

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КЕРУВАННЯ ЗАМОВЛЕННЯМИ В
ЗАКЛАДІ ГРОМАДСЬКОГО ХАРЧУВАННЯ
ORDER MANAGEMENT SOFTWARE IN A MASS CATERING
ESTABLISHMENT

Виконав: студент(ка) 4 курсу, групи КНТ-218сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Назаренко О.Ю.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Назаренка Олександра Юрійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення керування замовленнями в закладі громадського харчування
Order Management Software in a Mass Catering Establishment

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 17 травня 2021 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 10 ” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис) Назаренко О.Ю.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 85 с., 30 рис., 1 табл., 3 дод., 14 джерел.

ЗАКЛАД ГРОМАДСЬКОГО ХАРЧУВАННЯ, ОБСЛУГОВУВАННЯ КЛІЄНТІВ, РЕСТОРАН, КАВ'ЯРНЯ, КЕРУВАННЯ ЗАМОВЛЕННЯМИ, MODEL-VIEW-VIEWMODEL.

Об'єкт роботи – процес керування замовленнями в закладі громадського харчування. Предмет роботи – програмне забезпечення керування замовленнями в закладі громадського харчування.

Мета роботи – розроблення програмного забезпечення керування замовленнями в закладі громадського харчування для забезпечення консолідованого доступу до замовлень зі сторони всіх учасників процесу та підтримки реалізації даного процесу.

Матеріали, методи та технічні засоби: мова програмування C#, шаблон проєктування Model-View-ViewModel.

Результати. Виконано аналіз проблеми керування замовленнями в закладі громадського харчування, виконано проєктування програмного забезпечення на основі діаграм прецедентів та діаграми послідовностей. Описано прийняті під час реалізації рішення. Розроблено програмне забезпечення.

Висновки. Розроблене програмне забезпечення направлене на керування замовленнями на всіх етапах їх життєвого циклу.

Галузь використання. Керування діяльністю щодо роботи з замовленнями закладу громадського харчування невеликого розміру, включаючи кав'ярню, ресторан або будь-який інший заклад громадського харчування з власною кухнею і залом обслуговування.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 85 pages, 30 figures, 1 table, 3 appendixes, 14 sources.

CATERING INSTITUTION, CUSTOMER SERVICE, RESTAURANT, CAFE, ORDER MANAGEMENT, MODEL-VIEW-VIEWMODEL.

The object of the work is a process of order management in a catering establishment. The subject of the work is order management software in a catering establishment.

The purpose of the work is to develop order management software in a catering establishment to ensure consolidated access to orders by all participants in the process and to support the implementation of this process.

Materials, methods and technical means: C# programming language, Model-View-ViewModel design template.

Results. The analysis of the problem of order management in a public catering establishment is performed. The software design is performed on the basis of use case and sequence diagrams. The decisions made during the development of the software are presented. The software has been developed.

Conclusions. The developed software is aimed at managing orders at all stages of their life cycle.

Field of usage. Management of the activities of ordering in a small catering establishment, including a cafe, a restaurant or any other catering establishment with its own kitchen and service hall.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Постановка проблеми керування замовленнями в закладі громадського харчування.....	10
1.2 Огляд програмних аналогів.....	13
1.3 Висновки за розділом 1	18
2 Проєктування програмного забезпечення	20
2.1 Вибір мови програмування	20
2.2 Вимоги до програмного забезпечення	21
2.3 Проєктування бази даних	27
2.4 Визначення архітектури програмного забезпечення.....	31
2.5 Висновки за розділом 2	32
3 Основні рішення щодо реалізації компонентів системи.....	34
3.1 Структура розробленого програмного забезпечення	34
3.2 Послідовність виконання програми	37
3.3 Опис розроблених структурних одиниць	39
3.4 Висновки за розділом 3	43
4 Експлуатація, тестування та експериментальне дослідження програми	44
4.1 Призначення програми	44
4.2 Умови виконання програми	44
4.3 Робота з програмою	44
4.4 Висновки за розділом 4	53
Висновки	54
Перелік джерел посилання	56
Додаток А Технічне завдання	58
Додаток Б Текст програми	63
Додаток В Слайди презентації.....	78

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVVM	– Model-View-ViewModel;
XAML	– eXtensible Application Markup Language;
БД	– база даних.

ВСТУП

Ресторани та заклади громадського харчування відіграють життєво важливу роль в економіці, створюючи тисячі робочих місць та підтримуючи одночасно декілька інших галузей, включаючи виробництво продуктів, доставку страв з закладів додому тощо, при цьому згідно з дослідженнями Національної асоціації ресторанів, обсяг продажів даного господарства лише у 2019 році досяг 863 мільярдів доларів, але окрім такого величезного обсягу коштів варто відзначити, що ресторани, пекарні та кафе відіграють життєво важливу роль у структурі суспільства (вони забезпечують споживачів їжею, допомагають влаштовувати особливі випадки та допомагають визначити культурний склад громади, міста чи міста) – всі ці фактори також є причиною того, чому так багато підприємців приваблює ідея відкрити ресторан, але на противагу цьому виступають інші цифри, які також потрібно враховувати: за підрахунками, 60% закладів закриваються протягом першого року, а 80% припиняють свою діяльність протягом п'яти років, тому для того щоб керувати успішним закладом, власники та менеджери повинні чітко розуміти основні принципи управління обслуговуванням [1].

Через надзвичайно насичений графік і зосередженість на професійній та особистій роботі, сучасне покоління не бажає витратити час на чергу в ресторані, як то зайняття місця чи отримання замовленого меню, часто робітники збираються в ресторанах та інших закладах під час перерви, яка обмежена, тому вони не вважають за краще чекати в черзі з будь-якої причини, тоді програмне забезпечення з віддаленим доступом допоможе такому клієнту заощадити свій цінний час, замовляючи їжу без будь-яких додаткових втручань та затримок, для цього при розробці проекту потрібно враховувати:

- замовлення їжі протягом обмеженого періоду часу без будь-якого втручання;
- забезпечення більшої зручності та вибору для клієнтів;
- автоматизована система замовлення та виставлення рахунків [2].

Метою роботи є розроблення програмного забезпечення керування замовленнями в закладі громадського харчування для забезпечення консолідованого доступу до замовлень зі сторони всіх учасників процесу та підтримки реалізації даного процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка проблеми керування замовленнями в закладі громадського харчування

Система керування замовленнями в закладі громадського харчування – це багатогранне програмне забезпечення, яке допомагає впорядкувати роботу закладу та забезпечує автоматизацію, щоб заклад міг турбуватися про те, що найважливіше, – готувати їжу, така система надзвичайно допомагає всім типам закладів громадського харчування, починаючи від ресторанів, пекарень, барів, кафе, бістро та підприємств доставки [3].

Створення власної системи допомагає втримувати бізнес безпосередньо на своїй платформі, при цьому наявність власних активів має декілька переваг:

- збережена комісія, яку потрібно було б заплатити стороннім агрегаторам;
- кожне замовлення приносить більше заощадженого прибутку;
- можливість продемонструвати та продати свій унікальний бренд та відокремитись від конкурентів;
- це дозволяє збирати інформацію про клієнтів – додатково дозволяючи проводити власні кампанії ремаркетингу і змушувати клієнтів повертатися за новими послугами;
- надає неоціненний електронний ресурс [3].

Підсистема бронювання столиків має важливе значення в даному сегменті, мета її використання полягає в тому, щоб мати можливість приймати бронювання столиків з самого застосунку та обробляти всі бронювання столиків з однієї інформаційної панелі: можливість пильно стежити за списком гостей та часом їх очікування є важливою в закладі, де можна повечеряти, адже наприклад, завжди потрібно знати, коли прийдуть лояльні клієнти, щоб підготувати для них особливий прийом [3].

Загалом система керування закладом громадського харчування поєднує в собі все, що добре в традиційних системах точок продажу, з інструментами, які приймають бронювання столиків, впорядковують управління запасами, обробляють рахунки, надають ефективну аналітику, а також допомагають у маркетинговій діяльності, надають програми лояльності та створення присутності в Інтернеті, окрім того також безперебійно працює з існуючими системами ресторанних технологій (програмним забезпеченням для бухгалтерії та управління співробітниками) і використовує відкриті прикладні бібліотеки, що дозволяють інтегруватися з будь-яким стороннім інструментом, таке програмне забезпечення допомагає ресторанам:

- скоротити час обробки замовлення;
- автоматизувати надлишкову роботу;
- забезпечити чудовий досвід клієнтів;
- визначити прибуток та витрати [4].

Усі успішні закладки громадського харчування дотримуються декількох фундаментальних правил:

- належний етикет обслуговування: хоча звичайно, що основна частина діяльності ресторану, природно, буде зосереджена на їжі, але це лише один із аспектів загального досвіду користувача, отриманого від ресторану, це означає, що керівники ресторанів та персонал повинні бути уважними, щоб надати якісне обслуговування в той момент, коли клієнт проходить через двері, це задає тон усьому досвіду споживачів;

- не можна змушувати клієнтів чекати: коли клієнту доводиться чекати декілька хвилин, перш ніж він може замовити закуски або перший напій, не має значення, чи подають йому найкращий стейк чи коктейль у місті, це вже не змінить негативне враження, відповідно якщо це трапиться, буде важко повернути розчарованих клієнтів, навіть якщо їм подобається їжа ресторану, їм, швидше за все, буде що сказати про якість послуг, які вони отримали, задля цього потрібно враховувати наступне: потрібно мати достатньо персоналу під час обіду або вечері, щоб гарантувати, що клієнти не будуть чекати занадто

довго, коли клієнт замовляє продукт, який готується тривалий час, потрібно бути готовим визначити його очікування, бувають випадки, коли несподівані проблеми спричиняють затримки, тому коли це трапиться, обов'язково потрібно повідомити про це та попросити вибачення у клієнта, потрібно визначити швидкість обслуговування (час, який ресторан витрачає на обслуговування клієнта);

– потрібно негайно вирішувати проблеми: третім ключем до надання якісного обслуговування є ефективне вирішення скарг та проблем клієнтів, навіть бездоганно керований ресторан в якийсь момент зіткнеться з проблемами з клієнтами, рішення полягає в тому, щоб зрозуміти розчарування клієнтів, також не менш важливо швидко вирішувати проблеми, адже розчарований клієнт стане більш роздратованим, лише якщо йому доведеться почекати і просунутися вгору по ланцюжку управління, для цього потрібно ніколи не звинувачувати клієнта (навіть якщо ситуація поза контролем ресторану, завжди необхідно співчувати, щоб максимально розрядити ситуацію), запитати клієнта, в чому його проблема, і уважно вислухати, залишатися спокійними, навіть якщо клієнт виявляється нерозумним, спробувати домовитись про результат, прийнятний для всіх сторін, повідомити клієнту, які персонал вживає заходи;

– заохочення клієнтів ділитися відгуками: через форми зворотного зв'язку з клієнтами, платформи онлайн-огляду, такі як Google і Yelp, або соціальні мережі, клієнтам має бути де поділитися своїми думками – хорошими, поганими та всіма проміжними, заохочення споживачів ділитися своїми відгуками та пропозиціями показує, що ресторан цінує їхні думки, чого може бути достатньо, щоб повернути їх, у свою чергу для менеджерів зворотний зв'язок – це те, як вони можуть виявити, чи ресторан щось робить правильно, або є критична проблема, коментарі клієнтів можуть надати глибоке розуміння областей та процесів, які можна вдосконалити, створивши кращі страви;

– використання технологій: сучасні клієнти розраховують на можливість замовлення їжі та бронювання столиків через мережу, включення технологій у діяльність ресторану може допомогти персоналу стати більш ефективним, одночасно надаючи споживачам зручний спосіб взаємодії із послугами ресторану, прикладами ресторанних технологій, які шукають клієнти, є: замовлення – надання клієнтам можливості перегляду меню та підтвердження замовлень на своїх пристроях, це також дозволяє клієнтам замовляти їжу заздалегідь, а це означає, що їжа може бути подана відразу після того, як клієнт з'явився, бронювання столиків – клієнти оцінять можливість не просто забронювати місце, а й вибрати, який стіл вони хочуть [1].

Система керування замовленнями складається з монітора (пристрою), що дозволяє клієнтам вибирати бажану їжу за допомогою меню, що відображається на екрані, зроблений вибір буде надіслано кухонному помічнику, екран комп'ютера тоді може бути розташований як віддалено, так і на кожному столі, щоб клієнти могли робити замовлення, інтерактивний користувацький інтерфейс інтегрований із системою, що дозволяє клієнтам отримувати помічника в режимі реального часу, за такого підходу час, необхідний клієнтам для отримання допомоги, щоб підтвердити своє замовлення, клієнтам потрібно натиснути кнопку підтвердження замовлення, і замовлення буде надіслано кухарю, коли кухонний комп'ютер (виконує роль сервера) отримає замовлення, він надішле повідомлення про підтвердження на комп'ютер клієнтів, щоб клієнти знали, що їх замовлення готуються, сервер повідомляє кухонного помічника про замовлення клієнтів, кухонний асистент у свою чергу дає наказ кухареві готувати [2].

Таким чином, розроблюване програмне забезпечення повинно забезпечити керування всім ланцюжком виконання замовлення, від його формування до відгуку клієнта про виконане замовлення.

1.2 Огляд програмних аналогів

У даному підрозділі потрібно розглянути, як ці можливості реалізуються в реально використовуваних на практиці програмних застосунків.

Toast (рис. 1.1) представляє собою цілу платформу, що підходить для невеликих закладів (кафе, пекарні тощо), хоча також може впоратися практично з будь-яким рестораном із повним набором послуг, дані синхронізуються автоматично в усіх місцях, власнику мережі або франшизи потрібно використовувати одне і те ж ім'я користувача та пароль, тому Toast також є надійним варіантом для франшиз [5] - [6].

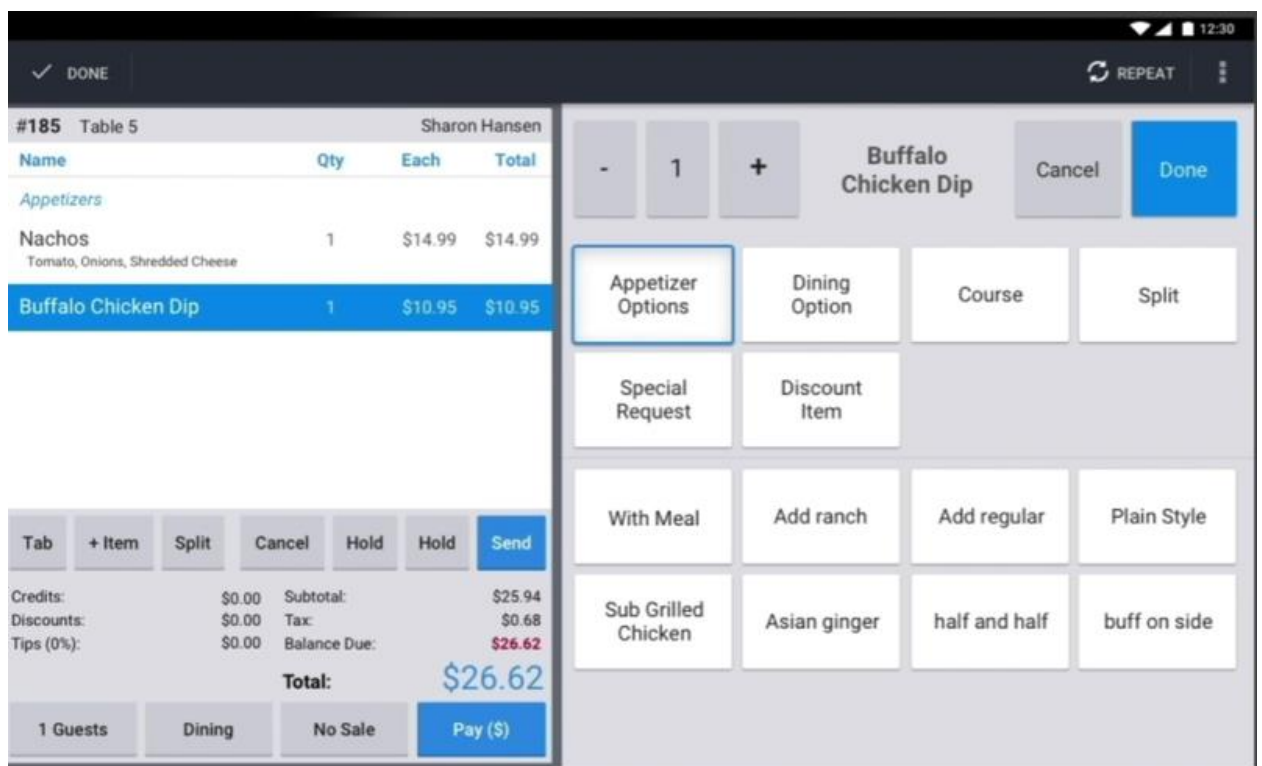


Рисунок 1.1 – Застосунок Toast [7]

