

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ ” _____ 2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для прийому та обробки замовлень в
ресторанному бізнесі

Виконав студент IV курсу, групи ІТ-82
(шифр групи)
Селюк Александер
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник доцент, к.т.н., доц., Новінський В.П.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Консультант
з графічної
документації доцент, к.т.н., доц., Ліщук К.І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент доцент каф ІСТ ФІОТ, к.т.н Попенко В.Д.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ –2022

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
комп'ютеризованих систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ” _____ 2022 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Селюку Александеру

(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі

керівник проєкту Новінський Валерій Петрович, к.т.н., доцент

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

затверджені наказом по університету від «10» червня 2022 р. №1033-с

2. Термін подання студентом проєкту « 19 » червня 2022 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.

2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.

3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.

4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.

5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використання _____

2) Схема структурна діяльності _____

3) Схема бази даних _____

4) Схема бізнес-процесів програмного забезпечення _____

5) Схема структурна класів програмного забезпечення _____

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2022 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	28.04.2022	
2	Аналіз існуючих методів розв'язання задачі	30.04.2022	
3	Постановка та формалізація задачі	05.05.2022	
4	Розробка інформаційного забезпечення	08.05.2022	
5	Алгоритмізація задачі	10.05.2022	
6	Обґрунтування вибору використаних технічних засобів	11.05.2022	
7	Розробка програмного забезпечення	15.05.2022	
8	Налагодження програми	20.05.2022	
9	Виконання графічних документів	21.05.2022	
10	Оформлення пояснювальної записки	22.05.2022	
11	Подання ДП на попередній захист	06.06.2022	
12	Подання ДП рецензенту	12.06.2022	
13	Подання ДП на основний захист	19.06.2022	

Студент

_____ (підпис)

Александр СЕЛЮК

_____ (ініціали, прізвище)

Керівник

_____ (підпис)

Валерій НОВІНСЬКИЙ

_____ (ініціали, прізвище)

Анотація

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 27 таблиць, 5 рисунків та 10 джерел – загалом 76 сторінки.

Дипломний проект присвячений створенню програмного продукту для прийому та обробки замовлень в ресторанному бізнесі.

Метою проекту є пришвидшення та оптимізація проходження життєвого циклу замовлень в ресторанному бізнесі.

У першому розділі наведені загальні положення, опис та аналіз предметної області, головна мета, цілі, функціональні і нефункціональні вимоги та призначення.

У другому розділі описане моделювання, розроблена архітектура та компоненти. Також наведений опис модулів, функцій та класів.

У третьому розділі описано процес тестування, наведені приклади тестів та проаналізована якість програмного забезпечення.

У четвертому розділі неведений опис та розгортання програми.

Результати роботи над проектом є програмне забезпечення, що допомагає в управлінні життєвим циклом замовленнями в ресторанному бізнесі.

КЛЮЧОВІ СЛОВА: КЕРУВАННЯ ПРОЦЕСАМИ, УПРАВЛІННЯ, ЗАМОВЛЕННЯ, РЕСТОРАННИЙ БІЗНЕС, КОМП'ЮТЕРНИЙ ДОДАТОК, JAVA, SPRING, JAVA FX, БАЗА ДАНИХ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 27 tables, 5 figures and 10 sources - a total of 76 pages.

The diploma project is devoted to the development of software for managing customer order processes at the restaurant business.

The purpose of this work is to improve and increase the efficiency of processing orders at a restaurant.

The first section provides general provisions, description and analysis of the subject area, the main purpose, objectives, functional and non-functional requirements and purposes.

The second section describes modeling, developed architecture and components. As well as, a description of modules, methods and classes.

The third section describes the testing process, gives examples of tests and analyzes the quality of the software.

The fourth section describes the deployment and accompaniment of the program.

The results of the project are software that helps manage the life cycle of customer orders in a restaurant.

KEYWORDS: PROCESS MANAGEMENT, ADMINISTRATION, ORDERS, RESTAURANT BUSINESS, COMPUTER SOFTWARE, JAVA, SPRING, JAVAFX, DATABASE.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2022 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРИЙОМУ ТА ОБРОБКИ
ЗАМОЛВЕНЬ В РЕСТОРАННОМУ БІЗНЕСІ**

Технічне завдання

КПІ.IT-8223.045430.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:

_____ Валерій НОВІНСЬКИЙ

Виконавець:

_____ Селюк Александер

Київ – 2022

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	6
4.3 Умови експлуатації	6
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.....	7
4.7 Вимоги до транспортування та зберігання	7
4.8 Спеціальні вимоги	7
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	9
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ	11
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	12

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: **Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі.**

Галузь застосування: управління життєвим циклом замовлення в ресторанному бізнесі

Наведене технічне завдання поширюється на розробку програмного забезпечення для управління життєвим циклом замовлень в ресторанах, котра використовується для прийому, створення, обробки, розрахування замовлень та призначена для ресторанного бізнесу різних категорій.

					КПІ.ІТ-8223.045430.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програми для прийому та обробки замовлень є завдання на дипломне проектування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

					КПІ.ІТ-8223.045430.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для пришвидшення та полегшення роботи з замовленнями, персоналом, меню.

Метою розробки є оптимізація прийому, обробки, змінення замовлень при обслуговуванні клієнтів у ресторанному бізнесі, через автоматизацію процесу за допомогою зрозумілого інтерфейсу та обробки даних програмним забезпеченням, а також збереження у базі даних.

					КПІ.ІТ-8223.045430.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

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:

- прийом замовлень;
- змінення замовлення;
- додавання клієнтів до замовлень;
- додавання страв до клієнтів в замовленні;
- розрахунок замовлення;
- реєстрація клієнтів;
- редагування клієнтів;
- видалення клієнтів.

4.1.1.2 Для адміністратора системи:

- створення меню;
- створення страв та додавання їх до меню;
- змінення даних меню;
- видалення меню;
- редагування страв;
- видалення страв з меню;
- реєстрація персоналу;
- редагування персоналу;
- видалення персоналу.

4.1.2 Розробку виконати на платформі Java версії 11

4.1.3 Додаткові вимоги:

- в якості бази даних використати MySQL;
- для інтерфейсу використати технологію JavaFX;

					КПІ.ІТ-8223.045430.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

4.2 Вимоги до надійності:

- передбачити перевірку на правильність типу введених даних;
- передбачити захист від некоректних дій користувача;
- забезпечити цілісність інформації в базі даних;
- передбачити перевірку повноти введених даних.

4.3 Умови експлуатації

- обслуговування потрібно у разі налаштування бази даних;
- обслуговуючий персонал мати мінімальну кваліфікацію налаштування та роботи з базою даних MySQL та розробку на мові Java.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на персональних комп'ютерах, ноутбуках зі встановленою віртуальною машиною Java версії 11 та новіше.

4.4.2 Мінімальна конфігурація технічних засобів:

- Вільне місце - 100мб;
- Об'єм ОЗП - 2 гб.

4.5 Вимоги до інформаційної та програмної сумісності

- програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows (Windows 10, Windows 11) або Unix(Linux, MacOS);
- програма має бути розроблена на мові Java версії 11;
- захист інформації повинен відбуватися надійними алгоритмами шифрування та хешування;
- програмне забезпечення повинно використовувати фреймворк Spring.

					КПІ.ІТ-8223.045430.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред'являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати виконавчий архів програмного забезпечення.

					КПІ.ІТ-8223.045430.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		8

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6):

- технічне завдання;
- керівництво користувача;
- керівництво системного програміста;
- керівництво адміністратора;
- програма та методика тестування.

5.4 Графічна частина повинна бути виконана не менше ніж на 4 аркушах формату А3, котрі включаються у якості додатків до пояснювальної записки:

- Схема структурна варіантів використання;
- схема структурна діяльності;
- схема бази даних;
- схема бізнес-процесів програмного забезпечення;
- схема структурна класів.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	28.04	
2.	Розробка технічного завдання	29.04	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	05.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	13.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	15.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	20.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	21.04	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	22.04	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

					КПІ.ІТ-8223.045430.01.91	Арк.
						11
Змін.	Арк.	№ докум.	Підп.	Дата.		

Пояснювальна записка до дипломного проєкту

на тему: Програмне забезпечення для прийому та обробки замовлень в
ресторанному бізнесі

КПІ.ІТ-8223.045430.02.81

Київ – 2022

ЗМІСТ

1	АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	6
1.1	Загальні положення	6
1.2	Змістовний опис і аналіз предметної області.....	6
1.3	Аналіз успішних ІТ-проектів	7
1.3.1	Аналіз відомих технічних рішень.....	7
1.3.2	Аналіз відомих програмних продуктів.....	9
1.4	Аналіз вимог до програмного забезпечення	11
1.4.1	Розроблення функціональних вимог	11
1.4.2	Розроблення нефункціональних вимог	16
1.4.3	Постановка комплексу завдань модулю.....	17
1.5	Висновки по розділу.....	18
2	МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	20
2.1	Моделювання та аналіз програмного забезпечення	20
2.2	Архітектура програмного забезпечення.....	22
2.2.1	Опис архітектури програмного забезпечення.....	26
2.3	Аналіз безпеки даних	50
3	АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 52	
3.1	Аналіз якості ПЗ	52
3.2	Опис процесів тестування.....	53
3.3	Опис контрольного прикладу	54
4	ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .	69
4.1	Розгортання програмного забезпечення.....	69
4.2	Робота з програмним забезпеченням.....	69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–Integrated Development Environment – інтегроване середовище розробки.
API	–Application programming interface, прикладний програмний Інтерфейс
SDK	– Software development kit
IT	– Інформаційні технології
ER	– Entity-Relation diagram
OC	– Операційна система
БД	– База даних.
POS	– Point of sale(точка продажу)
ПЗ	– програмне забезпечення
OMS	– Order Management System
DAO	– Data Access Object

Вступ

В сучасному суспільстві усюди використовується інформаційні технології щоб оптимізувати та пришвидшити бізнес-процеси у різних сферах. Ресторанний бізнес також не є виключенням. Завдяки втручання інформаційних технологій з'являється можливість збереження замовлень, клієнтів, блюд, їх автоматичне створення, обробка та облік, а також, швидкий доступ до них багатьом користувачам одночасно та на різних відстанях. Тому для полегшення адміністрування та для підвищення продуктивності впроваджують інформаційні технології у ресторанний бізнес.

Через необхідність відстежувати багато різних процесів та влаштовувати взаємодію різних відділів в одному закладі: кухні, офіціантів, менеджерів, бухгалтерії, зіставлення меню, закупівлі продуктів та клієнтів зростає необхідність та складність автоматизації цих бізнес процесів. Основним потоком бізнес процесів є клієнти що роблять замовлення, кухня що виконує ці замовлення та офіціанти що обслуговують клієнтів. Тому автоматизувавши взаємодію між ними можна значно зменшити навантаження на людські ресурси та пришвидшити обслуговування клієнтів.

На ринку існує багато рішень для розв'язання різних питань в ресторанному бізнесі: бухгалтерський облік, резервування столів, менеджмент інвентаря, аналітика показників ресторану, зіставлення меню, контроль потоку грошей, прийому онлайн замовлень, управління персоналом, POS(Point of sale). Але в загальному всі вони окремі програми і їх інтегрування може викликати деякі труднощі. Найнеобхіднішим є системи POS, що являє собою програмне забезпечення встановлене на пристроях різного типу і дозволяє обслуговуючому персоналу швидко та зручно створювати замовлення клієнтів та обробляти їх. Існує багато провідних компаній що надають вище вказаний тип програмного забезпечення, такі як Oracle, Square, Harbortouch. Сучасні, ефективні та зручні програми для обробки замовлень коштують багато та складні і коштовні в обслуговуванні або збирають процент

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

від кожного замовлення. Також вони нагромаджені різними функціями та не зрозумілі в використанні тому потребується час для розуміння.

