(ids);

            var tasks = new List<Task<Product>>();
            foreach (var id in uniqueIds) tasks.Add(Get(id));

            var result = await Task.WhenAll(tasks);

            return result.Where(x => x != null).ToList() ?? new List<Product>();
        }

        public async Task<IReadOnlyCollection<Product>> GetAll()
        {
            return await _repository.GetAll() ?? new List<Product>();
        }
    }
}

```

					КПІ.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

```

    }

    public Task CreateUpdate(Product product)
    {
        if (product.Id == Guid.Empty) product.Id = Guid.NewGuid();

        return _repository.CreateUpdate(product);
    }

    public Task Delete(Guid id)
    {
        return _repository.Delete(id);
    }
}

ServiceCollectionExtensions.cs
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using MongoDB.Driver;
using Vlogo.Suppliers.Application.Repositories;
using Vlogo.Suppliers.Application.Repositories.Mongo;
using Vlogo.Suppliers.Application.Services;
using Vlogo.Suppliers.Application.Settings;

namespace Vlogo.Suppliers.Application
{
    public static class ServiceCollectionExtensions
    {
        public static IServiceCollection AddSuppliersApplication(this IServiceCollection services)
        {
            return services
                .AddSuppliersSettings()
                .AddMongo()
                .AddMongoRepositories()
                .AddServices();
        }

        public static IServiceCollection AddServices(this IServiceCollection services)
        {
            services.AddScoped<IProductService, ProductService>();

            return services;
        }

        public static IServiceCollection AddMongo(this IServiceCollection services)
        {
            services.AddSingleton<IMongoClient>(provider =>
            {
                var settings = provider.GetRequiredService<SuppliersSettings>();
                return new MongoClient(settings.MongoConnectionString);
            });
        }
    }
}

```

```
        return services;
    }

    public static IServiceCollection AddMongoRepositories(this IServiceCollection services)
    {
        services.AddScoped<IProductRepository, MongoProductRepository>();

        return services;
    }

    public static IServiceCollection AddSuppliersSettings(this IServiceCollection services)
    {
        services.AddSingleton(provider =>
        {
            var config = provider.GetRequiredService<IConfiguration>();

            return config.GetSection("Suppliers").Get<SuppliersSettings>();
        });

        return services;
    }
}
```

					КП.ІП-6329.045440.11.13	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“ ____ ” _____ 2020 р.

КОМПЛЕКС ПРОГРАМ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ
ЗАМОВЛЕНЬ ТОВАРІВ ТА ПОСЛУГ (КОМПЛЕКСНА ТЕМА)

Технічне завдання

КП.ІІ-6329.045440.10.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ М.С. Храмченко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Д.М. Шелудько

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2 ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3 ПРИЗНАЧЕННЯ РОЗРОБКИ	5
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
4.1 ВИМОГИ ДО ФУНКЦІОНАЛЬНИХ ХАРАКТЕРИСТИК.....	6
4.2 ВИМОГИ ДО НАДІЙНОСТІ	7
4.3 УМОВИ ЕКСПЛУАТАЦІЇ	7
4.4 ВИМОГИ ДО СКЛАДУ І ПАРАМЕТРІВ ТЕХНІЧНИХ ЗАСОБІВ.....	7
4.5 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ ТА ПРОГРАМНОЇ СУМІСНОСТІ.....	7
4.6 ВИМОГИ ДО МАРКУВАННЯ ТА ПАКУВАННЯ	8
4.7 ВИМОГИ ДО ТРАНСПОРТУВАННЯ ТА ЗБЕРІГАННЯ	8
4.8 СПЕЦІАЛЬНІ ВИМОГИ	8
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	9
5.1 ПОПЕРЕДНІЙ СКЛАД ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ.....	9
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ	10
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	11
7.1 Види випробувань.....	11

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Корпоративний бот для ІТ-компаній

Галузь застосування: Малі та середні за розміром компанії у сфері ІТ

Наведене технічне завдання поширюється на розробку корпоративного бота для ІТ-компаній, котра використовується для автоматизації рутинних процесів та призначена для малих та середніх за розміром компаній у сфері ІТ. Користувачами можуть бути будь-які працівники та розробники ІТ-компаній.

					КПІ.ІП-6305.045490.02.91	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки корпоративного бота для ІТ-компаній є завдання на дипломне проєктування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

					КПІ.ІП-6305.045490.02.91	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для компаній в сфері ІТ, та інших компаній що використовують в якості засобу для комунікації Slack.

Метою розробки є автоматизація рутинних процесів в ІТ-компаніях за допомогою взаємодії з ботом в Slack, а саме оформлення відпусток чи відгулів і проведення опитувань серед співробітників, з можливістю для розробників додавати нові модулі для автоматизації інших процесів компанії.

					КПІ.ІП-6305.045490.02.91	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1.1 Для користувача:

- створення запитів на отримання відпусток чи відгулів;
- перегляд власної історії відпусток та відгулів;
- перегляд доступних днів кожного типу відгулів;
- отримання сповіщень якщо хтось з користувачів у відпустці;
- створення опитувальників та розсилання іншим користувачам;
- нагадування учасникам опитувальника що треба заповнити відповіді;
- заповнення відповідей в опитувальниках;

4.1.1.2 Для адміністратора системи:

- схвалення та відхилення запитів на отримання відпусток чи відгулів;
- перегляд історії відпусток та відгулів всіх користувачів;
- додавання користувачам доступних днів по кожному типу відгулів;

4.1.1.3 Для розробника:

- можливість розширення функціоналу для автоматизації.

4.1.2 Розробку виконати на платформі Linux

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Не висуваються.

4.3.2 Обслуговування та обслуговуючий персонал.

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на серверах з операційною системою Ubuntu.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору

Процесори Intel i3 або кращі.

4.4.2.2 Об'єм ОЗП

512 Мб, або більше.

4.5 Вимоги до інформаційної та програмної сумісності

– Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix;

– вхідні дані повинні бути представлені в наступному форматі: тексти повідомлень Slack;

					КПІ.ІП-6305.045490.02.91	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- Результати повинні бути представлені в наступному форматі: повідомлення в Slack та колекції MongoDB у форматі BSON;
- Програмне забезпечення повинно надавати REST API для підключення нових модулів.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати установочну версію програмного забезпечення.

					КПІ.ІП-6305.045490.02.91	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

а) Супроводжувальна документація:

- 1) пояснювальна записка;
- 2) керівництво користувача;
- 3) керівництво програміста;
- 4) програма та методика тестування.

б) Довідникова документація:

- 1) всі розроблені API повинні бути задокументовані за специфікацією OpenAPI;
- 2) програмне забезпечення повинно мати вбудовану довідку.

в) Графічна документація:

- 1) схема структурна бізнес-процесів.
- 2) схема структурна розгортання.
- 3) креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	15.03.2020	
2.	Розробка технічного завдання	25.03.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	29.03.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	20.04.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	30.04.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	24.05.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проєкту	17.05.2020	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	24.05.2020	Технічна документація

Змн.	Арк.	№ докум.	Підпис	Дата

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

					КПІ.ІП-6305.045490.02.91	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		