До основних функцій програмного забезпечення Toast належать:

– замовлення: за допомогою Toast персонал може приймати замовлення двома різними способами, використовуючи портативні планшети або за допомогою поворотних терміналів, портативні планшети Toast можуть отримати доступ до повноцінної програми (включаючи звіти), безпосередньо з планшета співробітники служби очікування можуть приймати замовлення,

надсилати замовлення та вказівки на кухню, а також керувати вкладками та способами оплати своїх клієнтів з будь-якої точки ресторану;

– створення меню: меню у даному випадку відповідає будь-якому фізичному меню, яке можна використовувати в ресторані, і його можна розділити так само, як звичайне меню, на сніданок, обід, вечерю, коктейлі, десерти тощо, кожне окреме меню в системі Toast може надалі бути розбите на групи (у меню вечері це можуть бути закуски, салати, закуски тощо), потім кожна група містить фактичні пункти меню, елементи можуть бути додатково налаштовані за допомогою будь-якої кількості модифікаторів та субмодифікаторів, це дозволяє застосунку бути корисним для піцерій, сандвічевих магазинів та інших ресторанів, які регулярно роблять їжу на замовлення;

– цінові позиції меню: програма дозволяє скласти стратегію для оцінки вартості елементів меню та створити правила ціноутворення, які можна застосувати до цілих груп, різних розмірів елементів або до товарів, які потребують спеціального ціноутворення на основі меню, в якому вони знаходяться;

– звітність: доступна широка різноманітність звітів – від простих звітів про продаж до звітів про працю, звітів про товари тощо, звіти оновлюються в режимі реального часу, підсумковий звіт відображає чистий дохід, загальну кількість гостей, час обслуговування за столом та розбивку на всі види послуг та способи оплати;

– збір даних про клієнтів: система Toast має велику внутрішню звітність, але вона також має інтегровану систему управління взаємовідносинами з клієнтами, яка працює для збору контактної інформації клієнтів та історії замовлень, за допомогою звіту про клієнтів можна побачити, які клієнти відвідували ресторан найбільше або витратили найбільшу суму грошей за будь-який вибраний час, потім можна детальніше ознайомитися, вибравши окремих клієнтів і переглянувши детальну інформацію про середні

чеки людини, дні з моменту останнього відвідування та найчастіше замовлені пункти меню;

- кухонна система: дана система має кольорове кодування і може містити час для кожного замовлення, з іншого боку, сервери можуть отримувати сповіщення, коли замовлення готове;

- Toast Go – це повністю оптимізований та інтегрований застосунок, який допомагає швидко та точно отримувати замовлення на кухню, він також дозволяє серверам покращити зв'язок з клієнтами та надає можливість вносити зміни миттєво;

- програма полегшує роботу клієнтів, дозволяючи користувачам замовляти самостійно, зменшуючи кількість помилок та мінімізуючи кількість контактних осіб, також має функцію швидкого повторного замовлення для постійних клієнтів, які часто мають однакове замовлення [6]-[7].

При величезній кількості функцій загалом всієї платформи Toast потрібно відзначити, що це доволі коштовне програмне забезпечення:

- підписка починається від 75 доларів за термінал на місяць;

- доповнення коштують додатково (додаткові програмні модулі включають Toast Payroll&Team Management, Toast Online Ordering, Toast Marketing, Toast Loyalty, Kitchen Display System, Kiosk, Toast Inventory, Multi-Location Management), тобто велика кількість тих функцій, що були перелічені вище, вимагають додаткової сплати;

- окрім програмного забезпечення Toast потребує відповідного апаратного забезпечення, вартість якого починається з 999 доларів за початковий набір, при цьому теоретично є можливість використовувати власне обладнання, хоча Toast настійно не рекомендує його використовувати і не може надати підтримку відповідному обладнанню, при цьому віддалена установка починається з 499 доларів, що є знижкою [6].

Ще одним програмним аналогом є TouchBistro [8] (рис. 1.2).

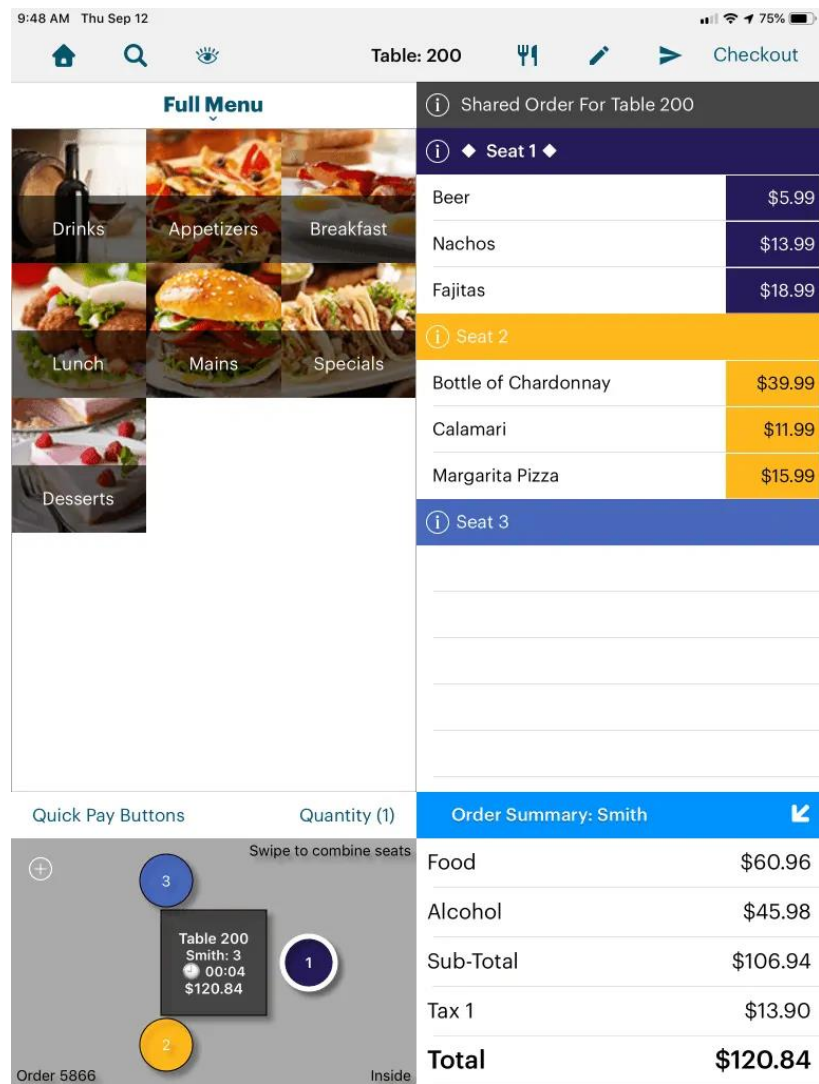


Рисунок 1.2 – Застосунок TouchBistro [9]

TouchBistro розроблений для будь-якого бізнесу, пов'язаного з закладами громадського харчування (ресторани, кафе, бари тощо), він є успішним застосунком для продажу, який при цьому працює виключно на iPad, синхронізованих через локальну мережу, відповідно використання такої системи потребує придбання відповідного і доволі вартісного обладнання, система підтримує вибір столів та місць, підбір меню, прийняття замовлень, бездротову передачу замовлень на кухню, розділені вкладки та щоденні звіти про продажі, мобільність дозволяє ресторанам виконувати кожне завдання з більшою легкістю та швидкістю, персонал може більше часу проводити з клієнтами, створюючи позитивні враження від обіду, при цьому забезпечується виконання наступних функцій:

– прийом замовлень: персонал може приймати замовлення клієнтів за столом за допомогою програми, при цьому можна надіслати замовлення на будь-який принтер компанії, надсилати замовлення на кухню можна відразу, можна створити вкладки, що відповідають налаштуванням бару чи пабу, елементи можуть ділитися цілою таблицею, переміщуватися від одного гостя до іншого або редагуватися, можна робити нотатки або робити спеціальні умови для кожного замовлення, переглянути всі поточні замовлення відразу;

– управління столиками: можна сконструювати макет розташування столиків на основі шаблонів або з нуля, окрім того існуючий макет можна відредагувати в будь-який час, наприклад, об'єднуючи столи для великих вечірок, можна призначати місця кожному клієнтові та поєднувати замовлення з призначеннями місць, менеджери можуть безпосередньо призначати офіціантів, коли відкривають стіл від їх імені;

– управління меню: можна створити власне меню з окремими категоріями, кожен пункт меню містить повноекранну фотографію, яку можна завантажити або зробити безпосередньо за допомогою функції камери iPad, можна переключатися між меню сніданку, обіду та вечері [9].

1.3 Висновки за розділом 1

Розглянуто переваги створення системи керування замовленнями в закладі громадського харчування, базові складові і особливості такого програмного забезпечення, принципи організації успішного обслуговування і способи перенесення цих принципів на програмне забезпечення.

Розглянуто основні складові та функціональні можливості програмної і апаратної платформи Toast, яка окрім наявності великої кількості засобів потребує сплати достатньо великих коштів, та програмного застосунку TouchBistro, який потребує використання iPad і відповідно його придбання.

У даній дипломній кваліфікаційній роботі мають бути розв'язано наступні завдання:

- визначення вимог до програмного забезпечення керування замовленнями закладу громадського харчування;
- вибір мови і засобів проєктування і розробки;
- проєктування програмного забезпечення;
- реалізація і налагодження програмного забезпечення.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

Програмне забезпечення керування замовленнями в закладі громадського харчування потребує створення повноцінної платформи, на якій будуть ґрунтуватися десктопний застосунок та вебзастосунок, за допомогою якого може надаватися універсальний доступ, а також потенційно програмні застосунки для мобільних операційних систем. Однак розробляти всю платформи одночасно недоцільно і дуже витратно. Перши кроком, який і має бути реалізовано у даній роботі, було визначено створення засобів роботи сервера на основі системи управління БД та програмного застосунку для операційної системи Windows.

В якості показників для порівняння мов програмування було обрано:

- об'єктно-орієнтовані засоби розробки: дана парадигма є з одного боку найпопулярнішою і відповідно в якості базового засобу вимагається в даному рішенні для його розробки, а з іншого боку об'єктно-орієнтованість підвищує зручність повторного використання коду, що важливо для уникнення написання зайвого коду, адже в даному випадку обмежений дуже час розробки;
- простота створення коду: простота створення коду впливає на зручність роботи розробника, а в результаті і на час, витрачений на розробку;
- робота з БД: основна функціональність роботи в даній сфері пов'язана з доступом до даних, відповідно зручність роботи з БД є дуже важливим фактором, наприклад, відслідковування стану замовлення потребує постійного доступу до пакунку та зміни однієї з характеристик замовлення, стану, на відповідне значення;
- робота над графічним користувацьким інтерфейсом під керуванням операційної системи Windows.

За даними критеріями було проведено порівняння двох основних мов розробки десктопних застосунків для операційної системи Windows. Аналіз

наведено в таблиці 2.1 C# [10]-[11] та C++ [12].

Таблиця 2.1 – Вибір мови програмування

Показник	C#	C++
Об'єктно-орієнтовані засоби розробки	Так	Так
Простота створення коду	Висока	Нижча
Робота з БД	Інтерфейси роботи з різними системами керування БД, наявні вбудовані засоби роботи з MS SQL Server	Інтерфейси роботи з різними системами керування БД
Робота над графічним користувацьким інтерфейсом	Проста за допомогою середовища Microsoft Visual Studio	Має певні складнощі через Microsoft Visual Studio, потребує більше часу через фреймворк Qt

Розробку було вирішено виконувати за допомогою мови програмування C# через загальну перевагу за розглянутими показниками.

2.2 Вимоги до програмного забезпечення

Для реалізації у програмному забезпеченні на основі виконаного огляду програмних аналогів у розділі 1 було визначено наступні функціональні вимоги:

– створення меню з категоріями та різними варіантами для різного часу дня і різних подій;

- створення страв з зображеннями;
- змінювання страв у меню;
- вибір активного варіанту меню;
- перегляд меню;
- бронювання столику;
- перегляд власних бронювань;
- перегляд заброньованих столів у закладі за заданий час;
- скасування бронювання столику;
- створення замовлення;
- створення замовлення на заданий час;
- скасування замовлення;
- створення замовлення на основі статистики попередніх замовлень;
- перегляд власних замовлень;
- перегляд стану невиконаних замовлень;
- перегляд замовлень для приготування;
- перегляд замовлень, готових до видачі;
- приготування страви;
- видача замовлення;
- внесення даних про сплату замовлення;
- внесення відгуків про виконання замовлення;
- перегляд відгуків клієнтів;
- визначення особливостей виконання замовлень;
- авторизація і реєстрація облікових записів.

З програмним забезпеченням може працювати:

- клієнт;
- офіціант;
- менеджер;
- кухар.

Сценарії взаємодії клієнта (рис. 2.1) з програмою включають:

- перегляд меню;

- бронювання столику;
- перегляд власних бронювань;
- скасування бронювання столику;
- створення замовлення;
- створення замовлення на заданий час;
- скасування замовлення;
- створення замовлення на основі статистики попередніх замовлень;
- перегляд власних замовлень;
- перегляд стану власних невиконаних замовлень;
- внесення відгуків про виконання замовлення;
- авторизація і реєстрація облікових записів.



Рисунок 2.1 – Діаграма прецедентів роботи клієнта з програмою

Сценарії взаємодії офіціанта з програмою включають:

- перегляд меню;
- бронювання столику;
- перегляд заброньованих столів у закладі за заданий час;
- скасування бронювання столику;
- створення замовлення;
- створення замовлення на заданий час;
- скасування замовлення;
- перегляд стану невиконаних замовлень;
- перегляд замовлень, готових до видачі;
- видача замовлення;
- внесення даних про сплату замовлення;
- визначення особливостей виконання замовлень;
- перегляд відгуків клієнтів;
- авторизація облікових записів.

Сценарії взаємодії кухаря з програмою (рис. 2.2) включають:

- перегляд замовлень для приготування;
- приготування страви;
- авторизація облікових записів.

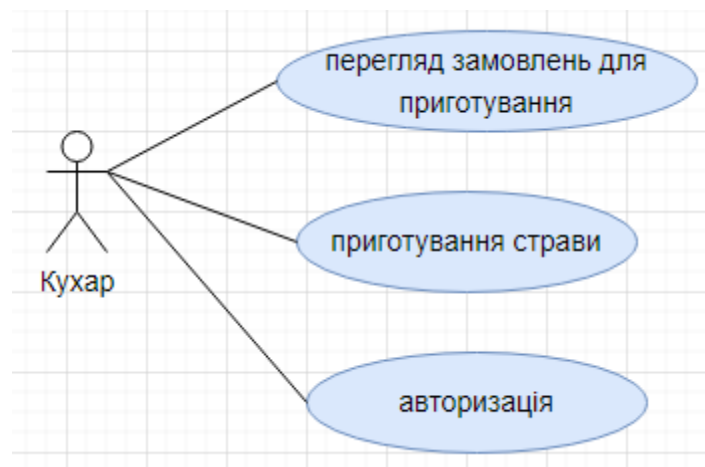


Рисунок 2.2 – Діаграма прецедентів роботи кухаря з програмою

Діаграма прецедентів роботи офіціанта з програмою представлена на рис. 2.3.



Рисунок 2.3 – Діаграма прецедентів роботи офіціанта з програмою

Діаграма прецедентів роботи менеджера з програмою представлена на рис. 2.4.