Ресторани з невеликим бюджетом за частую не можуть дозволити собі отримати якісну та зрозумілу програму зі всіма потрібними функціями для обробки заказів за невеликі кошти. Таким чином з'являється необхідність в програмному забезпеченні що зможе задовільнити ці вимоги.

Завданням стане розробка безкоштовного програмного забезпечення для виконання швидкого, якісного та зручного прийому, обробки редагування та зберігання замовлень і клієнтів. Також програма буде мати можливості автоматичної передачі замовлень на виконання, створення замовлень без втручання офіціанту, розподіл чеку по клієнтах.

Застосуванням буде інтеграція програми в ресторани для виконання функцій прийому, обробки та редагування замовлень.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						5
Змін.	Арк.	№ докум.	Підп.	Дата.		

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Ресторан – багато ручної однотипної роботи, світ прямує до оптимізації. Сучасний світ прямує до зменшення ручної роботи та автоматизації її. Ресторанний бізнес гарний приклад однотипної роботи, тому оптимізація перебігання бізнес процесів відіграє ключову роль в збільшенні прибутковості бізнесу, а іноді і його виживання. Оптимізація, зручність в використанні, швидкодія та робота без перебоїв віршують наскільки якісне те чи інше програмного забезпечення, в ресторанному бізнесі це також грає ключову роль. Безперебійна робота програми буде гарантувати стабільний процес обслуговування клієнтів, а зручність в використанні, швидкодія та оптимізація зменшити витрачання часу на одноманітні дії, тим самим збільшивши пропускну здатність ресторану, задоволення клієнтів та покращення інших бізнес-процесів. Також програмне забезпечення має гарантувати збереження всієї інформації та швидкий і одночасний доступ до неї. Тому сучасне програмне забезпечення має гарантувати вище вказані параметри для покращення ефективності управління. Через необхідність, системи POS(Point of Sale), основною задачею яких є прийом і обробка замовлень, почали розробляти ще у 1970-х роках і мали дуже примітивний вигляд. З 90-х років ці системи вже мали програмне забезпечення і постійно модифікувалися з того часу, таким чином сьогодні вони мають різні реалізації та масштаби і можуть покривати усі необхідні вимоги сучасного ресторанного бізнесу.

1.2 Змістовний опис і аналіз предметної області

POS (Point of Sale) – термін, що позначає точку яка дозволяє користувачу створювати, передавати, записувати, оброблювати, редагувати та завершати замовлення. Можна виділити основні відмінності: за

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						6
Змін.	Арк.	№ докум.	Підп.	Дата.		

пристроями з вибором пристрою для встановлення, за функціоналом та за децентралізацією. касовий апарат з програмним забезпеченням та просто програмне забезпечення з вибором пристрою для встановлення Пристроями можуть слугувати комп'ютери, планшети, мобільні телефони, а також касові апарати. Також точки продажу можуть відрізнятися за функціоналом, вони можуть мати базовий функціонал та розширений. Базові POS можуть виконувати всі необхідні функції, тобто завдяки ним обслуговуючий персонал може створювати, редагувати, обробляти, оплачувати та закривати замовлення. Цей варіант гарно підходить для невеликих ресторанів. Продвинуті точки продажу можуть мати великий функціонал, завдяки ним можна відстежувати історію кількості клієнтів, транзакції, тренди. Також вони можуть відстежувати товари та повідомляти при недостачі, управляти персоналом, аналізувати та надавати вказівки по найприбутковішим позиціям та робити підказки для покращення меню. Ще є можливість централізованої та децентралізованої системи, тобто програма працює на сервері та всі термінали підключаються до нього, або є єдина точка доступу. Таким чином існують різні імплементації що підійдуть під різні потреби ресторанів. Для великих закладів з багатьма місцями посадки треба брати децентралізовану систему з декількома точками доступу і з розширеним функціоналом для аналізу всіх даних та підвищення продуктивності. Для малих закладів достатньо мати єдину точку продажу або декілька на пристроях, а вибір об'єму функціоналу може мінятися від потреб, бажаних результатів та фінансування.

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих технічних рішень

Існують різні рішення для обробки замовлень в ресторанному бізнесі. Одне програмне забезпечення дозволяє оброблювати та сплачувати замовлення, інше має наміри контролювати, налаштовувати, оптимізувати та

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

автоматизувати різні аспекти обробки замовлень з збереженням історії продажів, або всі бізнес процеси закладу.

Термінали оплати – найпростіший варіант створення і обробки замовлення, де клієнту створюють його замовлення і він його оплачує. Зазвичай представляє собою касовий апарат з не складним програмним забезпеченням для виконання мінімальних функцій роботи із замовленням та оплати.

POS – рашніше регламентувалося як місце де клієнт може оплатити замовлення. З розвитком технології це точка з програмним забезпеченням яке виконує функції прийому, обробки, зміни, оплати, завершення і прискорення виконання замовлень. Одним із важливіших аспектів є збереження історії в базах даних. Зазвичай мають розширені функції для допомоги в додатковій оптимізації, збиранні та аналізу даних.

OSPOS – опен-сорсне програмне забезпечення, заточений під веб використання та написаний на мові програмування PHP з використанням MySQL для менеджменту бази даних. Це програмне забезпечення повністю покриває основні потреби при обробленні замовлень. Дозволяє персональне налаштування та аналітику даних.

SambaPOS – опен-сорсне програмне забезпечення, повністю написане на мові програмування C#. Має примитивний та застарілий інтерфейс, але буди використане для обробки замовлень, додатковим функціоналом програма не виділяється.

WallacePOS – опен-сорсне програмне забезпечення. Написане веб засобами для онлайн використання, без встановлення на апарати. Підтримує багато різних пристроїв та принтери чеків. Має основні функції для обробки замовлень, а також аналітика по продажах.

OMS – Комплексна система менеджменту, яка включає функції для контролю та управління ресторанним бізнесом з основною функцією прийом та обробка замовлень. Ці системи направлені на покриття усіх потреб в автоматизації закладів і мають технічні рішення для управління

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						8
Змін.	Арк.	№ докум.	Підп.	Дата.		

співробітниками, відстеженням та аналізом продаж і маркетингу, контролем продуктів, відстеження клієнтів, доставки і багатьох інших бізнес-процесів.

1.3.2 Аналіз відомих програмних продуктів

Серед сучасних POS систем, що використовуються в світі можна виділити декілька сучасних проектів, які відрізняються складністю та областю призначення. Це Micros by Oracle, Square, Revel, Upserve, Touch Bistro, Toast, Clover, Lightspeed, в СНГ регіоні успішним є система R-Keeper.

Micros - це велика та комплексна POS система, створена компанією Oracle, та надсилається на різних видах апаратних систем від Oracle. Компанія надає можливість встановлення програмного забезпечення на пристрої власного виробництва: великі та маленькі касові апарати, планшети та станції самообслуговування. Також опцією є встановлення кухонного дисплею, що буде відображати замовлення з різних джерел та завдяки якому можна буде організовувати та пріоритезувати виконання замовлень на кухні. Разом із великою різноманітністю пристроїв, програмне забезпечення виділяється значною гнучкістю в налаштуванні і незлічений функціонал та можливість його доповнення. Micros має гарних захист, можливість приймати онлайн замовлення, доставку та ефективно організовувати роботу кухні, а також швидке з'єднання з менеджерами. Однією з найбільш новітніх функцій це встановлення програмного забезпечення на хмарні сервіси та доступ до них через точки продажу. Якраз великий функціонал та можливість автоматизувати багато сторін бізнесу і робить micros однією з популярніших платформ для ресторанного бізнесу середніх та вищих класів. Недоліком може слугувати велика ціна встановлення та підтримки системи.

Square – доволі нова POS система від компанії Square Capital. Стала популярною завдяки її простоті, дешевизни та зрозумілому інтерфейсу, а також можливості оплати завдяки смартфонам. Square пропонує безкоштовне встановлення та налаштування, в якості оплати вони знімають процент з оплати кожного замовлення. Програмне забезпечення може бути встановлене

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

на касовий регістр, але вони спеціалізуються на малих пристроях як планшети, смартфони або мобільні термінали. Для оплати смартфоном компанія пропонує спеціальний зчитувальний пристрій, що вставляється в роз'єм для зарядки. З розширеного функціоналу надається можливість менеджменту персоналу та налаштування маркетингу. За додаткову ціну ресторанам надається ринок з великою кількістю сторонніх додатків і можливість легко підключати їх до своїх POS для додавання незліченої кількості різних функцій. Ця система користується попитом серед ресторанів з невеликим бюджетом, завдяки дешевизні, великим можливостям персоналізації та використанням невеликих пристроїв для користуванням точкою доступу, але в безкоштовній версії бракує аналітики продаж.

Lightspeed – програмне забезпечення POS, створено компанією LightSpeed, основною задачею якої є оптимізація та максимальне пришвидшення процесу обробки замовлень для закладів швидкого типу як бар або кафе. Встановлюється програма на планшети або мобільні пристрої та ефективно справляється з великою кількістю онлайн замовлень та замовлень на виніс. Також надається можливість легкого створення та налаштування своєї системи доставки, що є значним плюсом, порівняно з конкурентами. Компанія також поставляє функцію аналітики продажів та покращення маркетингу, що також допоможе підняти ефективність ресторану. Тому Lightspeed стає ідеальним вибором для закладів з малим часом виконанням замовлень, де мінімалізм та швидкодія стають основними вимогами бізнесу.

TouchBistro – програмне забезпечення для POS систем, зроблено компанією TouchBistro, яке розроблялося під всі вимоги ресторанного бізнесу. Має дуже простий та інтуїтивно зрозумілий інтерфейс та одночасно дуже багато складних та корисних функцій за досить невелику ціну. З додаткових та корисних технічних можливостей можна виділити відстеження та менеджмент продуктів, розділ чеку по замовленням клієнтів, резервація столів, легке налаштування маркетингу та програма лояльності. Можна встановлювати на планшети, мобільні або спеціально призначені для цього

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		10

пристрої. TouchBistro може ефективно справлятися з усіма об'ємами надходження замовлень і з різних джерел тому підходить для ресторанів усіх типів та розмірів, а за додаткову ціну можна налаштувати онлайн замовлення. Через велику кількість корисних допоміжних функції для покращення обробки замовлень та оптимізації і автоматизації за не велику ціну програма стає універсальною POS для будь-якого ресторану.

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Розроблення функціональних вимог

Діаграма варіантів використання зображення на рисунку 1.1.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						11
Змін.	Арк.	№ докум.	Підп.	Дата.		