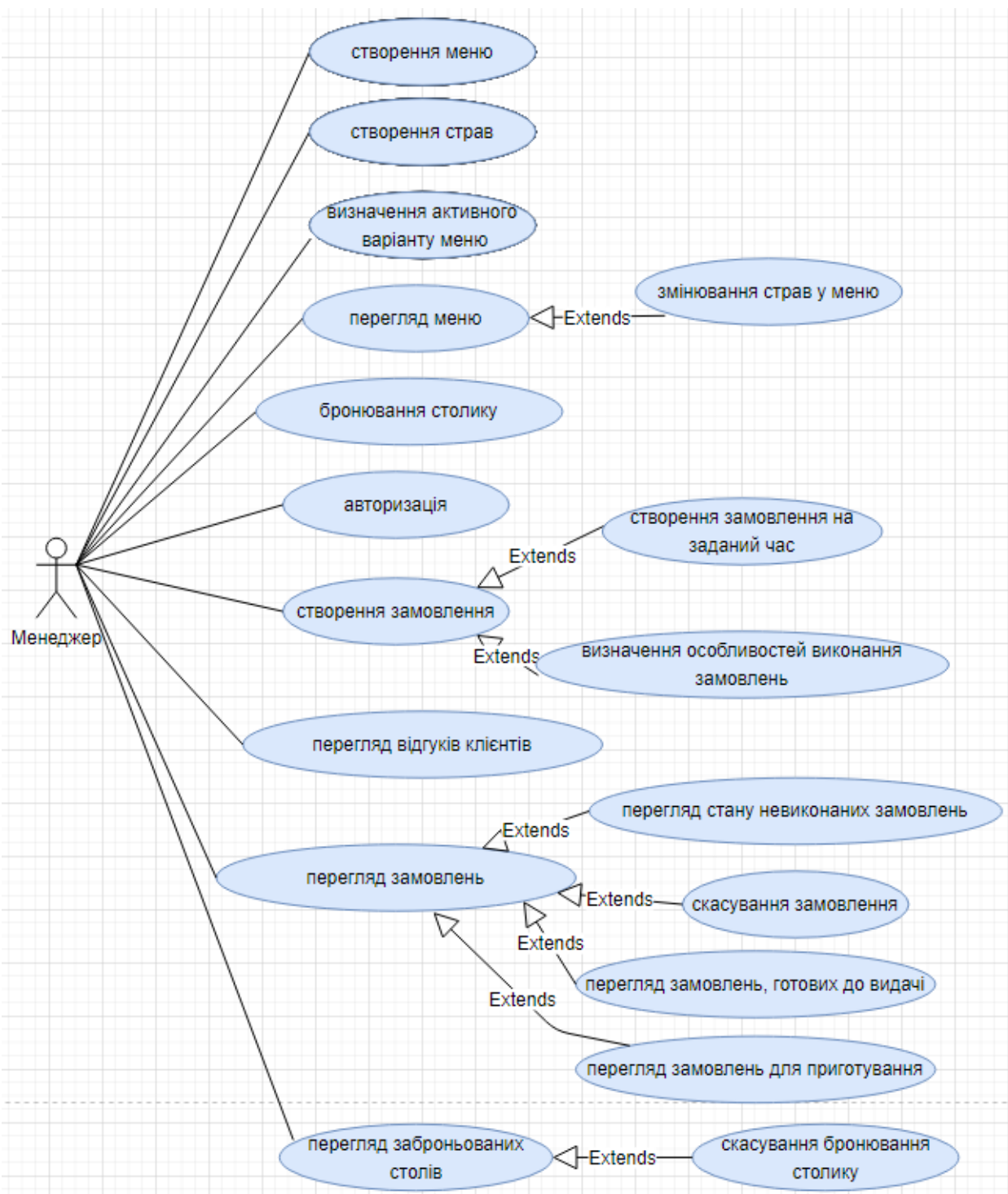


Рисунок 2.4 – Діаграма прецедентів роботи менеджера з програмою

Сценарії взаємодії менеджера з програмою включають:

- створення меню з категоріями та різними варіантами для різного часу дня і різних подій;
- створення страв з зображеннями;
- змінювання страв у меню;

- вибір активного варіанту меню;
- перегляд меню;
- бронювання столику;
- перегляд заброньованих столів у закладі за заданий час;
- скасування бронювання столику;
- створення замовлення;
- створення замовлення на заданий час;
- скасування замовлення;
- перегляд стану невиконаних замовлень;
- перегляд замовлень для приготування;
- перегляд замовлень, готових до видачі;
- перегляд відгуків клієнтів;
- визначення особливостей виконання замовлень;
- авторизація облікових записів.

2.3 Проєктування бази даних

Для проєктування і реалізації БД було обрано використовувати систему управління БД Microsoft SQL Server [13], оскільки вона є традиційним відповідним засобом для застосунків мовою програмування C#.

Основні сутності предметної області:

- User – користувач;
- Order – замовлення;
- Dish – страва;
- Reservation – бронювання столу;
- Table – стіл;
- Menu – меню (відповідний варіант меню, тобто меню сніданків, обідів, вечерь, універсальне, меню на якесь свято);
- Category – категорія страви в меню.

Оскільки сутності Dish і Order пов'язані зв'язком багато-до-багатьох, адже кожна страва може входити в різні замовлення, але і кожне замовлення може складатися з багатьох страв, то для реалізації даного зв'язку було створено таблицю OrderDishes.

Сутності Dish і Menu пов'язані зв'язком багато-до-багатьох, адже кожна страва може входити в різні варіанти меню, але і кожен варіант меню включає багато страв страв, тому для реалізації даного зв'язку було створено таблицю MenuDishes.

Таблиця User включає такі поля:

- id – номер;
- name – ім'я користувача;
- login – логін у системі;
- password – пароль у системі;
- phone – номер телефону для зв'язку;
- email – електронна пошта;
- role – роль в системі (клієнт, офіціант, кухар або менеджер);
- status – стан (активний або неактивний).

Таблиця Category включає такі поля:

- id – номер;
- title – назва категорії.

Таблиця Table включає такі поля:

- id – номер;
- number – символічний номер столу (код);
- places – кількість місць за столом;
- status – стан (активний, неактивний, тобто такий, який не можна використовувати, або позначення місця на виніс).

Таблиця Order включає такі поля:

- id – номер;
- reservationId – номер бронювання столу (може бути порожнім);
- client – номер запису клієнта;

- waiter – номер запису офіціанта, який обслуговував замовлення;
- chef – номер кухаря, який приготував замовлення;
- tableId – номер столу, за яким знаходився клієнт (за одним столом може знаходитися декілька замовлень, номер столу може не співпадати з номером столу в бронюванні);
- status – стан замовлення;
- price – сумарна вартість замовлення;
- paid – сплачена сума (клієнт може сплатити чайові);
- from – початок обслуговування (за попереднього замовлення визначає, коли має бути приготовано замовлення);
- to – завершення обслуговування;
- feedback – відгук клієнта;
- description – опис замовлення для передачі особливостей замовлення від офіціанта (доступно тільки офіціанту та менеджеру) до кухаря.

Таблиця Dish включає такі поля:

- id – номер страви;
- title – назва страви;
- description – опис;
- image – зображення страви;
- categoryId – номер категорії;
- price – вартість страви.

Таблиця Menu включає такі поля:

- id – номер меню;

Таблиця OrderDishes включає такі поля:

- id – номер запису;
- orderId – номер замовлення;
- menudishId – номер страви з меню;
- status – стан приготування страви;
- quantity – кількість.
- title – назва меню.

Повна схема БД приведена на рис. 2.5.

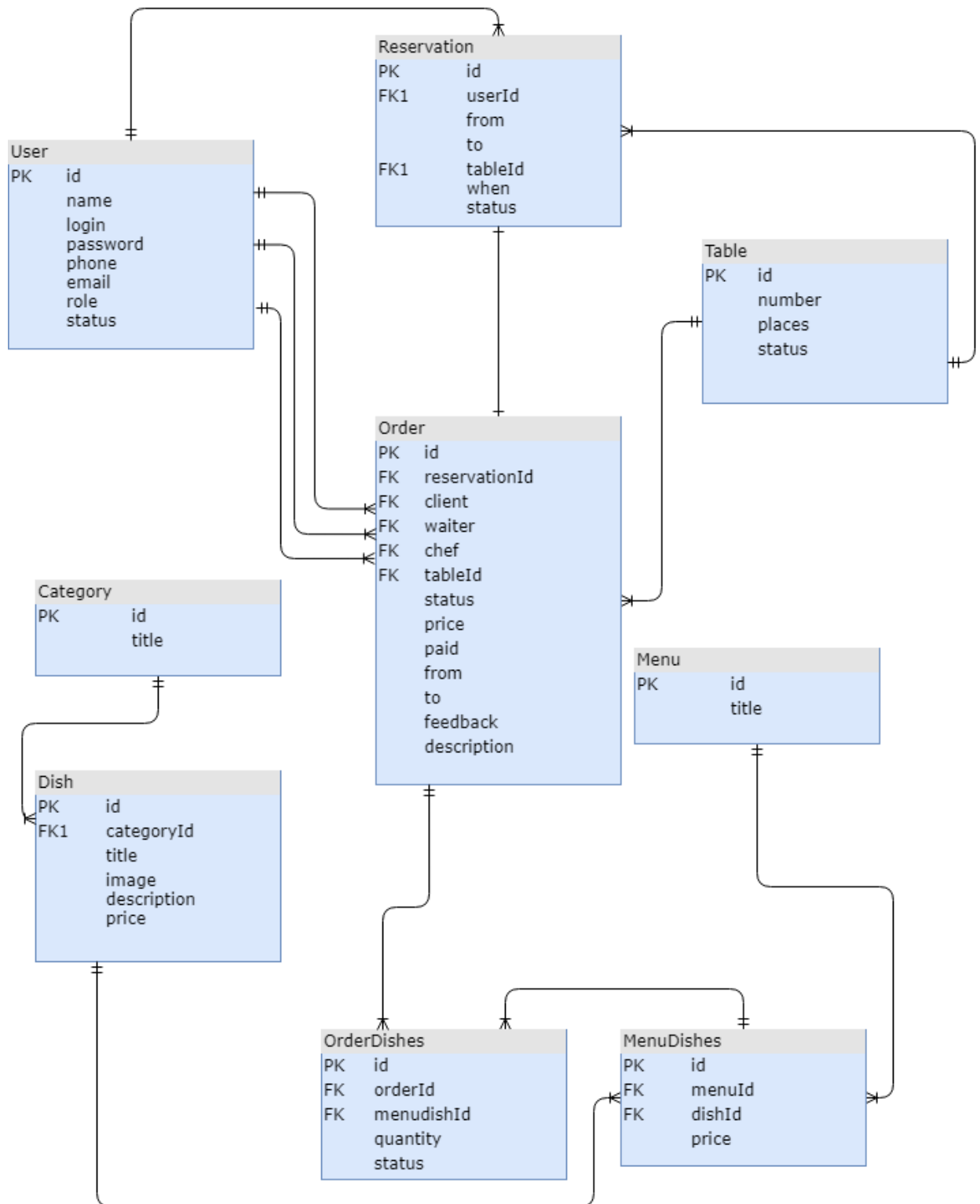


Рисунок 2.5 – Схема БД

Таблиця Reservation включає такі поля:

– id – номер бронювання;

- userId – користувач, який виконав бронювання;
- from – початок бронювання столику;
- to – кінець бронювання столику;
- when – дата і час формування бронювання;
- tableId – номер столу;
- status – стан (активний або неактивний).

Таблиця MenuDishes включає такі поля:

- id – номер запису;
- menuId – номер варіанту меню;
- dishId – номер страви;
- price – вартість страви в даному варіанті меню (може бути пропущена, тоді відповідає стандартній вартості).

2.4 Визначення архітектури програмного забезпечення

Архітектуру програмного забезпечення було визначено на основі шаблону Model-View-ViewModel (MVVM).

Особливостями шаблону MVVM (рис. 2.6) є наступні:

- розділяє бізнес-шар і презентаційний шар, подібно до того, як це роблять Model-View-Presenter та Model-View-Controller;
- поліпшує розділення проблеми (View, ViewModel та Model);
- надає кращий робочий процес щодо взаємодії дизайнера та розробника;
- підвищення простоти та тестування;
- підтримується надійна можливість прив'язки даних мови eXtensible Application Markup Language (XAML);
- не потрібно використовувати «code-behind»;
- надає можливість розробки застосунків для різних середовищ, що важливо для систем, в яких створюється платформа для застосунків різних операційних систем;

- потужне прив'язування даних, команди, валідація та багато іншого;
- дизайнер і розробник можуть працювати разом [14].

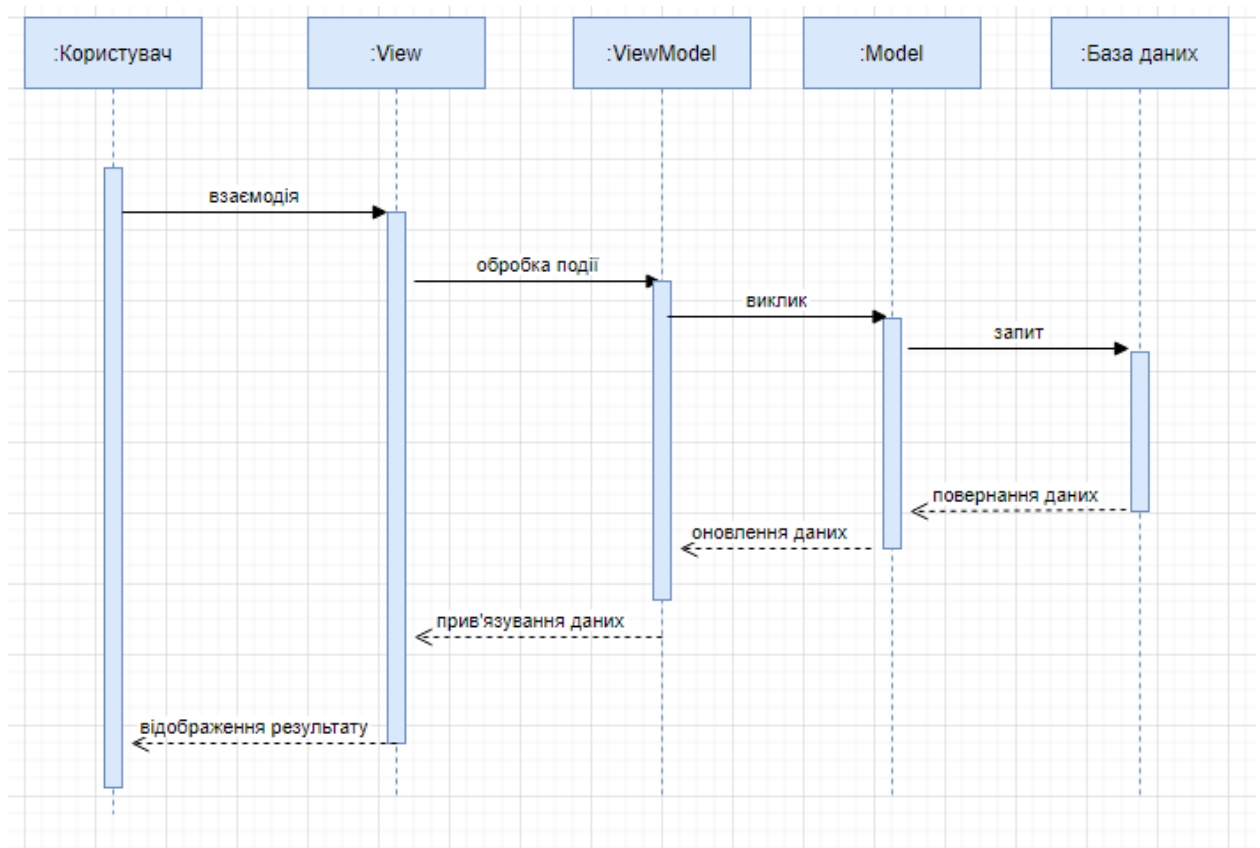


Рисунок 2.6 – Діаграма послідовностей MVVM

2.5 Висновки за розділом 2

Виконано вибір мови програмування для реалізації програмного забезпечення – C#, ґрунтуючись на результатах порівняння з мовою програмування C++.

Сформульовано функціональні вимоги до програмного забезпечення, виділено основні сценарії взаємодії клієнта, офіціанта, менеджера та кухаря з програмою, для кожного набору сценаріїв взаємодії відповідного типу користувачів побудовано діаграми прецедентів.

Забезпечено проектування БД на основі використання системи управління БД Microsoft SQL Server, що було акумульовано у вигляді схеми БД.

Визначено архітектуру програмного забезпечення на основі використання шаблону MVVM, використання якого зокрема важливо для можливостей подальшого розширення платформи іншими застосунками. Представлено відповідну діаграми послідовностей.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура розробленого програмного забезпечення

У відповідності з обраним шаблоном проєктування MVVM структурні одиниці програмного забезпечення представлені моделями, представленнями і моделями представлень (рис. 3.1).

Перелік створених моделей включає:

- UserModel – базова модель, що визначає обліковий запис користувача;
- ChefModel – модель кухаря, яка ґрунтується на моделі UserModel, має інтерфейс для роботи з БД через клас DBConnection, що забезпечує отримання всіх потрібних даних та їх змінювання для кухаря;
- WaiterModel – модель офіціанта, яка ґрунтується на моделі UserModel, має інтерфейс для роботи з БД через клас DBConnection, що забезпечує отримання всіх потрібних даних та їх змінювання для офіціанта;
- ClientModel – модель клієнта, яка ґрунтується на моделі UserModel, має інтерфейс для роботи з БД через клас DBConnection, що забезпечує отримання всіх потрібних даних та їх змінювання для клієнта;
- ManagerModel – модель кухаря, яка ґрунтується на моделі UserModel, має інтерфейс для роботи з БД через клас DBConnection, що забезпечує отримання всіх потрібних даних та їх змінювання для кухаря;
- DBConnection – клас, який забезпечує взаємодію з БД, зокрема визначає параметри підключення до БД, відповідно сам об'єкт підключення зберігається в класах ManagerModel, ClientModel, WaiterModel, ChefModel;
- OrderModel – модель замовлення;
- DishModel – модель страви;
- ReservationModel – модель бронювання столику;
- MenuModel – модель меню, що включає набір категорій, а за кожною категорією – страви;

- CategoryModel – модель категорії страв;
- TableModel – модель столику в закладі.