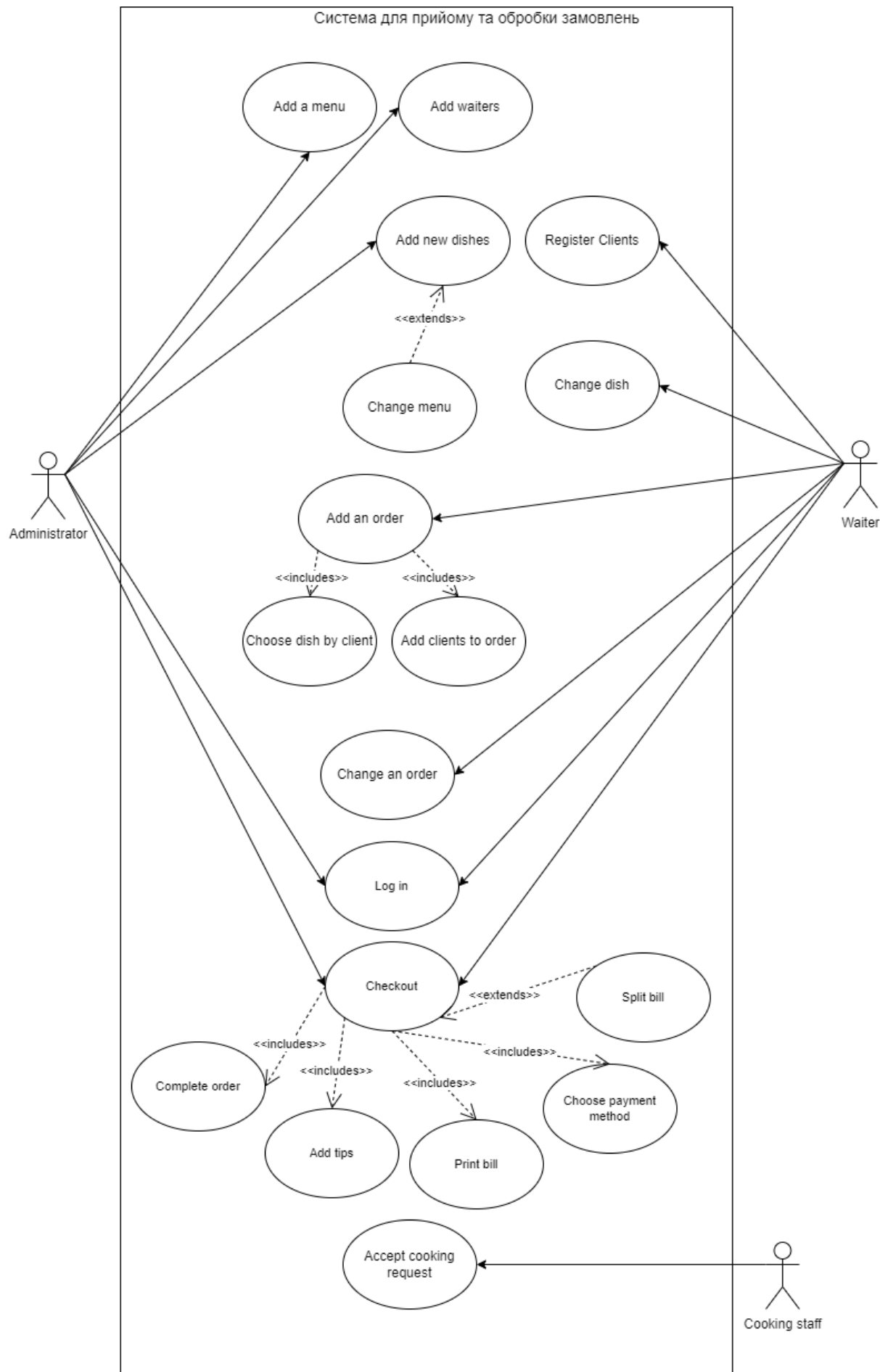


Рис. 1.1 – Схема структурна варіантів використання

Далі в таблиці 1.2 наведені функціональні вимоги та їх пріоритети.

Таблиця 1.2 – Функціональні вимоги та їх пріоритети

Варіант використання	Функціональна вимога	Пріоритет	Виконавець
Add waiter	1. Програма має зареєструвати нового офіціанта та надати йому доступ до системи.	Середній	Адміністратор
Add new dishes	2. Програма має додати нове блюдо до списку блюд. 2.1 Програма має надати можливість змінити меню	Середній	Адміністратор
Change dish	3. Програма має дати можливість внести зміни до існуючої страви.	Середній	Адміністратор
Register Client	4. Програма має зареєструвати нового клієнта.	Високий	Адміністратор, Офіціант
Add an order	5. Програма має додати нове замовлення. 5.1 Програма має дати можливість додати кількість клієнтів. 5.2 Програма має надати можливість розподілити замовлення по кожному клієнту.	Високий	Адміністратор, Офіціант
Change an order	6. Програма має дати можливість користувачу змінити замовлення. 6.1 Програма має змінити блюда та клієнтів в замовленні.	Середній	Адміністратор, Офіціант

Продовження таблиці 1.2

Log in	7. Програма має дати можливість користувачу увійти до системи завдяки своїм вхідним даним.	Низький	Адміністратор, Офіціант, Кухарь
Checkout	8. Програма має дати можливість зробити закінчення замовлення. 8.1 Програма має дати користувачу можливість вибрати метод оплати. 8.2 Програма має дати можливість додати чайові так їх кількість. 8.3 Програма має дати можливість розділити чек по клієнтам. 8.4 Програма має роздрукувати чек на кожного клієнта. 8.5 Програма має завершити замовлення та зберегти його у базі даних.	Високий	Адміністратор, Офіціант
Accept cooking request	9. Програма має дати можливість кухарю прийняти замовлення для виготовлення.	Високий	Кухарь
Add cooked dishes	10. Програма має дати кухарю можливість відмітити страви що вже приготовлені	Середній	Кухарь

Кінець таблиці 1.2

Call waiter for pickup	11. Програма має дати можливість викликати офіціанта щоб зібрати готові страви.	Високий	Кухарь
Finish cooking order	12. Програма має дати кухарю позначити замовлення приготованим. 12.1 Програма має викликати офіціанта щоб зібрати готові страви.	Високий	
Add a menu	13. Програма має додати меню. 13.1 Програма має дати можливість заповнити інформацію про меню	Нормальний	Адміністратор

Далі розписано основну структуру діяльності для опрацювання замовлень. Спочатку користувач входить в систему під своїм телефоном та паролем, далі можна вибрати дії, адміністраторські дії не будуть доступні офіціантам. Користувач може створити, змінити або розрахувати замовлення. Для адміністратора доступні дії створення меню, додавання нових страв та персоналу.

На рисунку 1.3 наведена схема основного алгоритму, за яким персонал може використовувати програму.

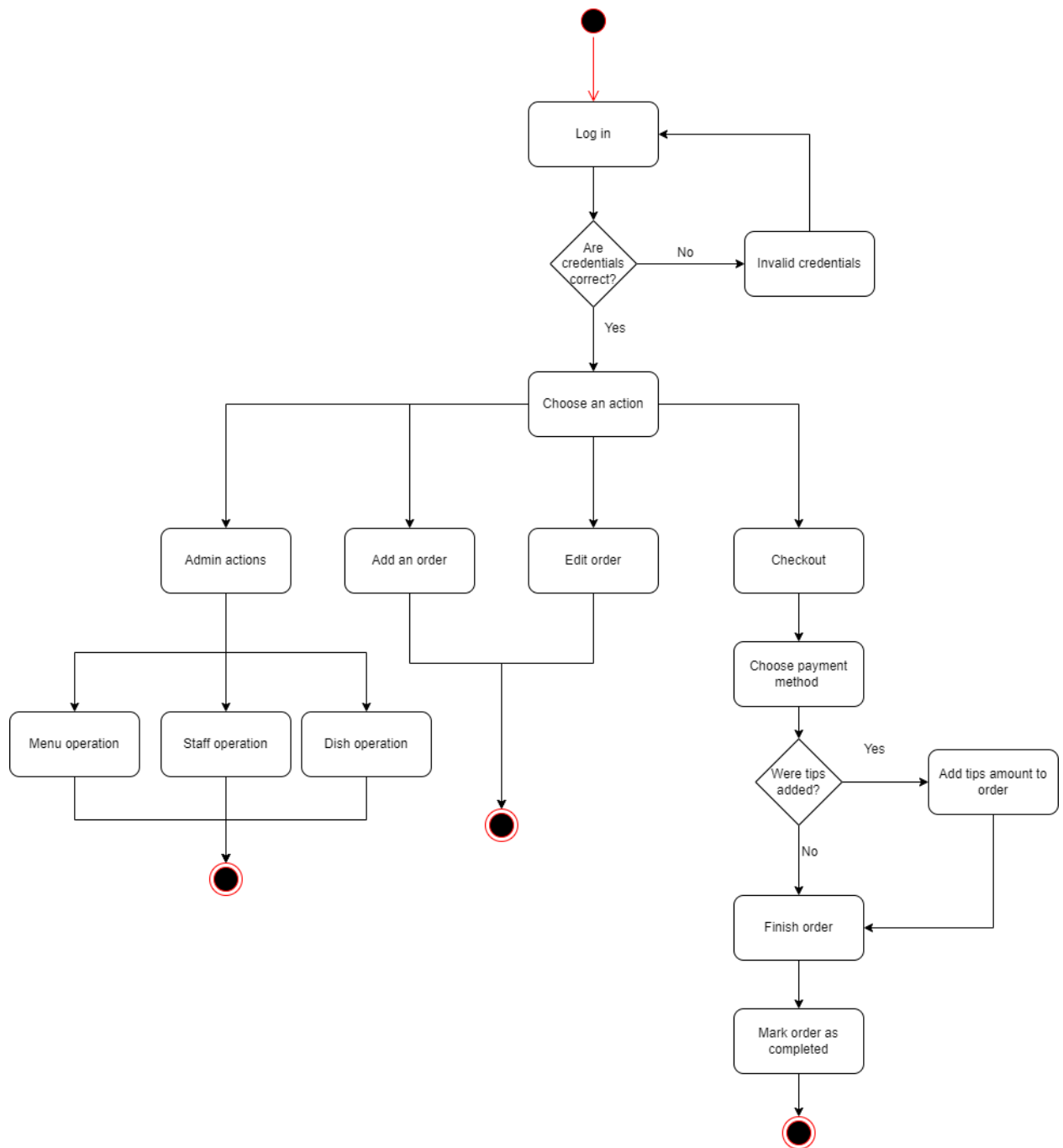


Рис. 1.3 – Спрощена схема структурна діяльності

1.4.2 Розроблення нефункціональних вимог

Були виділені наступні нефункціональні вимоги:

- продукт повинен працювати на операційній системі Windows 10 та новіше;
- дані клієнтів та працівників мають зберігатися у зашифрованому виді в реляційній базі даних;

- на комп'ютері має бути встановлені Java версії 8 та вище;
- інтерфейс має бути простий та інтуїтивно зрозумілий для користувача;
- правильна обробка некоректних дій та некоректно введених даних користувачем;
- програма має безотказно працювати 91% або більше часу;
- відновлюваність системи має бути швидкою та зрозумілою для користувача.

1.4.3 Постановка комплексу завдань модулю

Призначення розробки є програмне забезпечення, що дозволяє обслуговуючому персоналу ресторану створювати, оброблювати, змінювати, завершувати замовленнями клієнтів, а також надсилати замовлення на приготування до кухарів та повідомляти про приготовлене замовлення.

Метою розробки є підвищення ефективності, простоти використання та розуміння програми при створенні, обробки, редагуванні, завершенні замовлень та приготуванні страв, завдяки надання інтуїтивно зрозумілого інтерфейсу та оптимізації роботи з програмним забезпеченням. Для досягнення назначеної мети потрібно розібратися з наступними питаннями:

- визначити елементи інтерфейсу що погіршують вигляд та розуміння програми;
- розробити оптимальний спосіб збереження та діставання інформації з бази даних;
- визначити оптимальний метод входу користувачів до програмного забезпечення для прискорення цього процесу. Він має бути оптимізований щоб займати менше часу та рідше запитувати повторної авторизації;
- створити модуль реєстрації та авторизації. Модуль має забезпечувати безпечного збереження та діставання даних;
- створити функцію додавання та зміни меню;

– створити модуль завершення замовлення з урахуванням всіх можливих випадків.

1.5 Висновки по розділу

В результаті аналізу вимог до програмного забезпечення, було проведено аналіз предметної області – POS тип системи обробки замовлень. Ці системи мають широку розповсюдженість в використанні та вважаються необхідним для ресторанного бізнесу. Є різні реалізації що підійдуть під різні типи та потреби ресторанів, від маленьких кафе до великих закладів престижного класу. Також різні системи можуть привносити свої додаткові особливості, що можуть покращити перетікання тих чи інших бізнес-процесів.

Було досліджено алгоритм роботи системи для різних користувачів: офіціанта, адміністратора, кухаря, який складається з прийому замовлення, вибору кількості клієнтів, додавання страв для кожного клієнта, потім очікування приготування, розрахунок клієнтів з вибором типу оплати та додаванням чайових і завершення замовлення для офіціанта або адміністратора.