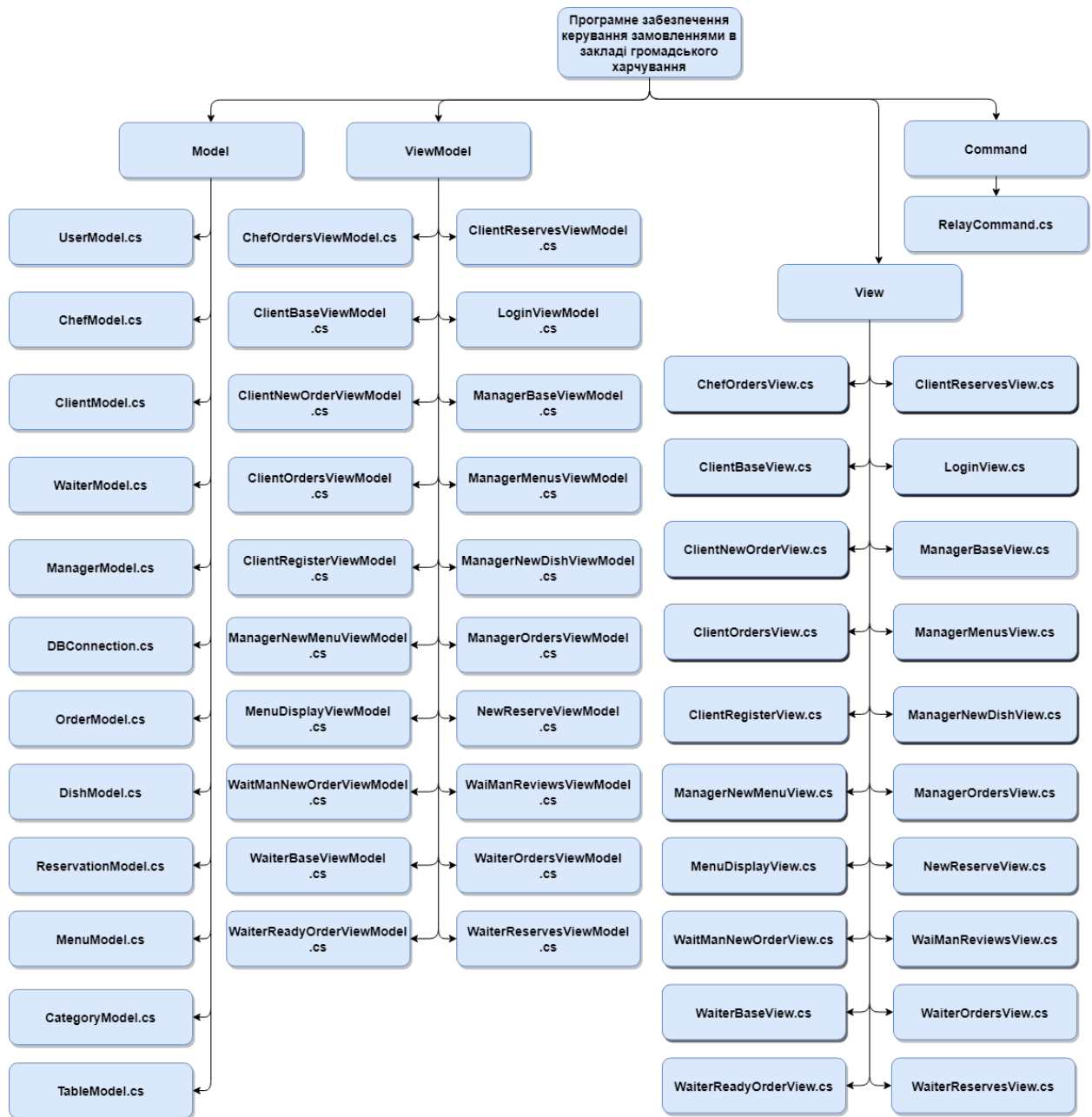


Рисунок 3.1 – Структура розробленого програмного забезпечення

Перелік створених класів моделей представлень включає:

- ChefOrdersViewModel – модель представлення роботи кухаря з замовленнями на приготування;

- ClientBaseViewModel – модель основного представлення клієнта, з якого він отримує доступ до всіх інших представлень, натискаючи на відповідні кнопки;
- ClientReservesViewModel – модель клієнтського представлення роботи з власними бронюваннями;
- LoginViewModel – модель представлення авторизації;
- ClientNewOrderViewModel – модель клієнтського представлення оформлення власного замовлення;
- ManagerBaseViewModel – модель основного представлення менеджера, з якого він отримує доступ до всіх інших представлень, натискаючи на відповідні кнопки;
- ClientOrdersViewModel – модель клієнтського представлення роботи з замовленнями;
- ManagerMenusViewModel – модель менеджерського представлення роботи з використовуваними в закладі громадського харчування меню;
- ClientRegisterViewModel – модель представлення реєстрації клієнтського облікового запису;
- ManagerNewDishViewModel – модель менеджерського представлення додавання нової страви в меню;
- ManagerNewMenuViewModel – модель менеджерського представлення створення нового меню;
- ManagerOrdersViewModel – модель менеджерського представлення роботи з замовленнями;
- MenuDisplayViewModel – модель представлення перегляду меню;
- NewReserveViewModel – модель представлення нового бронювання столику в закладі;
- WaitManNewOrderViewModel – модель представлення створення нового замовлення в закладі офіціантом або менеджером (дозволяє визначати особливості виконання замовлення);

– WaiManReviewsViewModel – модель представлення перегляду відгуків клієнтів про виконання замовлень в закладі громадського харчування офіціантами та менеджерами;

– WaiterBaseViewModel – модель основного представлення офіціанта, з якого він отримує доступ до всіх інших представлень, натискаючи на відповідні кнопки;

– WaiterOrdersViewModel – модель офіціантського представлення роботи з замовленнями;

– WaiterReadyOrderViewModel – модель офіціантського представлення роботи з готовими до видачі замовлень;

– WaiterReservesViewModel – модель офіціантського представлення роботи з бронюваннями столів на вказаний момент часу.

3.2 Послідовність виконання програми

Оскільки ключовою сутністю для даного програмного забезпечення є замовлення (рис 3.2), то розглянути суттїсну послїдовнїсть закладену в роботу програми можна пїд час розгляду послїдовностї станїв, у яких замовлення їснує. Таким чином розкривається вся послїдовнїсть взаємодїї рїзних типїв користувачїв з програмою.

Життєвий цикл замовлення розпочинається з його створення клїєнтом або офіціантом чи менеджером.

Якщо замовлення створюється клїєнтом, то спочатку воно отримує стан передзамовлення. Для того щоб передзамовлення стало остаточно замовленням, яке буде приготовлено, його має пїдтвердити спочатку офіціант або менеджер, в результатї чого замовлення стане виконуваним. До того моменту, як замовлення стане виконуваним, його можна скасувати.

Якщо замовлення було внесено в програму офіціантом, то воно отримує стан створеного. Пїсля цього воно може бути скасованим або одразу перейти до приготування.

Як тільки створене або виконуване замовлення підтверджує на виконання кухар, замовлення переходить до стану «готується».

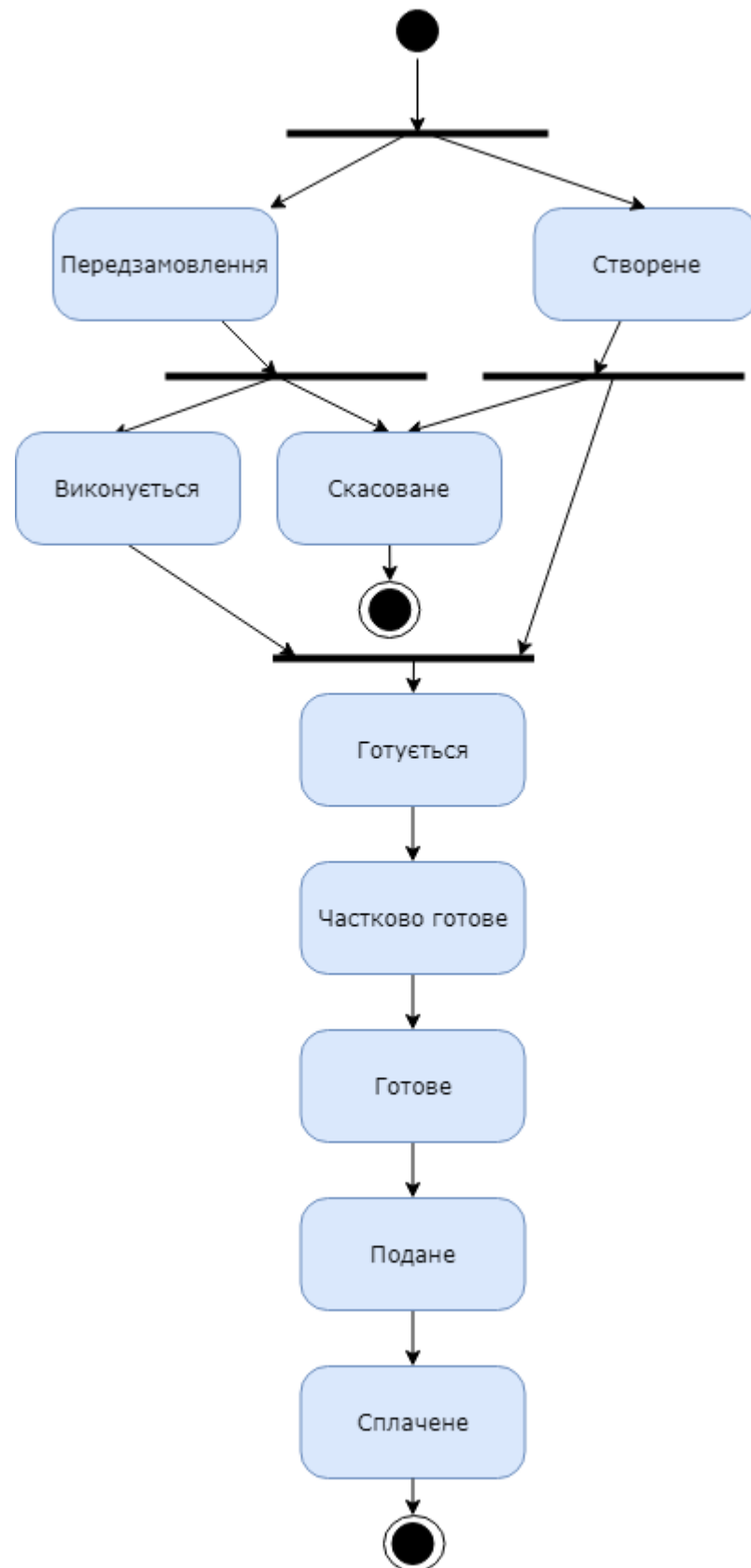


Рисунок 3.2 – Діаграма станів замовлення

Якщо частина страв із замовлення приготовлена, то замовлення стає в цей момент «частково готовим». Як тільки всі страви у замовленні приготовлені, замовлення стає готовим, тобто його може повністю подати на стіл клієнтам офіціант. Після цього замовлення вважається поданим.

Далі замовлення має бути сплачено клієнтами і після цього завершує свій життєвий цикл.

3.3 Опис розроблених структурних одиниць

Кожен клас моделей відповідає за своєю сутністю структурі таблиць БД, що було розглянуто в розділі 2. Для зручності було додано лише окрім поля: наприклад, для класу `OrderModel` окрім структури відповідної таблиці було додано поле `wtime`, яке визначає кількість часу, що пройшла з моменту створення замовлення. У кожного класу відповідного користувача міститься інтерфейс роботи з БД, а також набір методів, що виконують запити до БД. Оскільки даний час потрібно розраховувати тільки для створених або виконуваних замовлень, адже якщо створено передзамовлення, то його обробка без виконання дій офіціантом ще не настала, то було створено відповідні методи, які виконують запити до БД з метою витягання замовлень, що знаходяться у відповідних станах:

- `getProcessedOrders()` – отримання переліку невиконаних, тобто підтверджених або створених офіціантом, замовлень;
- `getFeedOrders()` – отримання переліку замовлень, готових до видачі;
- `getKitchenOrders()` – отримання переліку замовлень для приготування.

Відповідні переліки мають за натискання на відповідні кнопки відображатися менеджеру під час його роботи. Для цього реалізуються команди.

Для кожного представлення було реалізовано відповідну розмітку графічного інтерфейсу на основі використання мови XAML для визначення розташування даних, отриманих за допомогою кожної відповідної ViewModel.

Розглянемо основні особливості створення моделі представлення на основі класу ClientNewOrderViewModel. Відповідне представлення реалізує створення нового замовлення клієнтом, а модель представлення виконує підготовку даних і збереження отриманих в результаті даних. При цьому набір підготовлених даних включає меню, страви, які найчастіше замовляв цей клієнт до цього. Окрім того робота відбувається з переліком страв у замовленні, що відповідають стравам із поточного меню у заданій кількості, при чому страви з замовлення можуть вилучатися, а їх кількість змінюватися. Тому дана модель представлення потребує достатньо широку кількість засобів роботи з даними для їх подальшої адекватної презентації в представленні.

Модель представлення забезпечує виконання наступних команд:

- public RelayCommand CreateOrder – створення замовлення (фактично передзамовлення);
- public RelayCommand AddDish – додавання страви до замовлення;
- public RelayCommand AddPopularDish – додавання страви з переліку популярних до замовлення;
- public RelayCommand DeleteDish – вилучення страви з замовлення.

Створені властивості в даній моделі представлення:

- Order – поточне створюване замовлення;
- CurrentDish – поточна обрана страву з меню;
- DishQuantity – кількість страви;
- CurrentPopDish – поточна обрана популярна в клієнта страву;
- CurrentOrderDish – обрана страву зі створюваного замовлення (обраний пункт переліку);
- AddedDishQuantity – кількість обраної страви зі створюваного замовлення.

Інші атрибути моделі представлення:

– public ClientModel client – клієнт (авторизований користувач, який виконує створення замовлення);

– public ReservationModel reservation – резервування столику, на основі якого формується замовлення;

– public ObservableCollection<DishModel> PopularDishes – перелік страв, які поточний авторизований користувач обирає у своїх замовленнях найчастіше;

– public MenuModel menu – активне на даний момент меню.

Метод public ClientNewOrderViewModel (ClientModel user, ReservationModel reserv) реалізує конструктор, за допомогою якого виконується створення і ініціалізація об'єкту замовлення і визначення команд, які підтримуються в цьому представленні.

Метод public void DishPlus(object obj) реалізує додавання страви до замовлення.

Для цього спочатку виконується перегляд всіх страв, які вже було додано до замовлення, і зміненням значення змінної flag відзначається факт знаходження такої страви:

```
bool flag = false;
foreach(var (d, q) in order.Dishes)
{
    if (d == CurrentDish)
        flag = true;
}
```

Якщо страву не було знайдено, то вона додається до замовлення і оновлюється властивість, що зв'язує дані в моделі, моделі представлення та в представленні, а якщо було знайдено, то виводиться повідомлення:

```
if (!flag)
```

```

{
    order.Dishes.Add((CurrentDish, dishQuantity));
    OnPropertyChanged("Order");
}
else
{
    MessageBox.Show("Дане блюдо вже додано в замовлення");
}

```

Метод `public void PopDishPlus (object obj)` реалізує додавання страви до замовлення з переліку популярних у даного клієнта страв.

Метод `public void OrderDishMinus (object obj)` реалізує вилучення страви з переліку вже доданих до замовлення. Для цього виконується перевірка всіх доданих до замовлення страв і пошук тієї, що відповідає обраній для вилучення, якщо така страва була знайдена, то вона прибирається з переліку:

```

foreach(var v in order.Dishes)
{
    var (d, q) = v;
    if (d == CurrentOrderDish)
        order.Dishes.Remove(v);
}

```

Метод `public void ChangeOrderDish (object obj)` забезпечує зміну кількості заданої страви у замовленні.

Метод `public void NewOrder (object obj)` виконує створення замовлення.

Метод `public void NewPreOrder (object obj)` виконує створення передзамовлення, тобто час виконання замовлення буде пізнішим за поточний.

Вибір конкретної активної команди виконується на основі перевірки, чи було заповнено поле початку виконання замовлення. Якщо було, то реалізується передзамовлення, якщо ні – звичайне замовлення.

Метод `public bool OrderCheck (object obj)` виконує перевірку заданих даних замовлення перед його збереженням, зокрема перевіряється, що до замовлення були додані страви, а також те, чи було заповнено поле визначення початку обслуговування. Аналогічно працює метод `PreOrderCheck`, але по-іншому реагуючи на зазначення часу обслуговування.