Було проведено дослідження технологічних рішень обробки замовлень: OSPOS, SambaPos та WallacePOS, які мають різні технологічні рішення, наприклад веб реалізацію, або програмне забезпечення для встановлення для реалізації потрібних вимог. Також були проаналізовані програмні продукти: micros, TouchBistro, Lightspeed, Square, Upserve, Revel, Toast. Було виявлено що в загальному кожний продукт спеціалізується на своєму типі ресторану і мають нагромаджений інтерфейс. Також вони коштують багато або збирають великий процент з продаж. Але вони також надають ряд корисних функцій для менеджменту, аналітики, онлайн продажів або доставки. Тому хоча вони і дають можливість комплексного управління, аналізу та автоматизації, користування ними може бути не оптимальним. Виходячи з цього було вирішено створення безкоштовного та легкого для розуміння програмного продукту для прийому та обробки замовлень в ресторанному бізнесі.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		18

Для досягнення цілей були сформульовані головні вимоги створюваного програмного забезпечення: надання повного функціоналу прийому, обробки та закінчення замовлень, а також можливість налаштовувати персонал, створювати і редагувати меню зі стравами та працювати із клієнтами.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						19
Змін.	Арк.	№ докум.	Підп.	Дата.		

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Загальні процеси, що проходить користувач для прийому, обробки, зміни, завершення замовлень, також для налаштування меню, управління клієнтами, персоналом.

Послідовний опис прийому замовлення:

- користувач має увійти в систему за своїми персональними даними. У разі успішного входу йому буде пропоновано обрати дію, в інакшому випадку програма запросить повторне введення даних;
- наступним кроком користувач вибирає номер столу та вводить кількість клієнтів, потім для кожного клієнту користувач вибирає страви що були замовлення;
- після введення даних користувач має підтвердити коректність замовлення і воно відправляється на приготування.

Послідовний опис зміни замовлення:

- авторизований користувач вибирає активне замовлення що хоче змінити та може змінити кількість клієнтів, самих клієнтів або замовленні ними страви;
- нові страви відправляються до кухні на приготування.

Послідовний опис завершення замовлення:

- авторизований користувач натискає на активне замовлення та вибирає функцію розрахувати;
- наступним кроком є можливість розділити чек по клієнтам та надрукувати окремо;
- користувач вибирає спосіб розрахунку;
- після розрахунку користувач має вибрати кількість доданих чайових. Після чого замовлення буде завершено, збережене в базу даних та помічено як закінчене.

Послідовний опис обробки клієнта персоналом:

- персонал отримує дані від клієнта і після цього здійснює реєстрацію;
- при невірному вводі або зміні даних можна редагувати користувача;
- після реєстрації персонал може додати конкретного клієнта до замовлення.

Послідовний опис видалення клієнта персоналом:

- персонал вибирає клієнта для видалення та видаляє його;
- після видалення всі дані клієнта будуть видалені з бази.

Послідовний опис зміни клієнта персоналом:

- персонал вибирає клієнта для зміни даних, та вводить оновлені дані;
- після підтвердження зміни персональні дані клієнта будуть оновлені.

Послідовний опис реєстрації нового користувача:

- авторизований користувач із позицією адміністратор вводить ім'я, прізвище, вік, номер телефону, зарплатню та позицію та реєструє його;
- новий користувач буде доданий до програми та записаний у базу даних. Після цього авторизація під цим користувачем буде доступна.

Послідовний опис зміни користувача:

- авторизований користувач вибирає персонал для зміни та вводить нові дані та виконує зміну;
- нові дані персоналу замінюють застарілі дані користувача та виконується зміна у базі даних.

Послідовний опис додавання меню:

- авторизований користувач із правами адміністратора вводить назву, опис, тип, дату початка та дату кінця і додає меню.
- Користувач вводить назву, опис, інгредієнти, алергени, тип, категорію, ціну страви та додає до меню.

					КПІ.ІТ-8223.045430.02.81 ПЗ	Арк.
						21
Змін.	Арк.	№ докум.	Підп.	Дата.		

Послідовний опис зміни меню та страв:

- авторизований користувач із правами адміністратора вводить нові дані меню;
- користувач може вибрати страву та ввести нові дані, після цього її буде змінено;
- користувач за потреби видаляє страви з меню.

Послідовний опис видалення меню:

- авторизований користувач із правами адміністратора вибирає меню та видаляє його;
- меню та всі страви видаляються із програми та бази даних.

Детальний опис бізнес-процесів, що перебігають в програмі, представлено у додатку графічних матеріалів на схемі 7.

2.2 Архітектура програмного забезпечення

Програмне забезпечення складається з 5 основних модулів, які створені для виконання своїх задач:

- модуль роботи з базою даних;
- модуль сервісів;
- модуль сутностей;
- модуль інтерфейсу користувача.

Для створюваного програмного забезпечення необхідно ретельно працювати із сутностями, для реалізації усіх вимог були створені наступні сутності: Dish, Staff, Order, Menu, Client та допоміжні ClientsToOrders, DishesToMenus.

Сутність Order репрезентує замовлення та має такі параметри як номер столу та кількість клієнтів, а також посилання на клієнтів і замовлених ними страв та посилання на персонал, який обслуговував замовлення.

Сутність Staff репрезентує персонал, що працює в ресторані і має такі поля як ім'я, прізвище, вік, зарплатню, позицію та номер телефона. Вона зберігає увесь персонал.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		22

Сутність Client репрезентує клієнта та має поля: ім'я, прізвище, номер телефона, електронна пошта, дата народження та адреса. Використовується для збереження всіх клієнтів.

Сутність Dish репрезентує страву і має такі поля як: назва, опис, інгредієнти, алергени, тип, категорія. Створена для збереження страв в ресторані.

Сутність Menu створена для репрезентації меню та має поля: назва, опис, тип, дата початку, дата кінця та активне. Зберігає різні меню в ресторані. На цю сутність засилаються страви.

Сутність ClientsToOrders з'єднує конкретного клієнта за його стравою та замовленням. Має посилання на клієнта, страву та замовлення.

Сутність DishesToMenus з'єднує страви з меню. Має посилання на меню та страву.

Зв'язок між сутностями у вигляді спрощеної схеми бази даних представлено на рисунку 2.1. Повна структурна діаграма бази даних відображена у додатках в розділі графічних матеріалів під номером 8.

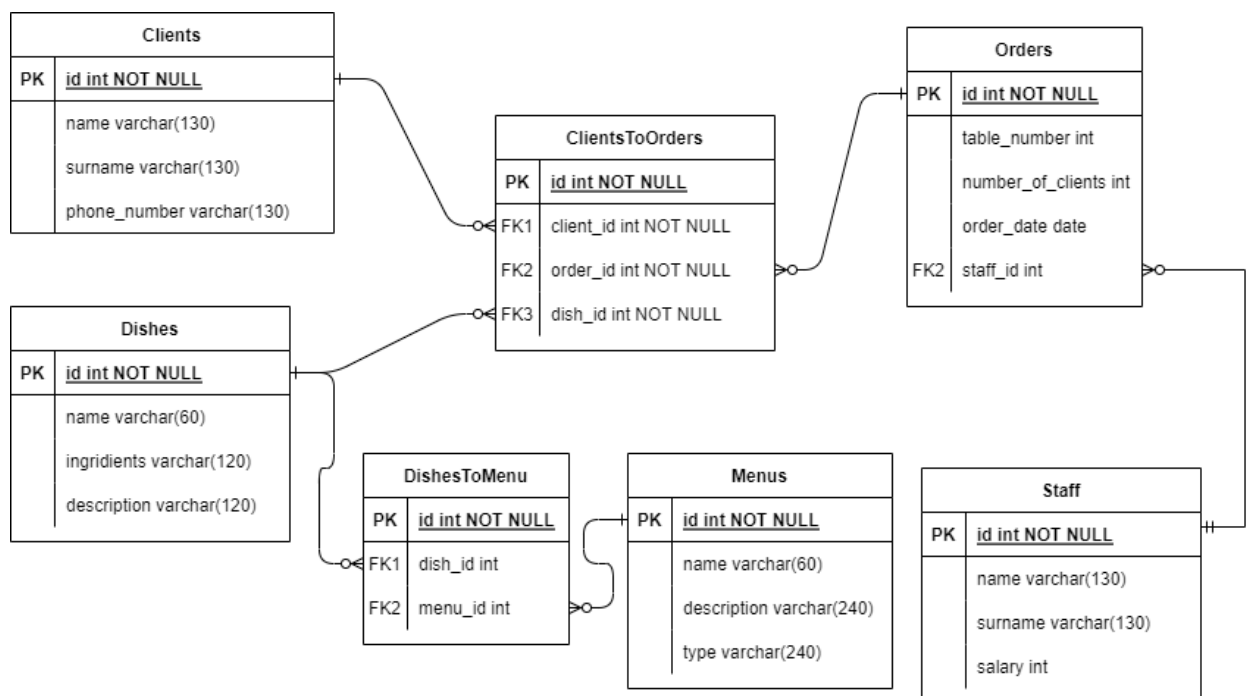


Рис. 2.1 – Спрощена схема бази даних

Далі в таблиці 2.1 наведений детальний опис структури бази даних.

Таблиця 2.1 – Опис структури бази даних

Назва таблиці	Назва поля	Тип поля	Допустимі нульові значення	Спеціальні позначки
Clients	id	int	ні	Primary key
	name	varchar(130)	ні	-
	surname	varchar(130)	ні	-
	phone_number	varchar(130)	ні	-
	address	varchar(150)	ні	-
	email	varchar(130)	ні	-
	iv	varchar(256)	ні	-
Staff	id	int	ні	Primary key
	name	varchar(130)	ні	-
	surname	varchar(130)	ні	-
	password	varchar(300)	ні	-
	salt	varchar(130)	ні	-
	phone_number	varchar(130)	ні	-
	age	varchar(130)	ні	-
	position	varchar(130)	ні	-
	iv	varchar(256)	ні	-
Dishes	id	int	ні	Primary key
	name	varchar(60)	ні	-
	ingridients	varchar(120)	ні	-
	description	varchar(120)	ні	-
	alergens	varchar(16)	ні	-
	price	int	ні	-
	type	varchar(30)	ні	-
	category	varchar(30)	ні	-
Menus	id	int	ні	Primary key

Кінець таблиці 2.1

	name	varchar(60)	ні	-
	description	varchar(240)	ні	-
	type	varchar(240)	ні	-
	start_date	date	ні	-
	end_date	date	ні	-
	active	boolean	ні	-
Order	number_of_clients	int	ні	-
	table_number	int	ні	-
	order_date	datetime	ні	-
	staff_id	int	ні	Foreign key
ClientsToOrders	id	int	ні	Primary key
	client_id	int	ні	Foreign key
	order_id	int	ні	Foreign key
	dish_id	int	ні	Foreign key
DishesToMenu	id	int	ні	Primary key
	dish_id	int	ні	Foreign key
	menu_id	int	ні	Foreign key

Для розробки програмного забезпечення було вирішено вибрати мову Java для написання функціональних модулів, це дозволить використовувати програму на різних пристроях та операційних системах. Допоміжним фреймворком був вибраний Spring для реалізації інверсії контролю та ін'єкції залежностей. В якості бази даних був вибран підтип MySQL - MariaDB, через її безкоштовність та ефективність, а також можливість швидше обробляти дані. Для інтерфейсу була вибрана технологія JavaFX, через її зручність, оптимальність та великого функціоналу.

2.2.1 Опис архітектури програмного забезпечення

Програма містить в собі такі модулі: Data Access Module, View, Service Module, Model. Кожен з них відповідає за певну частину функціоналу. В таблиці 2.1 описані кожен модуль та класи, що до нього входять.

Таблиця 2.2 – Опис модулів та належних до них класів

Назва модуля	Клас	Опис та призначення класу
DataAccess	ClientDAO	Клас для виконання операцій над клієнтів в базі даних.
	StaffDAO	Клас для виконання операцій над персоналом в базі даних.
	OrderDAO	Клас для виконання операцій над замовленнями в базі даних.
	MenuDAO	Клас для виконання операцій над меню в базі даних.
	DishDAO	Клас для виконання операцій над стравами в базі даних.
	ClientRowMapper	Клас для перетворення результату SQL запиту в об'єкт клієнт
	DishRowMapper	Клас для перетворення результату SQL запиту в об'єкт страва
	MenuRowMapper	Клас для перетворення результату SQL запиту в об'єкт меню
	OrderRowMapper	Клас для перетворення результату SQL запиту в об'єкт замовлення
	StaffRowMapper	Клас для перетворення результату SQL запиту в об'єкт персонал

Продовження таблиці 2.2

	AESHandler	Клас для зашифровки та розшифровки персональних даних.
Service Module	OrderManager	Клас що займається прийняттям, обробкою та зберіганням активних та історії замовлень.
	MenuManager	Клас що допомагає користувачу створювати і редагувати страви та зіставляти меню.
	ClientManager	Клас що займається обробкою даних клієнтів.
	StaffManager	Клас що займається операціями з персоналом.
	OrderCookingManager	Клас що відстежує та допомагає організувати стан приготування замовлення та страв.
	BillService	Сервіс для формування чеку для друку.
Model Module	Client	Клас-модель клієнта.
	Staff	Клас-модель персоналу.
	Order	Клас-модель замовлення.
	Menu	Клас-модель меню.
	Dish	Клас-модель страви.
View	ManagerController	Відображає та контролює форму для управління замовленнями, персоналом, клієнтами, меню, стравами. Відстежує та оброблює всі елементи управління для взаємодії з користувача з програмним забезпеченням. Відображає різний інтерфейс залежності від посади.

Кінець таблиці 2.2

	LoginController	Відображає форму входу персоналу, відстежує та оброблює взаємодію з формою.
	DialogController	Налаштовує форму діалогу з користувачем та оброблює події виклику нею.
	Main	Клас з якого запускається програма і виконується зміна сцен.

2.2.2 Архітектурні компоненти програмного забезпечення

Модуль роботи з базою даних.

Для збереження та відтворення всіх даних програми використовується реляційна база даних. Це важливий компонент, адже програмне забезпечення має зберігати багато зв'язаної інформації. Цей модуль призначений для забезпечення всіх потреб діставання та завантаження інформації до бази даних. Кожний клас реалізує інтерфейс DAO, що має шаблони методів, та спеціалізується на роботі з базою даних для своєї сутності. Також в модулі є клас для хешування паролей завдяки SHA3-256 і шифрування персональних даних клієнтів та персоналу. Клас використовує метод шифрування AES-GCM. Далі в таблиці 2.2 наведені класи та їх методи з детальним описом.

Таблиця 2.3 – Опис класів та методів модулю роботи з базою даних

Назва класу	Метод	Параметри, Тип значення, що повертається	Опис
ClientDAO DishDAO MenuDAO	add()	Client t void	Виконує Sql запит по додаванню клієнта до бази даних

Продовження таблиці 2.3

	delete()	Client t void	Виконує Sql запит по видаленню клієнта з бази даних
	getById()	Int id Client	Виконує Sql запит по діставанню клієнта по айді з бази даних та перетворення на об'єкт.
	getAll()	Void List<Client>	Виконує Sql запит по діставанню всіх клієнтів по з бази даних та перетворення на список об'єктів.
	update()	Client t void	Виконує оновлення даних вже існуючого клієнту.
	add()	Dish t void	Виконує Sql запит по додаванню страви до бази даних.
	delete()	Dish t void	Видаляє страву і всі її дані з бази даних.
	getById()	Int id Dish	Виконує Sql запит по діставанню страви по айді з бази даних та перетворення на об'єкт.

Продовження таблиці 2.3

	getAll()	void List<Dish>	Виконує Sql запит по діставанню всіх страв з бази даних та перетворення на список об'єктів.
	add()	Menu t	Виконує Sql запит по додаванню меню до бази даних та зв'язування його зі всіма стравами у меню.
	delete()	Menu t	Виконує Sql запит по видаленню меню та усіх зв'язків з базою даних.
	getById()	Int id Menu	Виконує Sql запит по діставанню меню та належних страв по айді з бази даних та перетворення на об'єкт.
	getDishesByMenuId()	Int id List<Dish>	Виконує Sql запити по діставанню всіх страв зв'язаних з конкретним меню.

Продовження таблиці 2.3

	getAll()	Void List<Menu>	Виконує Sql запит по діставанню всіх наявних меню та належних страв з бази даних та перетворення на об'єкт.
	update()	Menu t void	Оновлює дані існуючої страви.
OrderDAO	add()	Order t void	Виконує Sql запит по додаванню замовлення до бази даних та зв'язування зі всіма клієнтами та стравами що вони замовили і персоналом що обслуговував замовлення.
	getById()	Int id Order	Виконує Sql запит по діставанню замовлення, пов'язаних з ним клієнтів, страв та персоналу з бази даних по айді замовлення
	addDishesByClient()	Order t Void	Додає до існуючого замовлення зв'язку з клієнтами та їх стравами.
	selectStaffByOrderId()	Int id Staff	Виконує Sql запит по діставанню персоналу, пов'язаного з замовленням, з бази даних.

Продовження таблиці 2.3

	selectDishByClientOrder() ()	Int id Map<Client>, List<Dish>>	Виконує Sql запити по діставанню клієнта та замовлених ним страв в конкретному замовленні з бази даних по айді замовлення.
	getAll()	Void List<Order>	Виконує Sql запити по діставанню всіх замовлень з бази даних.
	update()	Order order void	Оновлює дані існуючого замовлення.
StaffDAO	add()	Staff t void	Виконує Sql запит по додаванню персоналу до бази даних.
	delete()	Int id void	Виконує Sql запит по видаленню персоналу з бази даних.
	getById()	Int id Staff	Виконує Sql запит по діставанню персоналу по айді з бази даних та перетворення на об'єкт.
	getAll()	Void List<Staff>	Виконує Sql запит по діставанню всього персоналу з бази даних та перетворення на список.
	update()	Staff t void	Оновлює дані існуючого персоналу.

Продовження таблиці 2.3

	login()	String phone, String password Staff t	Перевіряє на співпадіння введених даних з даними персоналу та повертає його при знаходженні.
AESHandler	encrypt()	String data, byte[] iv String	Виконує всі необхідні кроки для зашифровки даних способом блочного шифрування AES-GCM, ініціалізуючим вектором та ключем.
	decrypt()	String data, byte[] iv String	Виконує всі необхідні кроки для розшифровки даних алгоритмом шифрування AES-GCM, ініціалізуючим вектором та ключем.
	takeKeyOut()	void byte[]	Дістає ключ з текстового файлу.
	getKey()	byte[] key SecretKey	Перетворює послідовність байтів у ключ для алгоритму AES.
	generateVector()	Void byte[]	Генерує ініціалізуючий вектор.
	generateBytes()	Int n byte[]	Генерує послідовність байтів заданої довжини.

Кінець таблиці 2.3

	encryptAES()	String data, SecretKey key, byte[] iv String	Зашифровує данні способом блочного шифрування AES- GCM відповідним ініціалізуючим вектором та ключем.
	decryptAES()	String data, SecretKey key, byte[] iv String	Розшифровує данні способом блочного шифрування AES- GCM відповідним ініціалізуючим вектором та ключем.
Sha3256Hasher	encrypt()	String text, Byte[] salt String	Хешує дані за допомогою алгоритму SHA3- 256.
	generateSalt()	Void Byte[]	Генерую випадкову послідованість байтів.

Обробка SQL запитів проводиться завдяки класу JDBCTemplate фреймворку Spring. Перетворення на об'єкти повернутих строк від запитів виконують спеціальні класи маппери.

Модуль сервісів

Даний модуль займається оброкою запитів створених користувачами. Головною задачею модуля є прямою реалізацією всіх вимог до даного програмного забезпечення. Модуль містить завантажену інформацію з бази даних та різні способи її обробки, представлення та надання. OrderManager цього модуля займаються створенням, редагуванням та обробкою замовлень.. MenuManager дозволяють додавати нові страви та редагувати та видаляти їх, а також додавати, змінювати і видаляти меню. ClientManager та StaffManager дозволяють робити операції додавання, змінення та видалення над клієнтами

та персоналом. В таблиці 2.4 наведений опис класів та методів модулю сервісів.

Таблиця 2.4 – Опис класів та методів модулю сервісів

Назва класу	Метод	Параметри, Тип значення, що повертається	Опис
Order Manager	addOrder()	Order order Void	Створює нове замовлення з даними введеними користувачем
	changeOrder()	Int id, Order order Void changed	Змінює параметри конкретного замовлення
	addClients()	Int id, Client client void	Додає нового клієнта до замовлення
	addDishToClientOrder()	int id, int clientId, Dish dish	Додає страву до конкретного клієнту в замовленні.
	deleteDishFromClientOrder()	Int id, Int clientId, Dish dish	Видаляє страву конкретного клієнту в замовленні
	deleteClientFromOrder()	int id, int clientId Void	Видаляє клієнта з замовлення та всі його страви
	getClientsByOrder()	int orderId List<Client>	Знаходить та повертає список всіх клієнтів в замовленні.

Продовження таблиці 2.4

	getDishesByOrder()	Order order List<Dish>	Знаходить та повертає список усіх блюд в замовленні.
	getDishesByActiveOrder()	Int orderId List<Dish>	Повертає список страв в активному замовленні з конкретним айді.
	getDishesByOrderHistory()	Int orderId List<Dish>	Повертає список страв з історії замовлень з конкретним айді.
	findClientInOrderById()	Order order, int clientId Client	Знаходить та повертає клієнта за його айді в замовленні.
	findOrder()	Int id Order	Знаходить та повертає замовлення з активних.
	findOrderHistory()	Int id Order	Знаходить та повертає замовлення з історії.
	addDishes()	Int id, Int clientNum, List<Dish> dishes Void	Додає клієнту, замовлені ним, страви.
	removeDishes()	Int id, Int clientNum, Dish dish Void	Видаляє з замовлення непотрібні страви клієнта
	loadOrder()	Void Void	Завантажує замовлення до програми.

Продовження таблиці 2.4

	checkout()	Order order, Int tips, Boolean split bill, List<List<Integer>> splitting	Фіналізує замовлення, відсилає на друк чек на записує замовлення до бд.
Menu Service	addMenu()	Menu menu Void	Додає нове меню до програми.
	editMenu()	Int id, Menu menu Void	Змінює параметри меню та включені до нього страви за айді.
	editMenu()	String menuName, Menu menu Void	Змінює параметри меню та включені до нього страви за назвою меню.
	findMenu()	Int id Menu	Знаходить та повертає меню за його айді.
	findMenu()	String name Menu	Знаходить та повертає меню за його айді.
	removeMenu()	Int id, Menu menu Void	Повністю видаляє меню з програми за його айді.
	removeMenu()	String menuName, Void	Повністю видаляє меню з програми за його назвою.

Продовження таблиці 2.4

	loadMenus()	Void, Void	Завантажити всі меню до програми.
	addDishToMenu()	Int id, Dish dish Void	Додає нову страву до меню та оновлює його.
	removeDishInMenu()	Dish dish void	Видаляє страву з кожного меню
	findDishInMenus()	Dish dish Dish	Знаходить страву за параметрами в меню та повертає.
	addDishToMenu()	Int id, Int dishId Void	Додає страву с конкретним айді до меню.
	addDish()	Dish dish	Визиває запит на завантаження страви до бази.
	editDish()	Int id, Dish dish void	Змінює параметри страви в системі.
	removeDish()	Int id, Void	Видаляє страву з бази та зі всіх меню.
	findDish()	int id, Dish dish	Знаходить та повертає страву з меню за айді.
	getAllDishesByMenus()	Void Void	Знаходить всі страви зі всіх меню.
	getAllDishes()	Void List<Dish>	Повертає всі страви зі всіх меню.