3.4 Висновки за розділом 3

Розглянуто класи, які були створені в межах реалізації структури програмного забезпечення. Описано основні принципи реалізації цих класів.

Описано життєвий цикл програмного забезпечення на основі створення діаграми станів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Програмне забезпечення призначене для керування діяльністю закладу громадського харчування невеликого розміру, включаючи кав'ярню, ресторан або будь-який інший заклад громадського харчування з власною кухнею і залом обслуговування, щодо попереднього бронювання, оформлення, виконання та отримання зворотного зв'язку щодо вже виконаного замовлення від клієнта.

4.2 Умови виконання програми

Вимоги до технічних засобів, на яких має виконуватися клієнтська сторона:

- стаціонарний або мобільний (ноутбук) комп'ютер під керуванням операційної системи Windows, починаючи з версії 7;
- процесор не нижче Intel Celeron з тактовою частотою не менше 300 МГц;
- не менше 1 Гб оперативної пам'яті;
- 100 Мб вільного простору на жорсткому диску;
- монітор і відеокарта, які мають забезпечувати роботу з роздільною здатністю не менше 800х600.

4.3 Робота з програмою

Робота з програмою розпочинається з авторизації облікового запису користувача. Після цього обирається відповідна базова форма.

У випадку роботи клієнта (рис. 4.1) на його основній формі надається меню, що надає доступ до замовлень клієнта, бронювань столиків, інформації

про себе та виходу з облікового запису.

Вітаємо, шановний клієнте!

Замовлення

Бронювання

Про себе

Вийти

Всі **Невиконані**

Номер	Початок	Завершення	Вартість	Сплачено	Стан
3	07.05.2021 19:05	07.05.2021 19:57	537.0	537.0	Сплачено
7	08.05.2021 20:02	08.05.2021 21:43	1014.5	1015	Сплачено
8	09.05.2021 10:43	09.05.2021 11:05	250.0	250.0	Сплачено
9	09.05.2021 16:23		892.0	0.0	Подане

Страва	Кількість	Вартість
Цезар з креветками	1	80.0
Равіолі з вишнею	1	72.0
Борщ	1	53.0
Мус	1	45.0

Скасувати

Нове замовлення

Час залишити відгук...

Внести

Рисунок 4.1 – Керування власними замовленнями клієнтом

Під час перегляду власних замовлень клієнт може переглядати або всі замовлення, або невиконані, тобто ті, які оформлені, але ще не доведені до оплати, для цього призначені відповідні кнопки «Всі» та «Невиконані» вгорі

форми.

У верхній таблиці можна обрати потрібне замовлення, кожне з яких включає його номер, дату і час початку, завершення, вартість замовлення, стан та сплачену суму.

Після вибору одного з замовлень в таблиці нижче відображається склад замовлення, що включає назву страви, кількість замовлених страв, їх вартість.

Замовлення можна скасувати за допомогою відповідної кнопки, але операція призведе до результату тільки в тому випадку, якщо замовлення ще не оброблено офіціантом, тобто не передано кухарю (на формі офіціанта для замовлень, оформлених клієнтами, доступна кнопка підтвердження їх виконання).

Для створення нового замовлення потрібно з цієї форми натиснути відповідну кнопку і відкриється нове вікно створення нового замовлення (рис. 4.2).

Якщо замовлення вже сплачене, то клієнт може залишити відгук у відповідному багаторядковому полі. Для збереження відгука потрібно натиснути на кнопку «Внести».

На формі створення нового замовлення потрібно спочатку обрати категорію страви з меню з випадваючого переліку в верхньому лівому куті. Тоді в таблиці нижче відобразиться перелік страв за обраною категорією. Якщо натиснути на кнопку «Улюблені» правіше, то таблиця заповниться найпопулярнішими за вибором клієнта стравами.

Якщо обрати одну зі страв, то правіше відобразиться зображення страви та її опис.

Для того щоб додати обрану страву до замовлення, потрібно натиснути на кнопку «Додати».

Склад замовлення відображається в таблиці нижче. Склад включає назву страви та її кількість. Якщо обрати одну зі страв з замовлення, то правіше відобразиться її кількість у полі, яке можна редагувати.

Вітаємо, шановний клієнте!

Піца

Назва

Вартість

Піца Маргарита

95.0

Піца селямі

124.0

Піца Дияволо

131.0

Піца Кампаньола

138.0

Піца Іверія

162.0

Страва

Кількість

Піца Дияволо

1

Паста Карбонаро

2

Улюблені



Піца з помідорів, моцарелли та базилика з додавання оливкової олії за традиційним неаполитанським рецептом. Вихід: 450 г.

Додати

Кількість:

2

Вилучити

Сума: 251.0 грн.

☐ на час

12 травня 2021 р.

15

19:33

☐ за бронюванням

13.05.2021 21:00

Створити

Рисунок 4.2 – Створення нового замовлення клієнтом

Якщо потрібно не змінити кількість страв, а вилучити, то потрібно натиснути однойменну кнопку.

Сумарна вартість замовлення позначена нижче.

Для замовлення можна вказати варіант його створення на час в майбутньому. Для цього потрібно обрати прапорець «на час» та обрати з календаря потрібну дату, а в полі нижче вказати час початку замовлення. Тоді офіціант потім має підтвердити це замовлення для його відображення кухарю.

Замовлення можна створити на основі попереднього бронювання, для цього потрібно поставити прапорець «за бронюванням» і обрати з бронювань клієнта, які ще не виконано, з випадаючого переліку потрібний варіант.

Після того, як всі дані було внесено, потрібно натиснути на кнопку «Створити».

Зі сторінки бронювань доступна можливість створення нового бронювання (рис. 4.3), що доступна у вигляді окремого вікна. У цьому вікні доступна схема залу закладу громадського харчування.

Користувачу потрібно обрати дату початку бронювання з календаря та вказати в текстовому полі час початку бронювання. У відповідних полях правіше потрібно вказати дату і час завершення бронювання. Створити бронювання більше ніж на 2 години неможливо.

Далі потрібно обрати доступний стіл з випадаючого переліку. Тоді для нього нижче відображається кількість місць за столом.

Після цього потрібно натиснути на кнопку «Забронювати» для збереження бронювання або на кнопку «Забронювати і замовити» – тоді відбудеться збереження бронювання і одразу відкриється вікно створення замовлення, при цьому у відповідному випадаючому переліку вже буде обрано відповідне бронювання.

Офіціант у свою чергу під час роботи з програмою може керуватися меню, яке включає пункти меню, бронювань, замовлень (вносяться замовлення, робота з якими відбувається), історії (вносяться всі замовлення), відгуки клієнтів.

Вітаємо, шановний клієнте!

Початок бронювання

15 травня 2021 р.

21:00

Доступний стіл

Кількість місць: 2

Забронювати

Завершення бронювання

15 травня 2021 р.

22:00

2

Забронювати і замовити

Рисунок 4.3 – Бронювання столику клієнтом

При виборі пункту «Замовлення» (рис. 4.4) офіціанту доступні кнопки «Всі» – для відображення всіх неоплачених замовлень, «Готові» – для

відображення готових до видачі замовлень та «Невиконані» для відображення замовлень, які ще не приготовлені кухарем.



Рисунок 4.4 – Керування замовленнями офіціантом

За кожним замовленням офіціант може переглянути склад, скасувати його, створити нове замовлення, додати страву в замовлення або підтвердити

видачу замовлення при видачі його з кухні.

Офіціант може керувати замовленнями (рис. 4.5), при цьому йому відображається перелік зайнятих столів на вказаний проміжок часу.

Керуйте з задоволенням!

Меню 12 травня 2021 р. 15 19:00 Обрати

Бронювання

Замовлення

Історія

Відгуки

Вийти

Бронювання

Стіл	Початок	Завершення	Користувач
1	12.05.2021 19:00	12.05.2021 20:00	aleks
4	12.05.2021 19:00	12.05.2021 20:00	dmitriy
7	12.05.2021 19:00	12.05.2021 20:00	waiter12
11	12.05.2021 19:00	12.05.2021 20:00	admin

Забронювати

Скасувати

12.05.2021 19:00

12.05.2021 20:00

7

Рисунок 4.5 – Керування бронювання офіціантом

Для керування замовленнями офіціант обирає дату і час, а бронювання

вказуються на проміжок з заданого початку і тривалістю 30 хвилин після цього. Офіціант може відредагувати будь-яке бронювання, скасувати або створити нове.

Кухарю доступний тільки перелік замовлень, які було оформлено, час їх початку виконання настав, але їх ще не приготовлено (рис. 4.6).

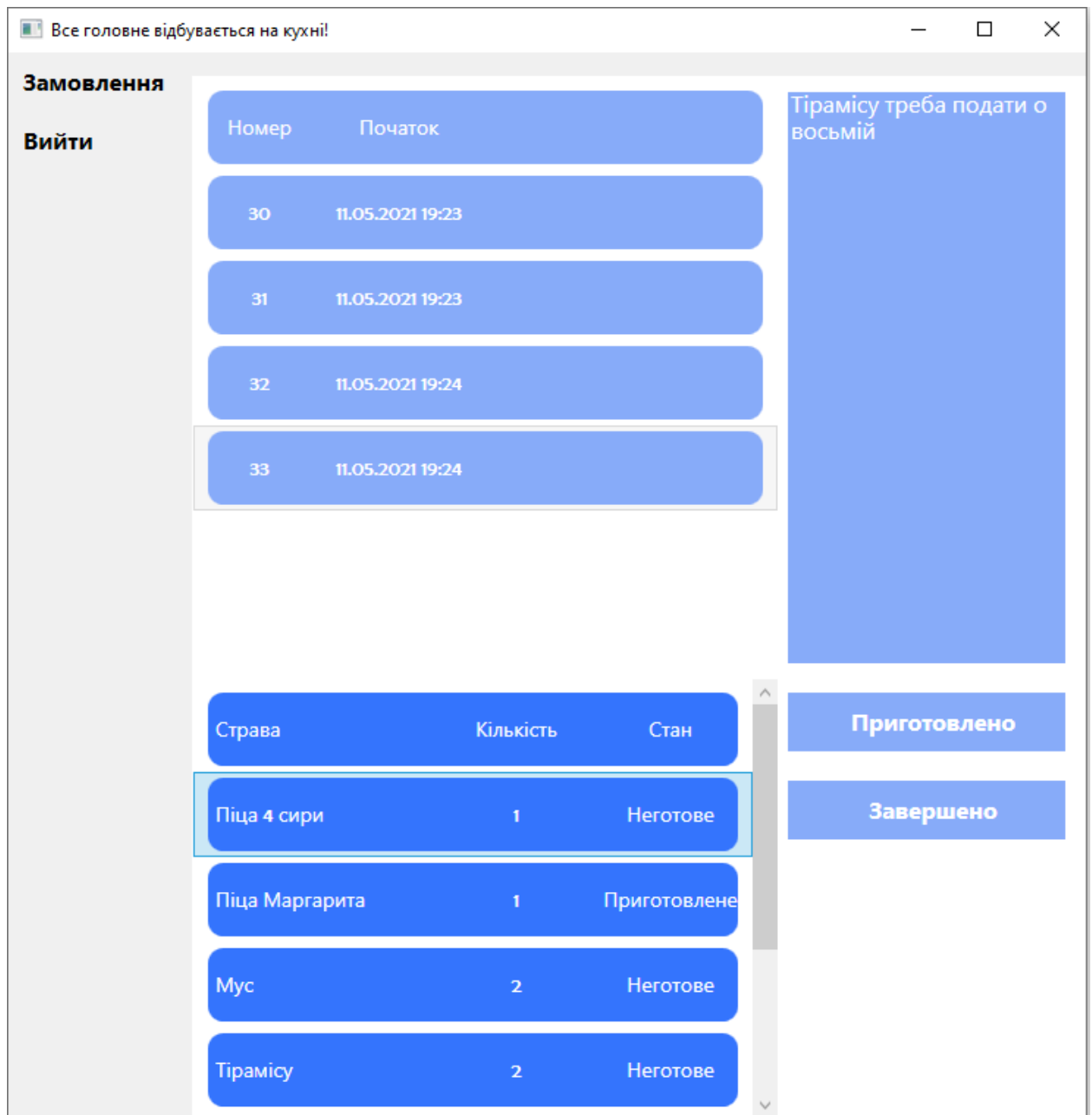


Рисунок 4.6 – Керування замовленнями кухарем

Кухар може підтвердити приготування страви з замовлення (кнопка «Приготовлено») або підтвердити приготування всього замовлення (кнопка «Завершено»).

4.4 Висновки за розділом 4

Режими роботи користувачів з програмою описано в даному розділі. Для кожного виділено основні сценарії взаємодії та доступні при цьому функції.

ВИСНОВКИ

Виконання дипломної кваліфікаційної роботи бакалавра присвячено проблемі керування замовленнями в закладі громадського харчування. У процесі аналізу проблеми розглянуто переваги створення системи керування замовленнями в закладі громадського харчування, базові складові і особливості такого програмного забезпечення, принципи організації успішного обслуговування і способи перенесення цих принципів на програмне забезпечення. Розглянуто програмні аналоги, зокрема Toast та TouchBistro, які потребують сплати суттєвих коштів за використання, а також специфічної апаратної платформи.

Обрано мову програмування C# для реалізації програмного забезпечення. Сформульовано функціональні вимоги до програмного забезпечення, виділено основні сценарії взаємодії клієнта, офіціанта, менеджера та кухаря з програмою, для кожного набору сценаріїв взаємодії відповідного типу користувачів побудовано діаграми прецедентів. Виконано проєктування БД на основі використання системи управління БД Microsoft SQL Server, що було акумульовано у вигляді схеми БД. Визначено архітектуру програмного забезпечення на основі використання шаблону MVVM, використання якого зокрема важливо для можливостей подальшого розширення платформи іншими застосунками.

Розроблене програмне забезпечення призначене для керування діяльністю закладу громадського харчування невеликого розміру, включаючи кав'ярню, ресторан або будь-який інший заклад громадського харчування з власною кухнею і залом обслуговування. Програма надає засоби роботи з замовленнями на всіх етапах їх життєвого циклу. Підтримується можливість зворотного зв'язку з клієнтами щодо обслуговування. Надається можливість формування замовлень на основі тих страв, які найчастіше раніше замовляв клієнт, що спрощує процес вибору його стандартного меню. Надається можливість визначення різних варіантів меню у різний час доби.

Всі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hyun, J. Complete Guide to Restaurant Service & Operations Management [Electronic resource] / J. Hyun. – Access mode : <https://zipordering.com/restaurant-service.html>.
2. An Online Food Court Ordering System Access mode [Electronic resource] / S. Reddy, K. G. K. Chaitanya, M. Abhinav, T. H. F. Khan // Journal of Information Technology & Software Engineering. – 2016. – Volume 6, Issue 4. – Access mode : <https://www.longdom.org/open-access/an-online-food-court-ordering-system-2165-7866-1000183.pdf>.
3. Restaurant Management System – All Components Explained [Electronic resource]. – Access mode : <https://medium.com/@LimeTrayTech/restaurant-management-system-all-components-explained-9cd0e2c446fe>.
4. Datta, S. A Complete Guide To Restaurant Management Systems [Electronic resource] / S. Datta. – Access mode : <https://limetray.com/blog/restaurant-management-system/>.
5. Toast | Restaurant Point of Sale & Management System [Electronic resource]. – Access mode : <https://pos.toasttab.com/>.
6. Sherman, M. Toast POS Review [Electronic resource] / M. Sherman. – Access mode : <https://www.merchantmaverick.com/reviews/toast-pos-review/>.
7. Toast POS Review [Electronic resource]. – Access mode : <https://www.fool.com/the-blueprint/toast-pos-review/>.
8. TouchBistro [Electronic resource] : iPad Restaurant POS System. – Access mode : <https://www.touchbistro.com/>.
9. Sherman, M. TouchBistro POS Review [Electronic resource] / M. Sherman. – Access mode : <https://www.merchantmaverick.com/reviews/touchbistro-review/>.
10. Stellman, A. Head First C# [Text] : A Learner's Guide to Real-World Programming with C# and .NET Core / Andrew Stellman , Jennifer Greene. – Sebastopol, CA : O'Reilly Media, 2021. – 800 p.

11. Price, M. J. C# 9 and .NET 5 – Modern Cross-Platform Development [Text] : Build intelligent apps, websites, and services with Blazor, ASP.NET Core, and Entity Framework Core using Visual Studio Code / M. J. Price. – Birmingham : Packt Publishing, 2020. – 822 p.
12. Bancila, M. Modern C++ Programming Cookbook [Text] / M. Bancila. – Birmingham : Packt Publishing, 2020. – 750 p.
13. Introducing Microsoft SQL Server 2019 [Text] : Reliability, scalability, and security both on premises and in the cloud / Kellyn Gorman, Allan Hirt, Dave Noderer, Mitchell Pearson et al. – Birmingham : Packt Publishing, 2020. – 488 p.
14. Kumar, V. MVVM (Model View ViewModel) Introduction [Electronic resource] : Part 1 / V. Kumar. – Access mode : <https://www.c-sharpcorner.com/UploadFile/0b73e1/mvvm-model-view-viewmodel-introduction-part-1/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення має бути направлене на надання зручного обслуговування в закладі громадського харчування, тому має бути забезпечене не тільки виконання замовлень, але і отримання зворотнього зв'язку від користувача, а також можливість участі в замовленнях користувачів різних типів у всьому життєвому циклі цього процесу.