Продовження таблиці 2.4

	loadDishes()	Void Void	Завантажує з бази даних всі страви.
	findDish()	String name Dish	Знаходить та повертає страви з меню за назвою.
Client Service	registerClient()	Client client Void	Додає нового клієнта до програми
	editClientData()	Int id, Client client Void	Змінює дані клієнта за його айді
	editClientData()	String email, Client client Void	Змінює дані клієнта за його електронною поштою
	deleteClient()	Int id Void	Видаляє клієнта з програми
	findById()	int id Client	Знаходить клієнта за айді та повертає його
	findByMail()	String email Client	Знаходить клієнта за поштою та повертає його
	findByPhone()	String phone Client	Знаходить клієнта за номером телефона та повертає його
	retrieveClients()	Void List<Client> clients	Повертає всіх клієнтів.
	loadClients()	Void void	Завантажує клієнтів з бази даних.

Продовження таблиці 2.4

StaffService	addStaff()	Staff staff Void	Додає новий персонал до програми та бази.
	editStaffData()	Int id, Staff staff Void	Змінює дані персоналу за його айді.
	login()	String phone, String password Void	Відправляє запит на авторизацію до бази даних.
	removeStaff()	Int id Void	Видаляє персонал з програми та ініціює видалення з бази.
	findStaffById()	Int id Staff	Знаходить персонал за айді.
	findStaffByPhone()	String phone Staff	Знаходить персонал за його телефоном та повертає.
	findStaffBySurname()	String surname Staff	Знаходить персонал за його фамілією та повертає.
	getStaff()	Void List<Staff>	Повертає список усього персоналу.
	loadStaff ()	Void void	Завантажує до програми увесь персонал з бази даних.
	markOrderAsCooked()	Order order, Dish dish Void	Помічає замовлення приготуванням.

Кінець таблиці 2.4

BillService	formBill()	Order order, Map<Dish,Integer> dishAmount Void	Формує чек за заданими параметрами та замовленням.
	SplitBill()	Order order, List<List<Integer>> split Void	Ділить замовлення на різні чеки за параметрами та вдіправляє на друк.
	printBill()	String billForm, Void	Друкує чек у задане місце.

Модуль сутностей

Цей модуль спеціалізується на всіх сутностях для роботи програми. Client - клас, що репрезентує сутність клієнта. Order репрезентує замовлення та зберігає список клієнтів і їхніх замовлень. Клас Staff репрезентує сутність усього персоналу, тобто адміністратора, офіціанта та кухаря. Dish – репрезентує страви, а Menu - меню там зберігає в собі всі страви конкретного меню.

Таблиця 2.5 – Опис класів та методів модулю сутностей

Назва класу	Метод	Параметри, Тип значення, що повертається	Опис
Order	setId()	int id Void	Встановлює замовленню айді.
	getId()	Void int	Отримує айді замовлення.
	setClientNumber()	int id Void	Встановлює кількість клієнтів.

Продовження таблиці 2.5

	getClientNumber()	Void int	Отримує кількість клієнтів.
	addClient()	Client client Void	Додає клієнта до замовлення.
Client	setName()	String name void	Встановлює ім'я клієнта.
	getName()	Void String	Повертає ім'я клієнта.
	setSurname()	String surname Void	Встановлює фамілію клієнта.
	getSurname()	Void String	Повертає фамілію клієнта.
Staff	setName()	String name void	Встановлює ім'я персоналу.
	getName()	Void String	Повертає ім'я персоналу.
	setSurname()	String surname void	Встановлює фамілію персоналу.
	getSurname()	Void String	Повертає фамілію персоналу.
Dish	setName()	String name Void	Встановлює назву страви.
	getName()	Void String	Повертає назву страви.
	setType()	String type Void	Встановлює тип страви.
	getType()	Void String	Повертає тип страви.
Menu	setName()	String name Void	Встановлює назву меню.
	getName()	Void String	Повертає назву меню.
	addDish()	Dish dish Void	Додає страву до меню.
	getDishes()	Void List<Dish>	Повертає список страв.

Модуль інтерфейсу користувача

Модуль інтерфейсу користувача забезпечує взаємодію через візуальну оболонку програми. Дані графічним чином відображаються на екрані користувача завдяки різним елементам. Для вводу даних використовуються текстові поля, випадаючі списки та кнопки. Наступні елементи були використані для відображення та обробки даних введених користувачем:

- Button – елемент що дозволяє користувачу натиснути на неї, після чого будуть проведені дії, визначені розробником;
- Scene – головний елемент на яку додаються та розміщуються усі інші елементи;
- Pane – допоміжний елемент для групування та компоновання інших елементів;
- Label – елемент для відображення текстових даних на сцені;
- TextField – поле для введення текстових даних користувачем;
- CheckBox – кнопка, яка при натисканні активується;
- ComboBox – випадаючий список для відображення елементів;
- ListView – Список для відображення великої кількості елементів.

В таблиці 2.6 наведений опис класів та методів інтерфейсу користувача.

Таблиця 2.6 – Опис класів та методів модулю інтерфейсу користувача

Назва класу	Метод	Параметри, Тип значення, що повертається	Опис
DialogController	initialize()	URL location, Resource Bundle resources Void	Ініціалізує параметри діалогового вікна.
	getTips()	Void Int	Повертає чайові, отримані з вікна діалогу.
	getSplitting()	Void String	Повертає тип розділення чеку.

Продовження таблиці 2.6

LoginController	loginButtonAction()	Void Void	Отримує значення логіну та паролю та відправляє запит на збіг з базою. Перемикає на головну сцену разі співпадіння даних.
Manager Controller	searchOrders()	Void Void	Додає на сцену замовлення за знайденими параметрами пошуку.
	updateOrderList()	List<Order> orders Void	Відображає переданий список.
	additionModeToggle()	Void Void	Переключає режим додавання клієнтів та страв.
	saveOrderAction()	Void Void	Отримує параметри з сцени та додає нове замовлення.
	addClientToOrder()	Void Void	Отримує параметри з сцени додає клієнта до замовлення.
	addDishesToOrder()	Void Void	Отримує параметри з сцени додає страву до клієнта у замовленні.
	deleteClientFromOrder()	Void Void	Отримує параметри з сцени та видаляє клієнта.

Продовження таблиці 2.6

	deleteDishesFromOrder()	Void Void	Отримує параметри з сцени та видаляє страви з замовлення.
	chooseOrderAction()	Void Void	Відображає на сцені клієнтів та страви, в залежності від обраного замовлення.
	dishOrderUpdate()	Void Void	Оновлює список всіх страв у замовлення на сцені.
	clientOrderUpdate()	Void Void	Оновлює список всіх клієнтів у замовлення на сцені.
	dishOrderListUpdate()	List<Dish> dishes Void	Додає список всіх переданих страв на сцену.
	clientOrderListUpdate()	List<Client> clients Void	Додає список всіх переданих клієнтів на сцені.
	chooseClientOrderAction()	Void Void	Оновлює список всіх клієнтів у замовлення на сцені.
	editOrderAction()	Void Void	Отримує параметри з сцени та викликає змінення параметрів замовлення.
	checkoutOrderAction()	Void Void	Відображає діалогову сцену, отримує параметри з неї, формує розбиття чеку та викликає завершення замовлення.

Продовження таблиці 2.6

	clientSearchAction()	Void Void	Отримує параметри з сцени та виконує пошук клієнта. Викликає відображення на сцені.
	showClientList()	List<Client> clients Void	Отримує список клієнтів та відображає його на сцені.
	showClients()	Void Void	Викликає відображення всіх клієнтів на сцені.
	clientListChooseAction()	Void Void	Вибирає клієнта зі списку та встановлює дані у поля.
	clientRegisterAction()	Void Void	Отримує параметри з сцени та викликає реєстрацію нового клієнта.
	clientEditAction()	Void Void	Отримує параметри з сцени та викликає зміну параметрів клієнта.
	clientDeleteAction()	Void Void	Отримує клієнта з сцени та викликає його видалення. Оновлює список клієнтів.
	showStaffList()	List<Staff> staffs Void	Відображає на сцену переданий список персоналу.

Продовження таблиці 2.6

	showStaff()	Void Void	Викликає відображення всього персоналу.
	staffListChooseAction()	Void Void	Отримує вибраний персонал з сцени та заповнює текстові поля даними.
	staffAddAction()	Void Void	Отримує параметри з сцени та викликає додавання персоналу з цими параметрами.
	staffEditAction()	Void Void	Отримує параметри з сцени та викликає змінення параметрів персоналу.
	staffDeleteAction()	Void Void	Отримує вибраний персонал з сцени та викликає його видалення.
	staffSearchAction()	Void Void	Викликає пошук персоналу за параметром, та виводить його на сцену при співпадінні.
	showMenuList()	List<Menu> menus Void	Відображає на сцену переданий список меню.
	showMenus()	Void Void	Викликає відображення всього персоналу.

Продовження таблиці 2.6

	menuSearchAction()	Void Void	Визиває пошук меню по параметру з сцени та викликає відображення результату.
	addMenuAction()	Void Void	Отримує параметри меню з сцени та викликає додавання меню з ними.
	editMenuAction()	Void Void	Отримує параметри меню з сцени та викликає змінення параметрів існуючого меню.
	deleteMenuAction()	Void Void	Отримує вибране меню та викликає його видалення.
	chooseMenuAction()	Void Void	Отримує параметри меню з сцени та виводить їх у текстові поля. Викликає відображення страв принаджених до цього меню.
	addDishAction()	Void Void	Отримує параметри страви з сцени та викликає додавання страви з цими значеннями.

Кінець таблиці 2.6

	editDishAction()	Void Void	Отримує параметри страви з сцени та викликає змінення вибраної страви.
	deleteDishAction()	Void Void	Отримує вибрану страву з сцени та викликає її видалення.
	chooseDishAction()	Void Void	Отримує вибрану страву з сцени та виводить її параметри у текстові поля. Якщо страва не вибрана, очищає текстові поля.
	showDishes()	Void Void	Викликає виведення всіх страв.
	logout()	Void Void	Виходить з акаунту користувача та виводить сцену авторизації.
	showDishesList	List<Dish> dishes Void	Виводить список страв на сцену.
Main	start()	Stage primaryStage Void	Встановлює загальні параметри інтерфейсу.
	initRootLayout()	Void Void	Завантажує та відображає сцену входу.
	initManagerLayout()	Void Void	Завантажує та відображає сцену управління.
	main	String[] args Void	Запускає графічний інтерфейс.

2.3 Аналіз безпеки даних

Розроблюване програмне забезпечення потребує збереження персональних даних користувачів та клієнтів. Збереження вразливих даних в чистому вигляді підвищує ризик їх викрадення та гарантує його при взломі зловмисниками бази даних. Тому персональні дані зберігаються у зашифрованому вигляді. Для шифрування даних нашого програмного забезпечення було вибрано поширений алгоритм шифрування – AES(Advanced Encryption Standard) у режимі роботи GCM(Galois Counter Mode), що забезпечує безпеку персональних даних та вважається дуже надійним. Ключ шифрування зберігається в файловій системі на сервері. Для паролей користувачів був вибран алгоритм хешування SHA3-256 із додаванням солі, він має дуже маленьку швидкість підбирання паролей, що робить підбор нераціональним і тим самим забезпечує конфіденційність паролю.

Висновки до розділу

На даному етапі було змодельоване програмне забезпечення яке оброблює всі прописані варіанти використання. В ході моделювання були визначені 4 основні модулі: модуль роботи з базою даних, модуль сервісів, модуль обробки запитів та модуль інтерфейсу користувача. Разом із цим було розроблено структуру реляційної бази даних та діаграму сутностей, що відображає їх взаємозв'язки.

Модуль роботи з базою даних спеціалізується на задоволення всіх потреб збереження, зміни, видалення та діставання даних. Досягається це виконанням складних SQL запити для операції з даними. Потрібні операції реалізовані для всіх існуючих сутностей.

Модуль сервісів надає функціонал для покриття функціональних вимог: оброблює запити від користувачів на створення, зміну, обробку та видалення меню, користувачів, клієнтів, страв, а також замовлень в середині програми.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						50
Змін.	Арк.	№ докум.	Підп.	Дата.		

Модуль обробки запитів створений для комунікації між розділеними частинами програми, наприклад коли кухар отримує запит на приготування замовлення або повідомляє про готовність блюд.

Модуль інтерфейсу користувача змодельований щоб вирішувати всі питання відображення даних та взаємодії з користувачами. Завдяки ньому обслуговуючий персонал взаємодіє з програмним забезпеченням.

Після аналізу безпеки було вирішено зашифровувати дані користувача для забезпечення збереження персональних даних у конфіденційності. В якості алгоритма шифрації був вибран AES-GCM як найнадійніший варіант. Для хешування паролей користувачів використовується bcrypt адже забезпечує мінімальну можливість знайдення паролю.

Таким чином було змодельоване програмне забезпечення для реалізації всіх функціональних та нефункціональних вимог.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						51
Змін.	Арк.	№ докум.	Підп.	Дата.		

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування програмного забезпечення це необхідна частина розробки, що допомагає виявити недоліки та некоректну роботу програми. Програмні тести допомагають розробнику по частинах перевірити правильну роботу програми та всього розроблюваного функціоналу і легше знаходити помилки та виправляти їх.

Тести використовуються для:

- виправлення помилок на етапі проектування ПЗ;
- випуску програмного продукту з мінімальної кількістю помилок в функціоналі;
- зменшення витрат на пошук недоліків у програмному забезпеченні;
- збільшення модульності коду, що призводить до кращого доповнення та змінення програмного забезпечення.

При тестуванні програмного забезпечення можна опиратися на автоматизоване та ручне тестування. При автоматизованому тестуванні тестувальник має ретельно налаштувати процес тестування та потім спеціальна програма автоматично виконає перевірку на помилки. Від початкових налаштувань будуть залежати результати тестування. Ручне тестування являє собою написанням розробником тест-кейсів і великим чином залежить від навичок тестування розробника. Хоча автоматизоване тестування швидше та дешевше, але іноді тестувальник при ручній перевірці може знайти помилки, що не знайшла би програма та виконати нестандартні тести.

Використовуваним видом тестування є функціональне тестування, при цьому тестуванні розробник тестує програму на коректне виконання всіх

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						52
Змін.	Арк.	№ докум.	Підп.	Дата.		

функціональних вимог. Цей вид допомагає розібратися чи правильно виконуються всі необхідні функції програми та виправити при необхідності.

Популярним видом тестування є модульне тестування, при модульному тестуванні розробники окремо перевіряє кожний метод програми та перевіряє правильність його функцій. Завдяки цьому можна легко знаходити помилки в функціонуванні маленьких частей та виправляти їх.

При проведенні процесу тестування програмного забезпечення треба ретельно перевірити взаємодію програми з базою даних, адже це важлива частина програмного забезпечення і при великій зв'язності сутностей може викликати багато помилок і вплинути на коректність даних.

Враховуючи складність та об'ємність розробки програмного забезпечення виникнення помилок є неминучим, тому виконання тестування це необхідна частина розробки програмного продукту.

3.2 Опис процесів тестування

Тестування програмного забезпечення відбувається на різних етапах розробки програмного продукту. Алгоритм тестування містить наступні етапи:

- аналіз вимог програмного забезпечення та визначення критеріїв оцінки;
- визначення підходів для комплексного тестування продукту;
- визначення необхідного результату тестування;
- розробка та написання тест кейсів за визначеними вимогами;
- проведення тестування;
- оцінка результатів тестування програми;
- виявлення дефектів програми;
- виправлення помилок програми;
- повторне проведення усіх кроків, починаючи з тестування, поки всі вимоги не будуть виконані.

Системні вимоги:

- операційною системою має бути 32 або 64 бітна Windows 10 або новіша або Unix система;
- оперативна пам'ять комп'ютера має бути не менше ніж 2Гб.

Тестування проводилось на персональному комп'ютері з процесором Core I7-6800K, з 6 ядрами та тактовою частотою 3.4 гігагерц та оперативною пам'яттю 16 гігабайт на операційній системі Windows 10. Базою даних слугувала MySql.

3.3 Опис контрольного прикладу

Протягом тестування програмного продукту було перевірено його на відповідність функціональним та нефункціональним вимогам, а також варіантам використання. Також було враховано правильність відображення інтерфейсу та обробку некоректних дій користувача. Результати тестування наведені в наступних таблицях:

- авторизація користувача з вірними даними через вікно інтерфейсу(таблиця 3.1);
- помилка авторизації користувача при вводі невірних даних через вікно інтерфейсу(таблиця 3.2);
- вихід авторизованого користувача через вікно інтерфейсу(таблиця 3.3);
- додавання клієнта за допомогою інтерфейсу(таблиця 3.4);
- редагування клієнта за допомогою інтерфейсу(таблиця 3.5);
- видалення клієнта за допомогою інтерфейсу(таблиця 3.6);
- додавання страв за допомогою інтерфейсу(таблиця 3.7);
- редагування страв за допомогою інтерфейсу(таблиця 3.8);
- видалення страв за допомогою інтерфейсу(таблиця 3.9);
- додавання меню за допомогою інтерфейсу(таблиця 3.10);
- редагування меню за допомогою інтерфейсу(таблиця 3.11);

- видалення меню за допомогою інтерфейсу(таблиця 3.12);
- додавання персоналу за допомогою інтерфейсу(таблиця 3.13);
- редагування персоналу за допомогою інтерфейсу(таблиця 3.14);
- видалення персоналу за допомогою інтерфейсу(таблиця 3.15);
- додавання замовлення за допомогою інтерфейсу(таблиця 3.16);
- додавання клієнтів та страв до замовлення за допомогою інтерфейсу(таблиця 3.17);
- видалення замовлення за допомогою інтерфейсу(таблиця 3.18);
- видалення з замовлення клієнтів та страв за допомогою інтерфейсу(таблиця 3.19);
- видалення з замовлення страв клієнта за допомогою інтерфейсу(таблиця 3.20);
- розрахунок замовлення за допомогою інтерфейсу(таблиця 3.21).

Таблиця 3.1 – Авторизація користувача з вірними даними через вікно інтерфейсу.

Мета тесту	Перевірити роботу можливість авторизації користувача через вікно інтерфейсу.
Початковий стан	MySQL сервер запущений. Програма запущена. Відображено сцену авторизації.
Вхідні дані	Логін, пароль.
Проведення тесту	Ввести існуючий у базі номер телефону у поле «Login», та відповідний йому пароль у поле «Password», натиснути кнопку авторизації.
Очікуваний результат	Програма переключиться на сцену для управління замовленнями, клієнтами, персоналом та меню. У правому верхньому куту виводиться привітання з ім'ям авторизованого користувача.

Кінець таблиці 3.1

Фактичний результат	Програма переключилася на сцену для управління замовленнями, клієнтами, персоналом та меню. У правому верхньому куту вивелося привітання з ім'ям авторизованого користувача.
---------------------	--

Таблиця 3.2 – Помилка авторизації користувача при вводі невірних даних через вікно інтерфейсу.

Мета тесту	Перевірити вивід помилки та відхилення авторизації при введенні невірних даних користувача.
Початковий стан	MySQL сервер запущений. Програма запущена. Відображено сцену авторизації.
Вхідні дані	Логін, пароль.
Проведення тесту	Ввести у існуючий у базі номер телефону у поле «Login», та невідповідний йому пароль у поле «Password», натиснути кнопку авторизації.
Очікуваний результат	Програма виведе вікно з повідомлення про невідповідність даних та після натискання кнопки «ОК» повернеться на вікно авторизації.
Фактичний результат	Програма вивела вікно з повідомленням про невідповідність даних та після натискання кнопки «ОК» повернеться на вікно авторизації.

Таблиця 3.3 – Вихід авторизованого користувача через вікно інтерфейсу.

Мета тесту	Перевірити вихід користувача з програми та повернення на форму авторизації.
Початковий стан	MySQL сервер запущений. Програма запущена. Користувач авторизований. Відображено сцену управління.
Вхідні дані	-

Кінець таблиці 3.1

Проведення тесту	Натиснути кнопку «Log out», у правому верхньому куту.
Очікуваний результат	Програма вийде з аккаунту поточного користувача та переключиться на форму авторизації.
Фактичний результат	Програма вийшла з аккаунту поточного користувача та переключиться на форму авторизації.

Таблиця 3.4 – Додавання клієнта за допомогою інтерфейсу.

Мета тесту	Перевірити правильне додавання клієнту до програми та вставлення у базу даних.
Початковий стан	Користувач авторизований. Відображено сцену управління.
Вхідні дані	Ім'я, фамілія, номер телефону, електронна пошта, дата народження, адреса.
Проведення тесту	Перейти на вкладку управління клієнтами та ввести дані у призначені для цього поля інтерфейсу та натиснути кнопку «Register».
Очікуваний результат	Програма додає клієнта, вставляє його до бази та виводить на екран оновлений список клієнтів.
Фактичний результат	Програма додала клієнта, вставила його до бази та вивела на сцену оновлений список клієнтів.

Таблиця 3.5 – Редагування клієнта за допомогою інтерфейсу.

Мета тесту	Перевірити редагування існуючого клієнту.
------------	---

Кінець таблиці 3.5

Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління клієнтами.
Вхідні дані	Ім'я, фамілія, номер телефону, електронна пошта, дата народження, адреса.
Проведення тесту	Вибрати потрібного клієнта зі списку, ввести оновлені дані у призначені для цього поля та натиснути кнопку «Edit».
Очікуваний результат	Програма знаходить клієнта та замінює його дані на введені у поля. Оновлює дані клієнта в базі даних та виводить оновлений список клієнтів до інтерфейсу.
Фактичний результат	Програма знайшла клієнта та змінила його дані на введені у поля. Правильно оновила дані клієнта в базі даних та вивела оновлений список клієнтів до інтерфейсу.

Таблиця 3.6 – Видалення клієнта за допомогою інтерфейсу.

Мета тесту	Перевірити видалення існуючого клієнту.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління клієнтами.
Вхідні дані	-
Проведення тесту	Вибрати потрібного клієнта зі списку, та натиснути кнопку «Delete».
Очікуваний результат	Програма знаходить користувача та його з програми. Видаляє клієнта в базі даних та виводить оновлений список клієнтів до інтерфейсу.
Фактичний результат	Програма знайшла користувача та його з програми. Видалила клієнта в базі даних та вивела оновлений список клієнтів до інтерфейсу.

Таблиця 3.7 – Додавання страв за допомогою інтерфейсу.

Мета тесту	Перевірити правильне додавання страв до програми, вставлення у базу даних та додавання страви до меню.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління меню.
Вхідні дані	Меню, назва, інгредієнти, опис, алергени, ціна, тип, категорія.
Проведення тесту	Вибрати меню, ввести дані у призначені для цього поля інтерфейсу та натиснути кнопку «Add Dish».
Очікуваний результат	Програма додає страву до меню, вставляє її до бази та зв'язує з меню. Виводить на екран оновлений список страв.
Фактичний результат	Програма додала страву до меню, вставила її до бази та зв'язала з меню, вивела на сцену оновлений список страв.

Таблиця 3.8 – Редагування страв за допомогою інтерфейсу.

Мета тесту	Перевірити коректне редагування існуючої страви в меню та базі даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління меню. Вибране меню.
Вхідні дані	Назва, інгредієнти, опис, алергени, ціна, тип, категорія.
Проведення тесту	Вибрати потрібну страву зі списку, ввести оновлені дані у, призначені для цього, поля та натиснути кнопку «Edit».
Очікуваний результат	Програма знаходить страву в меню та замінює її дані на введені у поля. Оновлює дані страви в базі даних та виводить оновлений список страв до інтерфейсу.