A.1 Підстави для розробки

Розробка має бути виконана в межах завдання на дипломну кваліфікаційну роботу, яке було поставлено за темою «Програмне забезпечення керування замовленнями в закладі громадського харчування» у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення призначене для керування діяльністю закладу громадського харчування невеликого розміру, включаючи кав'ярню, ресторан або будь-який інший заклад громадського харчування з власною кухнею і залом обслуговування, щодо попереднього бронювання, оформлення, виконання та отримання зворотного зв'язку щодо вже виконаного замовлення від клієнта.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для реалізації визначеного призначення програма має надавати такі функціональні можливості:

- створення меню з категоріями та різними варіантами для різного часу дня і різних подій;
- створення страв з зображеннями;
- змінювання страв у меню;
- вибір активного варіанту меню;
- перегляд меню;
- бронювання столику;
- перегляд власних бронювань;
- перегляд заброньованих столів у закладі за заданий час;
- скасування бронювання столику;
- створення замовлення;
- створення замовлення на заданий час;
- скасування замовлення;
- створення замовлення на основі статистики попередніх замовлень;
- перегляд власних замовлень;
- перегляд стану невиконаних замовлень;
- перегляд замовлень для приготування;
- перегляд замовлень, готових до видачі;
- приготування страви;
- видача замовлення;
- внесення даних про сплату замовлення;
- внесення відгуків про виконання замовлення;
- перегляд відгуків клієнтів;
- визначення особливостей виконання замовлень;
- авторизація і реєстрація облікових записів.

A.3.2 Вимоги до надійності

За результатами введених користувачем даних має здійснюватися перевірка того, чи було введено коректні дані для кожного поля (зокрема для

кожного рядка має перевірятися, чи введене значення не є більшим за довжину вказану для відповідного поля в таблиці БД).

Окремо має перевірятися, чи вірно задано співвідношення між даними (наприклад, дата і час бронювання столику і оформлення попереднього замовлення для такого бронювання не можуть відповідати даті і часу, ранішим за поточні значення).

А.3.3 Умови експлуатації

Для експлуатації програми особливих вмінь від користувачів окрім вміння користуватися комп'ютером не висувається.

Для забезпечення роботи БД на сервері необхідні вміння системного адміністратора для підтримки роботи сервера системи керування БД Microsoft SQL Server.

А.3.4 Вимоги до складу та параметрів технічних засобів

Вимоги до технічних засобів, на яких має виконуватися клієнтська сторона:

- стаціонарний або мобільний (ноутбук) комп'ютер під керуванням операційної системи Windows, починаючи з версії 7;
- процесор не нижче Intel Celeron з тактовою частотою не менше 300 МГц;
- не менше 1 Гб оперативної пам'яті;
- 100 Мб вільного простору на жорсткому диску;
- монітор і відеокарта, які мають забезпечувати роботу з роздільною здатністю не менше 800x600.

А.4 Порядок контролю та приймання

Принципи контролю виконання роботи визначаються календарним планом, розташованим у завданні на дипломну кваліфікаційну роботу, де вказано назви етапів, строки їх виконання та результат.

ДОДАТОК Б
Текст програми

Б.1 Текст файла ClientNewOrderViewModel.cs

```

using MCE.Model;
using MCE.Command;

using System.ComponentModel;
using System.Windows;
using System.Runtime.CompilerServices;

namespace MCE.ViewModel
{
    class ClientNewOrderViewModel : INotifyPropertyChanged
    {
        public ClientModel client { get; set; }
        public ReservationModel reservation { get; set; }

        private OrderModel _order;
        public OrderModel order
        {
            get { return _order; }
            set
            {
                if (_order != value)
                {
                    _order = value;
                    OnPropertyChanged("Order");
                }
            }
        }

        public ObservableCollection<DishModel> PopularDishes
        {
            get { return client.getPopularDishes(); }
        }

        public MenuModel menu
        {
            get { return client.getCurrentMenu(); }
        }

        public RelayCommand CreateOrder { get; set; }
        public RelayCommand CreatePreOrder { get; set; }
        public RelayCommand AddDish { get; set; }
        public RelayCommand AddPopularDish { get; set; }
        public RelayCommand DeleteDish { get; set; }
    }
}

```

```

        public RelayCommand EditDish { get; set; }

        public ClientNewOrderViewModel (ClientModel user,
ReservationModel reserv)
        {
            CreateOrder = new RelayCommand(NewOrder,
OrderCheck);
            CreatePreOrder = new RelayCommand(NewPreOrder,
PreOrderCheck);
            AddDish = new RelayCommand(DishPlus);
            EditDish = new RelayCommand(ChangeOrderDish);
            AddPopularDish = new RelayCommand(PopDishPlus);
            DeleteDish = new RelayCommand(OrderDishMinus);
            order = new OrderModel ()
            { client = user,
              reservation = reserv,
              status = 0
            };
        }

        private DishModel _currentDish;

        public DishModel CurrentDish
        {
            get { return _currentDish; }
            set
            {
                _currentDish = value;
                RaisePropertyChanged("CurrentDish");
            }
        }

        private DishModel _dishQuantity;

        public int dishQuantity
        {
            get { return _dishQuantity; }
            set
            {
                _dishQuantity = value;
                RaisePropertyChanged("DishQuantity");
            }
        }

        public void DishPlus(object obj)
        {

```

```

        bool flag = false;
        foreach(var (d, q) in order.Dishes)
        {
            if (d == CurrentDish)
                flag = true;
        }
        if (!flag)
        {
            order.Dishes.Add((CurrentDish,
dishQuantity));
            OnPropertyChanged("Order");
        }
        else
        {
            MessageBox.Show("Дане блюдо вже додано в
замовлення");
        }
    }

    private DishModel _currentPopDish;

    public DishModel CurrentPopDish
    {
        get { return _currentPopDish; }
        set
        {
            _currentPopDish = value;
            RaisePropertyChanged("CurrentPopDish");
        }
    }

    public void PopDishPlus (object obj)
    {
        bool flag = false;
        foreach(var (d, q) in order.Dishes)
        {
            if (d == CurrentPopDish)
                flag = true;
        }
        if (!flag)
        {
            order.Dishes.Add((CurrentPopDish,
dishQuantity));
            OnPropertyChanged("Order");
        }
        else
    
```

```

        {
            MessageBox.Show("Дане блюдо вже додано в
замовлення");
        }
    }
    private DishModel _currentOrderDish;

    public DishModel CurrentOrderDish
    {
        get { return _currentOrderDish; }
        set
        {
            _currentOrderDish = value;
            RaisePropertyChanged("CurrentOrderDish");
        }
    }

    public void OrderDishMinus (object obj)
    {
        foreach(var v in order.Dishes)
        {
            var (d, q) = v;
            if (d == CurrentOrderDish)
                order.Dishes.Remove(v);
        }
    }

    private DishModel _addeddishQuantity;

    public int addeddishQuantity
    {
        get { return _addeddishQuantity; }
        set
        {
            _addeddishQuantity = value;
            RaisePropertyChanged("AddedDishQuantity");
        }
    }

    public void ChangeOrderDish (object obj)
    {
        foreach(var v in order.Dishes)
        {
            var (d, q) = v;
            if (d == CurrentOrderDish)
                order.Dishes.Remove(v);
        }
    }

```

```

        order.Dishes.Add((d, addeddishQuantity));
    }
}

public void NewOrder (object obj)
{
    if (!order.from.HasValue)
    {
        order.from = DateTime.Now;
        order.status = 1;
    }
    if (client.NewOrder(order))
    {
        MessageBox.Show("Вітаємо, замовлення збережено  
в системі");
        order = new OrderModel ()
        {
            client = user,
            reservation = reserv,
            status = 1
        };
    }
    else MessageBox.Show("На жаль, в даний момент не  
вдалося створити замовлення");
}

public void NewPreOrder (object obj)
{
    if (client.NewOrder(order))
    {
        MessageBox.Show("Вітаємо, замовлення збережено  
в системі");
        order = new OrderModel ()
        {
            client = user,
            reservation = reserv,
            status = 0
        };
    }
    else MessageBox.Show("На жаль, в даний момент не  
вдалося створити замовлення");
}

public bool OrderCheck (object obj)
{
    return !order.from.HasValue && order.AllCheck;
}

```

```

        public bool PreOrderCheck (object obj)
        {
            return order.from.HasValue && order.AllCheck;
        }

        public event PropertyChangedEventHandler pceh;

        public void OnPropertyChanged([CallerMemberName]string
cmn = "")
        {
            if (pceh != null)
                PropertyChanged(this, new
PropertyChangedEventArgs (cmn) );
        }
    }
}

```

Б.2 Текст файла ClientOrdersViewModel.cs

```

using MCE.Model;
using MCE.Command;
using MCE.View;

using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows;

namespace MCE.ViewModel
{
    class ClientOrdersViewModel : INotifyPropertyChanged
    {
        public Client client { get; set; }

        public RelayCommand CancelOrder { get; set; }
        public RelayCommand AddFeedback { get; set; }
        public RelayCommand AddOrder { get; set; }

        public ObservableCollection<OrderModel> OwnOrders
        {
            get { return client.getOwnOrders(); }
        }
    }
}

```

```

        public ObservableCollection<OrderModel>
OwnUndoneOrders
    {
        get { return client.getOwnUndoneOrders(); }
    }

    private OrderModel _currentOrder;

    public OrderModel CurrentOrder
    {
        get { return _currentOrder; }
        set
        {
            _currentOrder = value;
            RaisePropertyChanged("CurrentOrder");
        }
    }

    private OrderModel _currentUndoneOrder;

    public OrderModel CurrentUndoneOrder
    {
        get { return _currentUndoneOrder; }
        set
        {
            _currentUndoneOrder = value;
            RaisePropertyChanged("CurrentUndoneOrder");
        }
    }

    public ClientOrdersViewModel (Client user)
    {
        client = user;
        CancelOrder = new RelayCommand(OrderDelete,
isCancelable);
        AddFeedback = new RelayCommand(NewOrderFeedback,
isFeed);
        AddOrder = new RelayCommand(NewOrder);
    }

    public void NewOrder (object obj)
    {
        var nw = new Window();
        nw.Content = ClieNewOrderModelView;
        nw.Show();
    }

```

```

        public void OrderDelete (object obj)
        {
            if (CurrentOrder.status == 0 ||
CurrentOrder.status == 1)
                if (client.ChangeOrderStatus(CurrentOrder.id,
8))
                {
                    MessageBox.Show("Замовлення скасовано!");
                    CurrentOrder.status = 8;
                }
                else MessageBox.Show("Не вдалося скасувати
замовлення");
            else MessageBox.Show("Скасувати замовлення на
даному етапі неможливо");
        }

        public void NewOrderFeedback (object obj)
        {
            if (CurrentOrder == null && CurrentUndoneOrder !=
null)
                if
(client.AddOrderFeedback(CurrentUndoneOrder.feedback))
                    MessageBox.Show("Дякуємо за Ваш відгук!");
                else MessageBox.Show("На жаль, в даний момент
не вдалося внести відгук в систему");
            else if (CurrentOrder != null && CurrentUndoneOrder
== null)
                if
(client.AddOrderFeedback(CurrentOrder.feedback))
                    MessageBox.Show("Дякуємо за Ваш відгук!");
                else MessageBox.Show("На жаль, в даний момент
не вдалося внести відгук в систему");
                else MessageBox.Show("Має бути обрано одне з
замовлень!");
        }

        public bool isCancelable (object obj)
        {
            if (CurrentOrder.status == 0 ||
CurrentOrder.status == 1)
                return true;
            else return false;
        }

        public bool isFeed (object obj)
        {

```

```

        if ((CurrentOrder == null && CurrentUndoneOrder !=
null) || (CurrentOrder != null && CurrentUndoneOrder == null))
            return true;
        else return false;
    }

    public event PropertyChangedEventHandler pceh;

    public void OnPropertyChanged([CallerMemberName]string
cmn = "")
    {
        if (pceh != null)
            PropertyChanged(this, new
PropertyChangedEventArgs (cmn));
    }
}

```

Б.3 Текст файла ManagerOrdersViewModel.cs

```

using MCE.Model;
using MCE.Command;
using MCE.View;

using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace MCE.ViewModel
{
    class ManagerOrdersViewModel : INotifyPropertyChanged
    {
        public Manager manager { get; set; }

        public RelayCommand CancelOrder { get; set; }

        public ObservableCollection<OrderModel> Orders
        {
            get
            { return manager.getAllOrders(); }
        }

        public ObservableCollection<OrderModel>
ProcessedOrders

```

```

        {
            get
            {
                ObservableCollection<OrderModel> oc =
manager.getProcessedOrders();
                ObservableCollection<OrderModel> oc2 = new
ObservableCollection<OrderModel>();
                foreach (OrderModel o in oc)
                {
                    DateTime now = DateTime.Now;
                    TimeSpan ts = now - o.from;
                    o.wtime = ts.Hours.ToString() + " " +
ts.Minutes.ToString();
                    oc2.Add(o);
                }
                return oc2;
            }
        }

        public ObservableCollection<OrderModel> FeedOrders
        {
            ObservableCollection<OrderModel> oc =
manager.getFeedOrders();
            ObservableCollection<OrderModel> oc2 = new
ObservableCollection<OrderModel>();
            foreach (OrderModel o in oc)
            {
                DateTime now = DateTime.Now;
                TimeSpan ts = now - o.from;
                o.wtime = ts.Hours.ToString() + " " +
ts.Minutes.ToString();
                oc2.Add(o);
            }
            return oc2;
        }

        public ObservableCollection<OrderModel> KitchenOrders
        {
            ObservableCollection<OrderModel> oc =
manager.getKitchenOrders();
            ObservableCollection<OrderModel> oc2 = new
ObservableCollection<OrderModel>();
            foreach (OrderModel o in oc)
            {
                DateTime now = DateTime.Now;

```

```

        TimeSpan ts = now - o.from;
        o.wtime = ts.Hours.ToString() + " " +
ts.Minutes.ToString();

        oc2.Add(o);
    }
    return oc2;

}

private OrderModel _currentOrder;
public OrderModel CurrentOrder
{
    get { return _currentOrder; }
    set
    {
        _currentOrder = value;
        RaisePropertyChanged("CurrentOrder");
    }
}

private OrderModel _currentProcessedOrder;

public OrderModel CurrentProcessedOrder
{
    get { return _currentProcessedOrder; }
    set
    {
        _currentProcessedOrder = value;
        RaisePropertyChanged("CurrentProcessedOrder");
    }
}

private OrderModel _currentFeedOrder;

public OrderModel CurrentFeedOrder
{
    get { return _currentFeedOrder; }
    set
    {
        _currentFeedOrder = value;
        RaisePropertyChanged("CurrentFeedOrder");
    }
}

private OrderModel _currentKitchenOrder;

```

```

public OrderModel CurrentKitchenOrder
{
    get { return _currentKitchenOrder; }
    set
    {
        _currentKitchenOrder = value;
        RaisePropertyChanged("CurrentKitchenOrder");
    }
}

public ManagerOrdersViewModel (Manager user)
{
    manager = user;
    CancelOrder = new RelayCommand(OrderDelete,
isCancelable);
}

public void OrderDelete (object obj)
{
    if (CurrentOrder.status == 0 ||
CurrentOrder.status == 1)
        if
(manager.ChangeOrderStatus(CurrentOrder.id, 8))
        {
            MessageBox.Show("Замовлення скасовано!");
            CurrentOrder.status = 8;
        }
        else MessageBox.Show("Не вдалося скасувати
замовлення");
        else MessageBox.Show("Скасувати замовлення на
даному етапі неможливо");
}

public bool isCancelable (object obj)
{
    if (CurrentOrder.status == 0 ||
CurrentOrder.status == 1)
        return true;
    else return false;
}

public event PropertyChangedEventHandler pceh;

public void OnPropertyChanged([CallerMemberName]string
cmn = "")
{

```

```

        if (pceh != null)
            PropertyChanged(this,
PropertyChangeEventArgs (cmn) );
    }

}

```

Б.4 Текст файла WaiterReservesViewModel.cs

```

using MCE.Model;
using MCE.Command;

using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows;

namespace MCE.ViewModel
{
    class WaiterReservesViewModel : INotifyPropertyChanged
    {
        public Waiter waiter {get; set;}

        private DateTime _moment;

        public DateTime moment
        {
            get { return _moment; }
            set
            {
                _moment = value;
                RaisePropertyChanged("ReservationTime");
            }
        }

        public ObservableCollection<OrderModel> Reservations
        {
            get
            { return manager.getAllReservations(moment); }
        }

        public RelayCommand CancelReservation { get; set; }

        public WaiterReservesViewModel (Waiter w)
        {

```

```

        waiter = w;
        moment = DateTime.Now;
        CancelReservation = new
RelayCommand(ReservationDelete, isCancelable);
    }

    private ReservationModel _currentReserve;

    public ReservationModel CurrentReserve
    {
        get { return _currentReserve; }
        set
        {
            _currentReserve = value;
            RaisePropertyChanged("CurrentReservation");
        }
    }

    public void ReservationDelete (object obj)
    {
        if
(manager.ChangeReservationStatus(CurrentReserve.id, 2))
        {
            MessageBox.Show("Бронювання скасовано!");
            CurrentOrder.status = 2;
        }
        else    MessageBox.Show("Не вдалося скасувати
бронювання");
    }

    public bool isCancelable (object obj)
    {
        if (CurrentReserve.status == 0 ||
CurrentReserve.status == 1)
            return true;
        else return false;
    }

    public event PropertyChangedEventHandler pceh;
    public void OnPropertyChanged([CallerMemberName]string
cmn = "")
    {
        if (pceh != null)
            PropertyChanged(this, new
PropertyChangedEventArgs (cmn));
    }
}

```

ДОДАТОК В
Слайди презентації

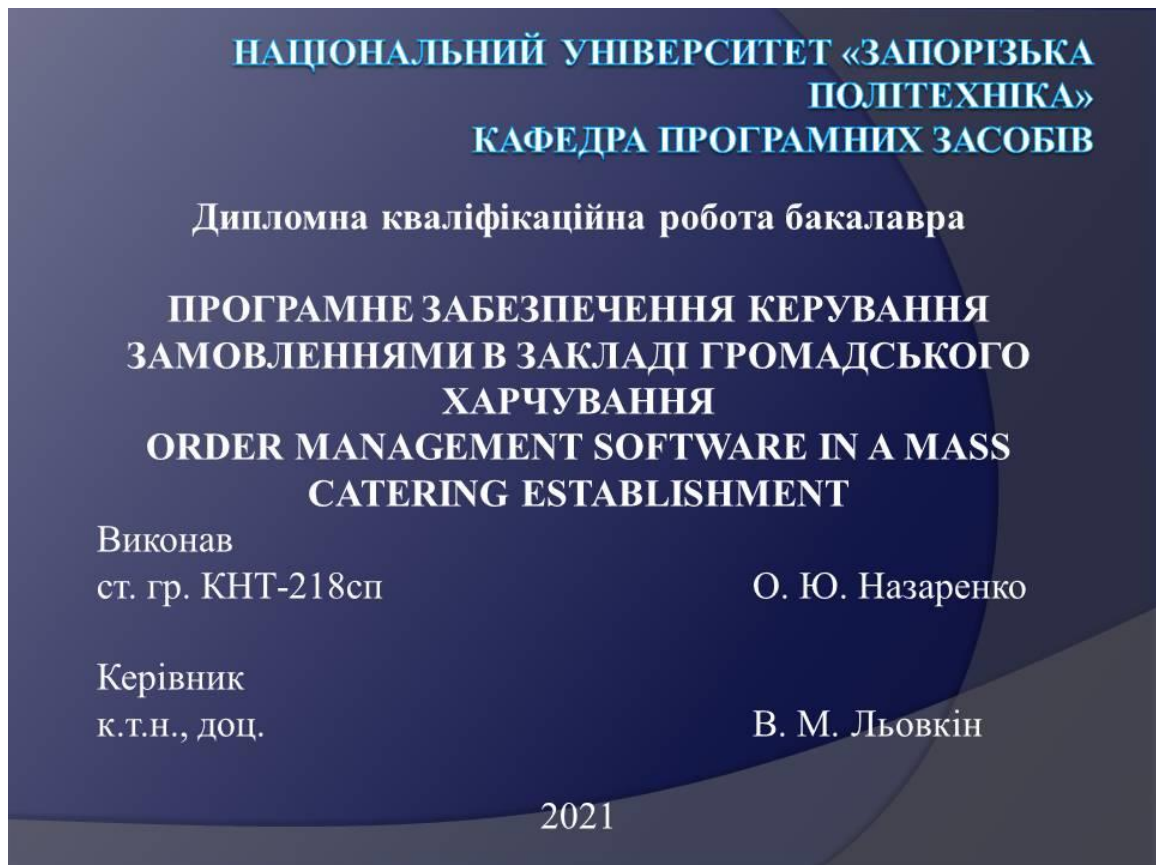


Рисунок В.1 – Слайд № 1

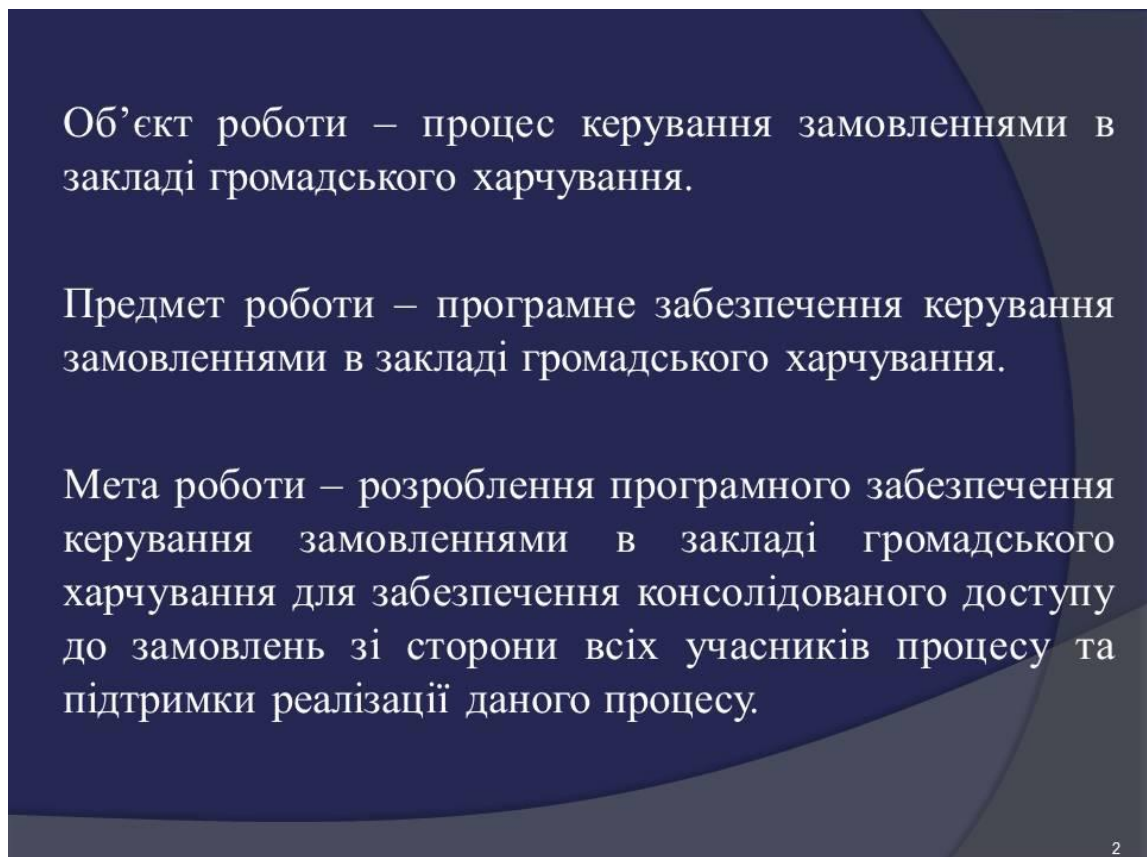


Рисунок В.2 – Слайд № 2

Завдання

- визначення вимог до програмного забезпечення керування замовленнями закладу громадського харчування;
- вибір мови і засобів проєктування і розробки;
- проєктування програмного забезпечення;
- реалізація і налагодження програмного забезпечення.

3

Рисунок В.3 – Слайд № 3

Вибір мови програмування

Показник	C#	C++
Об'єктно-орієнтовані засоби розробки	Так	Так
Простота створення коду	Висока	Нижча
Робота з базою даних	Інтерфейси роботи з різними системами керування базами даних, наявні вбудовані засоби роботи з MS SQL Server	Інтерфейси роботи з різними системами керування базами даних
Робота над графічним користувацьким інтерфейсом	Проста за допомогою середовища Microsoft Visual Studio	Має певні складнощі через Microsoft Visual Studio, потребує більше часу через фреймворк Qt

4

Рисунок В.4 – Слайд № 4

Діаграми прецедентів клієнта та кухаря

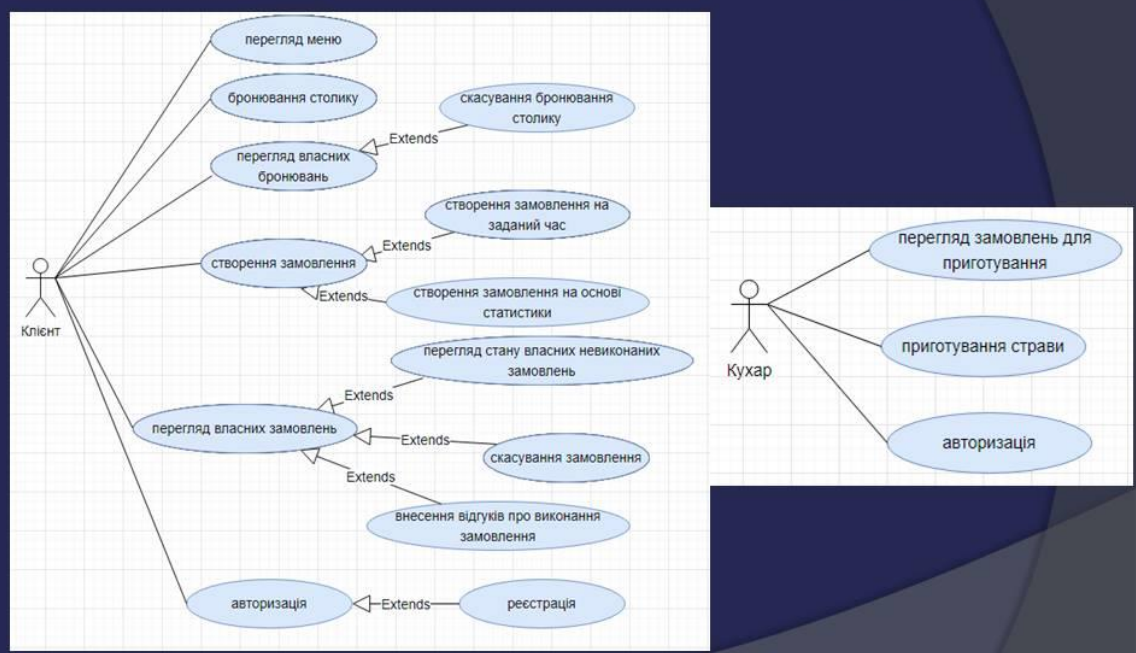


Рисунок В.5 – Слайд № 5

Діаграми прецедентів офіціанта та менеджера

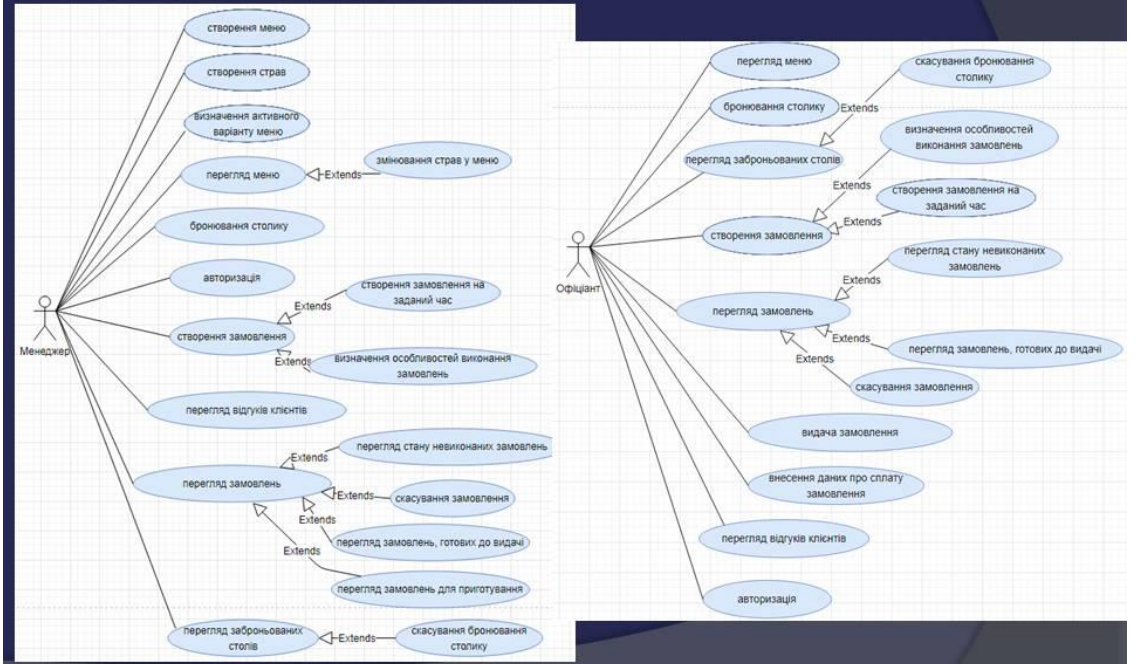


Рисунок В.6 – Слайд № 6

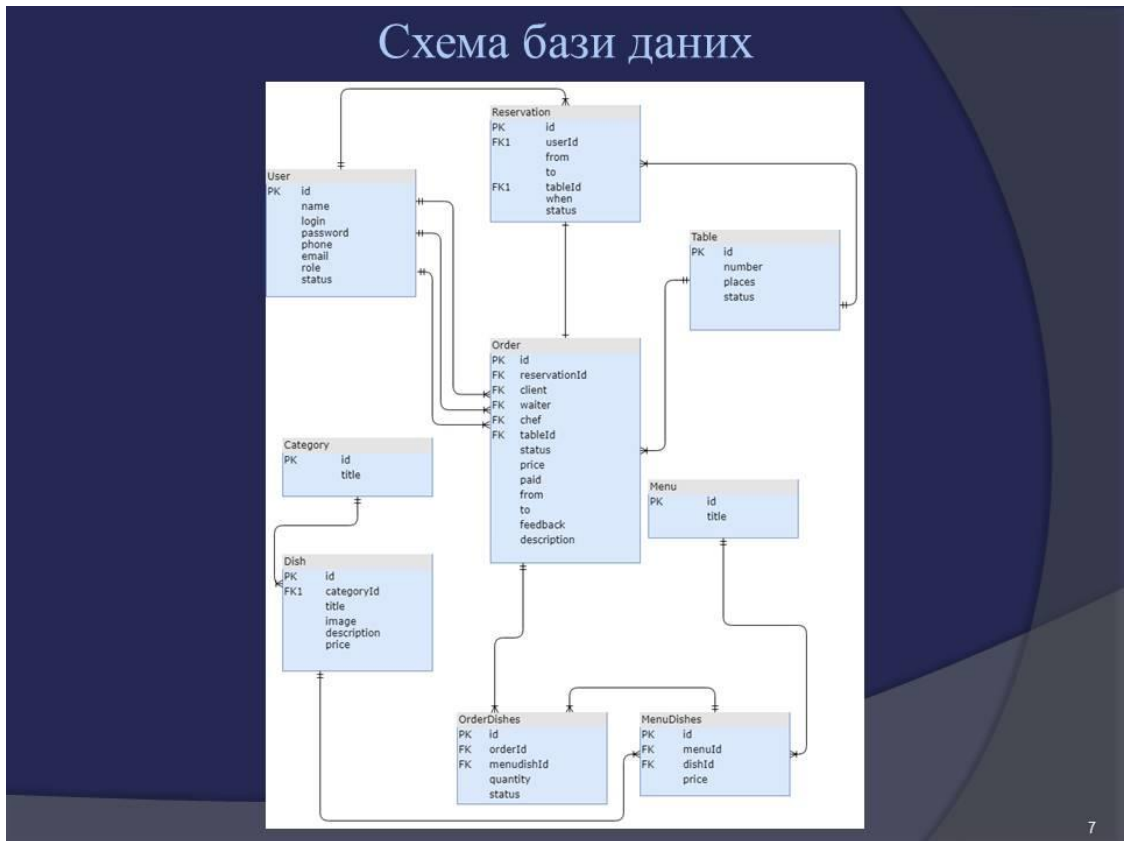


Рисунок В.7 – Слайд № 7

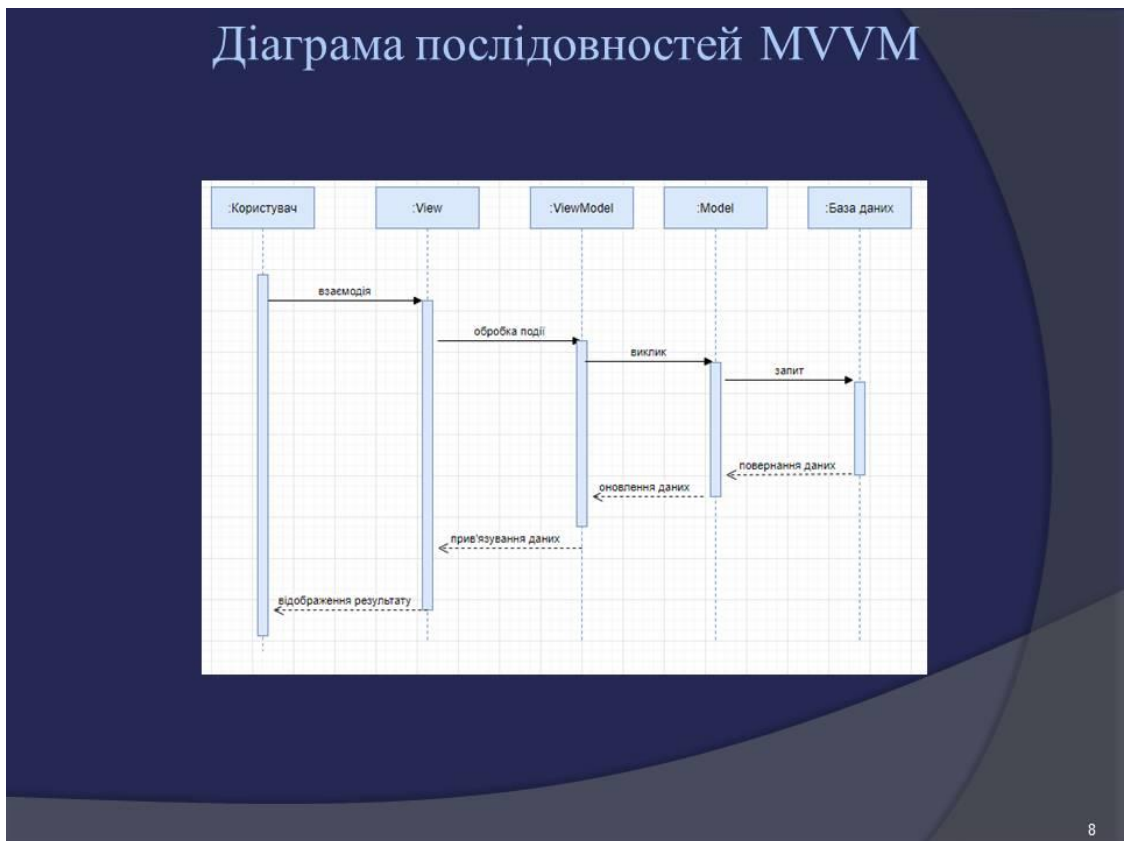
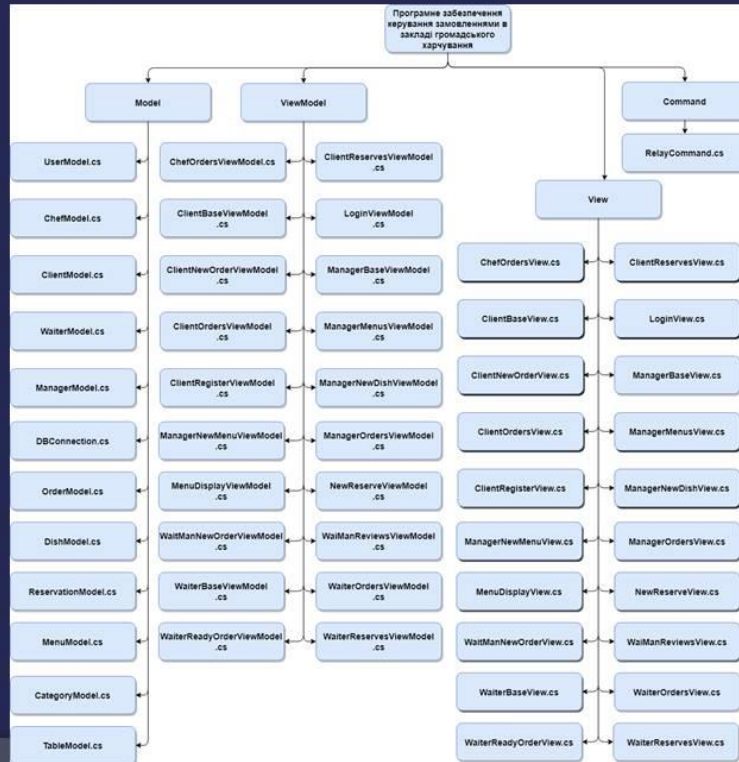


Рисунок В.8 – Слайд № 8

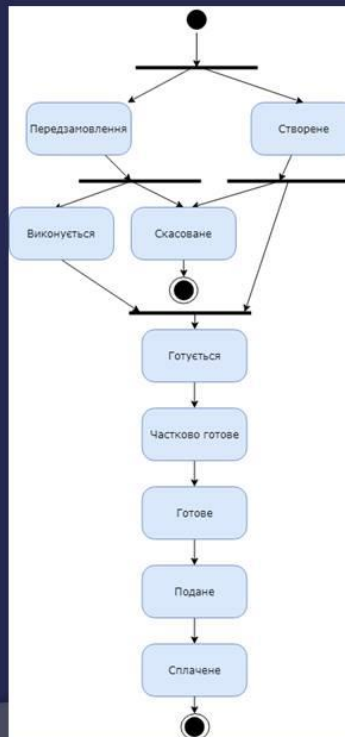
Структура розробленого програмного забезпечення



9

Рисунок В.9 – Слайд № 9

Діаграма станів замовлення



10

Рисунок В.10 – Слайд № 10

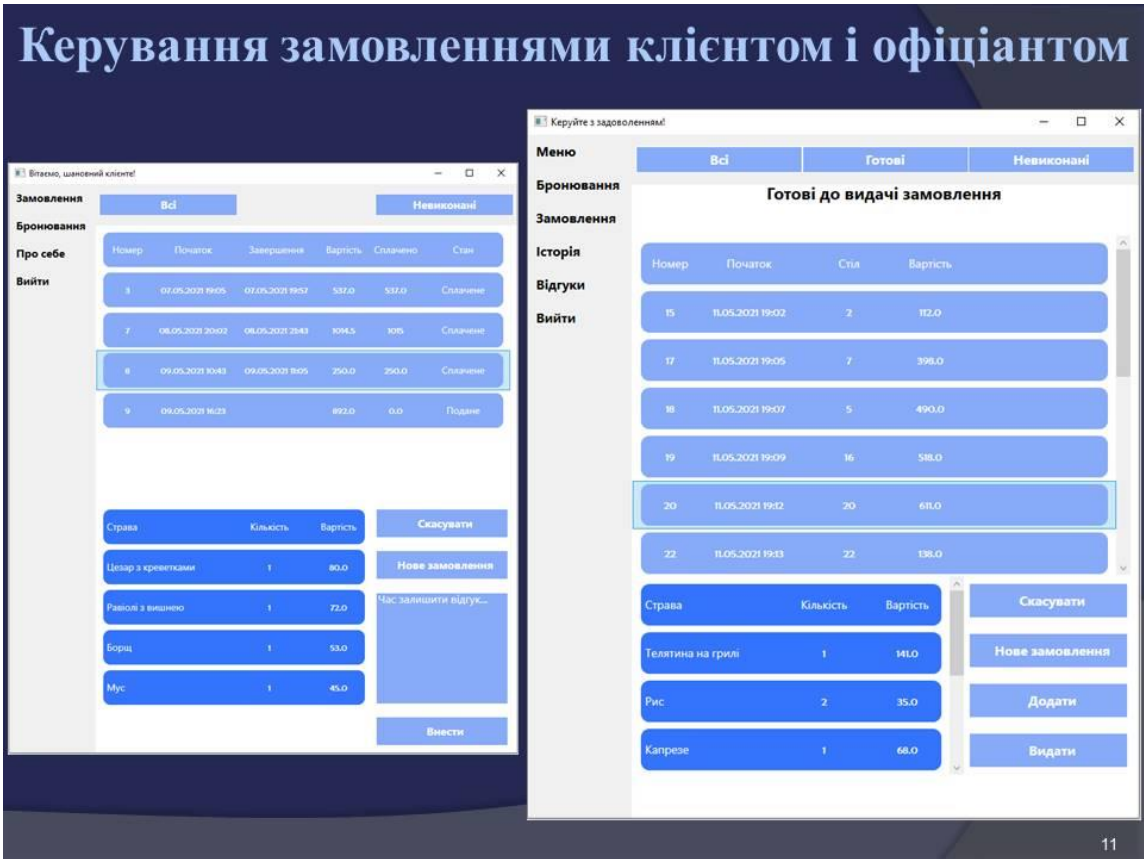


Рисунок В.11 – Слайд № 11

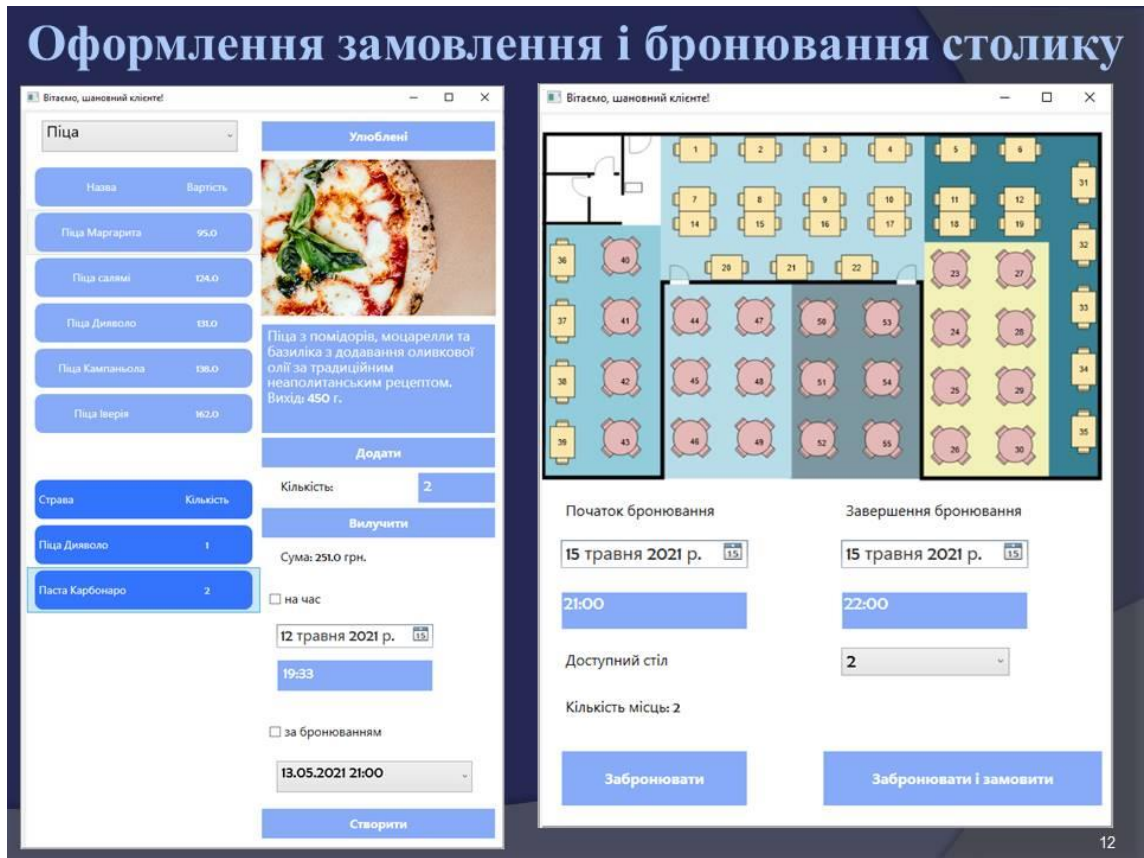


Рисунок В.12 – Слайд № 12

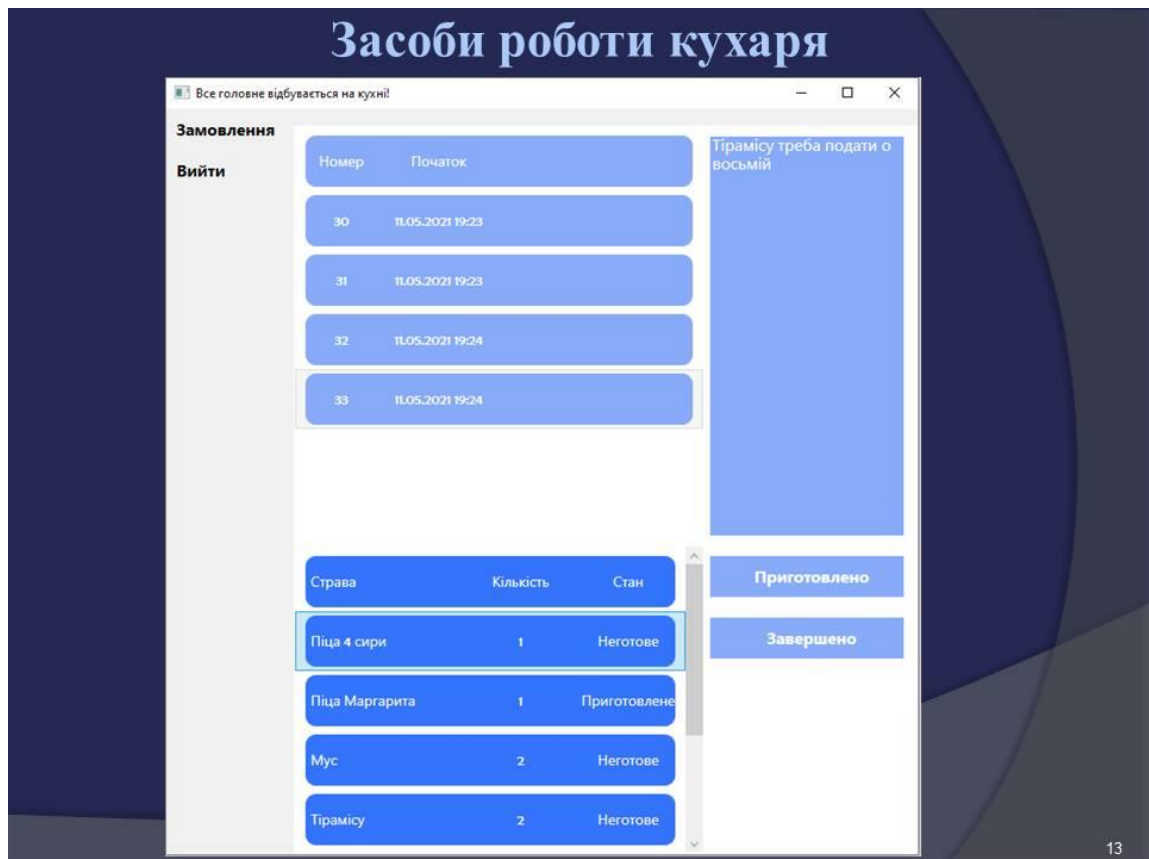


Рисунок В.13 – Слайд № 13

Висновки

Виконання дипломної кваліфікаційної роботи бакалавра присвячено проблемі керування замовленнями в закладі громадського харчування.

Обрано мову програмування C# для реалізації програмного забезпечення. Сформульовано функціональні вимоги до програмного забезпечення, виділено основні сценарії взаємодії клієнта, офіціанта, менеджера та кухаря з програмою, для кожного набору сценаріїв взаємодії відповідного типу користувачів побудовано діаграми прецедентів. Виконано проєктування БД на основі використання системи управління БД Microsoft SQL Server, що було акумульовано у вигляді схеми БД. Визначено архітектуру програмного забезпечення на основі використання шаблону MVVM.

Розроблене програмне забезпечення призначене для керування діяльністю закладу громадського харчування невеликого розміру, включаючи кав'ярню, ресторан або будь-який інший заклад громадського харчування з власною кухнею і залом обслуговування. Програма надає засоби роботи з замовленнями на всіх етапах їх життєвого циклу. Підтримується можливість зворотного зв'язку з клієнтами щодо обслуговування. Надається можливість формування замовлень на основі тих страв, які найчастіше раніше замовляв клієнт, що спрощує процес вибору його стандартного меню. Надається можливість визначення різних варіантів меню у різний час доби.

Всі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.

Рисунок В.14 – Слайд № 14