Кінець таблиці 3.8

Фактичний результат	Програма знайшла страву та замінила її дані на введені у поля. Оновила дані страви в базі даних та вивела оновлений список страв до інтерфейсу.
---------------------	---

Таблиця 3.9 – Видалення страв за допомогою інтерфейсу.

Мета тесту	Перевірити видалення існуючої страви з меню та бази даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління меню. Вибране меню.
Вхідні дані	-
Проведення тесту	Вибрати потрібну страву зі списку та натиснути кнопку «Delete».
Очікуваний результат	Програма знаходить страву в меню та видаляє її. Видаляє страву в базі даних та виводить оновлений список страв до інтерфейсу.
Фактичний результат	Програма знайшла страву в меню та видалила її. Видалила страву в базі даних та вивела оновлений список страв до інтерфейсу.

Таблиця 3.10 – Додавання меню за допомогою інтерфейсу.

Мета тесту	Перевірити правильне додавання меню до програми, вставлення у базу даних меню та зв'язування зі всіма стравами.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління меню.
Вхідні дані	Назва, опис, дата початку, дата кінця, тип.
Проведення тесту	Вибрати меню, ввести дані у, призначені для цього, поля інтерфейсу та натиснути кнопку «Add Menu».

Кінець таблиці 3.10

Очікуваний результат	Програма додає меню до списку, вставляє до бази. Виводить на екран оновлений список меню.
Фактичний результат	Програма додала меню до списку, вставила до бази та вивела на інтерфейс оновлений список меню.

Таблиця 3.11 – Редагування меню за допомогою інтерфейсу.

Мета тесту	Перевірити коректне редагування існуючого меню в програмі та базі даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління меню.
Вхідні дані	Назва, опис, дата початку, дата кінця, тип.
Проведення тесту	Вибрати меню зі списку, ввести оновлені дані у, призначені для цього, поля та натиснути кнопку «Edit Menu».
Очікуваний результат	Програма знаходить меню в списку та замінює дані на введені у поля. Оновлює дані меню в базі даних та виводить оновлений список меню до інтерфейсу.
Фактичний результат	Програма знайшла меню в списку та замінила дані на введені у поля. Оновила дані меню в базі даних та вивела оновлений список меню до інтерфейсу.

Таблиця 3.12 – Видалення меню за допомогою інтерфейсу.

Мета тесту	Перевірити коректне видалення існуючого меню та його страв в програмі та базі даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Відкрита вкладка управління меню.
Вхідні дані	-

Кінець таблиці 3.12

Проведення тесту	Вибрати меню зі списку, натиснути кнопку «Delete Menu».
Очікуваний результат	Програма знаходить меню в списку та видаляє його разом із всіма стравами. Видаляє меню в базі даних та зв'язки зі стравами, виводить оновлений список меню до інтерфейсу.
Фактичний результат	Програма знайшла меню в списку та видалила його разом із всіма стравами. Видалила меню в базі даних та зв'язки зі стравами, вивела оновлений список меню до інтерфейсу.

Таблиця 3.13 – Додавання персоналу за допомогою інтерфейсу.

Мета тесту	Перевірити правильне додавання персоналу до програми та вставлення у базу даних.
Початковий стан	Користувач авторизований. Відображено сцену управління.
Вхідні дані	Ім'я, фамілія, номер телефону, пароль, дата народження, позиція, зарплатня.
Проведення тесту	Перейти на вкладку управління персоналом та ввести дані у, призначені для цього, поля інтерфейсу та натиснути кнопку «Register».
Очікуваний результат	Програма додає персонал, вставляє його до бази та виводить на екран оновлений список персоналу.
Фактичний результат	Програма додала користувача, вставила його до бази та вивела на сцену оновлений список користувачів.

Таблиця 3.14 – Змінення персоналу за допомогою інтерфейсу.

Мета тесту	Перевірити правильне редагування персоналу в програми та зміні у базу даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління персоналом.
Вхідні дані	Ім'я, фамілія, номер телефону, пароль, дата народження, позиція, зарплатня.
Проведення тесту	Вибрати персонал зі списку, ввести змінені дані у, призначені для цього, поля інтерфейсу та натиснути кнопку «Edit».
Очікуваний результат	Програма змінює вибраний персонал, оновлює його дані в базі та виводить до інтерфейсу оновлений список персоналу.
Фактичний результат	Програма змінила вибраний персонал, оновла його дані в базі та вивела до інтерфейсу оновлений список персоналу.

Таблиця 3.15 – Видалення персоналу за допомогою інтерфейсу.

Мета тесту	Перевірити правильне видалення персоналу в програмі та в базі даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління персоналом.
Вхідні дані	-
Проведення тесту	Вибрати персонал зі списку та натиснути кнопку «Delete».
Очікуваний результат	Програма видаляє вибраний персонал, видаляє його дані в базі та виводить до інтерфейсу оновлений список персоналу.

Кінець таблиці 3.15

Фактичний результат	Програма видалила вибраний персонал, видалила його дані в базі та вивела до інтерфейсу оновлений список персоналу.
---------------------	--

Таблиця 3.16 – Додавання замовлення за допомогою інтерфейсу.

Мета тесту	Перевірити правильне додавання замовлення в програмі та в базі даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління замовленнями.
Вхідні дані	Номер столу, кількість клієнтів.
Проведення тесту	Ввести дані у, призначені для цього, поля інтерфейсу та натиснути кнопку «Save».
Очікуваний результат	Програма створює пусте замовлення з даними, введеними в поля, додає до списку активних замовлень. Оновлює список замовлень.
Фактичний результат	Програма створила пусте замовлення з даними, введеними в поля, додала до списку активних замовлень. Оновила список замовлень.

Таблиця 3.17 – Додавання клієнтів та страв до замовлення за допомогою інтерфейсу.

Мета тесту	Перевірити правильне додавання клієнтів та їх страв до замовлення.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління замовленнями.
Вхідні дані	-

Кінець таблиці 3.17

Проведення тесту	Включити режим додавання, натиснувши на конпку «Enable Addition Mode». Вибрати зі списку замовлення, для додавання клієнтів і страв. Вибрати клієнта зі списку та натиснути «Add Client», вибрати страву та натиснути «Add Dish».
Очікуваний результат	Програма перемикається у режим додавання. Додає обраного клієнта до замовлення, потім додає до обраного клієнта в замовленні вибрану страву. Оновлює список клієнтів та страв.
Фактичний результат	Програма перемикнулася у режим додавання. Додала обраного клієнта до замовлення, потім додала до обраного клієнта в замовленні вибрану страву. Оновила список клієнтів та страв.

Таблиця 3.18 – Зміна замовлення за допомогою інтерфейсу.

Мета тесту	Перевірити правильну зміну даних замовлення в програмі.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління замовленнями.
Вхідні дані	Номер столу, кількість клієнтів.
Проведення тесту	Вибрати замовлення зі списку, змінити дані у відповідних полях. Натиснути кнопку «Edit».
Очікуваний результат	Програма знаходить обране замовлення у списку, змінює його параметри на, введені у поля, дані. Оновлює список замовлень.

Кінець таблиці 3.18

Фактичний результат	Програма знайшла обране замовлення у списку, змінила його параметри на, введені у поля, дані. Оновила список замовлень.
---------------------	--

Таблиця 3.19 – Видалення з замовлення клієнтів та його страв за допомогою інтерфейсу.

Мета тесту	Перевірити правильне видалення клієнта та його страв з замовлення в програмі.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління замовленнями.
Вхідні дані	-
Проведення тесту	Вибрати замовлення зі списку. Вибрати клієнта зі списку клієнтів в замовленні. Натиснути кнопку «Delete Client».
Очікуваний результат	Програма знаходить обране замовлення у списку та клієнту у замовленні, видаляє його зі всіма стравами. Виводить на інтерфейс оновлений список клієнтів та страв.
Фактичний результат	Програма знайшла обране замовлення у списку та клієнта у замовленні, видалила його зі всіма стравами. Вивела на інтерфейс оновлений список клієнтів та страв.

Таблиця 3.20 – Видалення з замовлення страв клієнта за допомогою інтерфейсу.

Мета тесту	Перевірити правильну видалення клієнта та страв з замовлення в програмі.
------------	--

Кінець таблиці 3.20

Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління замовленнями.
Вхідні дані	-
Проведення тесту	Вибрати замовлення зі списку. Вибрати клієнта зі списку клієнтів та страву зі списку страв. Натиснути кнопку «Delete Dish».
Очікуваний результат	Програма знаходить обране замовлення у списку, клієнта у замовленні та його страву, видаляє обрану страву. Виводить на інтерфейс оновлений список страв клієнта.
Фактичний результат	Програма знайшла обране замовлення у списку, клієнта у замовленні та його страву, видалила обрану страву. Вивела на інтерфейс оновлений список страв клієнта.

Таблиця 3.21 – Розрахунок замовлення за допомогою інтерфейсу.

Мета тесту	Перевірити правильний розрахунок та розбиття замовлення, формування чеку та запис до бази даних.
Початковий стан	Користувач авторизований. Відображено сцену управління. Вибрана вкладка управління замовленнями.
Вхідні дані	Чайові, тип розбиття, спосіб оплати.
Проведення тесту	Вибрати замовлення зі списку замовлень. Натиснути кнопку «Checkout». На діалоговому вікні ввести кількість чайових, вибрати тип розбиття та тип оплати, натиснути кнопку «ОК».

Кінець таблиці 3.21

Очікуваний результат	Програма знаходить обране замовлення у списку, додає чайові. Проводить розбиття замовлення на чеки за обраними параметрам, формує чеки та записує замовлення до бази, зв'язує з клієнтами і їхніми стравами.
Фактичний результат	Програма знайшла обране замовлення у списку, додала чайові. Провела розбиття замовлення на чеки за обраними параметрам, сформувала чеки та записала замовлення до бази, зв'язала з клієнтами і їхніми стравами.

Висновки по розділу

В цьому розділі було проведено аналіз та оцінка якості програмного забезпечення. Було виявлено, що найважливішою частиною програми є коректне виконання функціональних вимог. При виникненні помилки може статися невідповідність даних, що призведе складного процесу усунення недоліків, тому функціональні вимоги є обов'язковими для тестування.

У цьому розділі було досліджено критерії оцінки розроблюваного програмного продукту, вимоги до тестування, а також, визначений необхідний результат. Був описаний повний алгоритм тестування програми. Були сформульовані апаратні та програмні вимоги для проведення етапу тестування.

Також були написані тест-кейси під всі функціональні вимоги та с задоволенням усіх критеріїв оцінки програмного забезпечення і були вказані у виді таблиць. Таблиці містять: вимогу тест-кейсу, початковий стан, вхідні дані, етапи проведення тесту, очікуваний результат та фактичний. Тест-кейси допомагають зрозуміти важливі етапи роботи програмного продукту та проаналізувати етапи тестування для повної перевірки функціоналу.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розгортання програмного забезпечення може відбуватися на комп'ютері з будь-якою операційною системою (Windows, MacOS, Linux) з підтримкою Java версії 11 або новіших версій. Для розгортання необхідною умовою є встановлена реляційна база даних MySQL або MariaDB версії 10.4.17 та новіше. Перед використанням програмного забезпечення слід перечитати довідку користувача для кращого розуміння роботи з застосунком та його повного функціоналу. Запуск програми відбувається відкриттям файлу Restaurant.jar. Після виконання всіх дій програма готова до використання.

4.2 Робота з програмним забезпеченням

Робота з програмним забезпеченням відбувається внаслідок взаємодії користувача з графічним інтерфейсом. Дії користувача відстежуються завдяки натисканню кнопок, а дані передаються через текстові поля та відсилаються на обробку. При запуску програми користувач потрапляє на сцену авторизації, яку потрібно пройти щоб отримати доступ до основного функціоналу. При успішній авторизації користувач перенаправляється на сцену управління, яка містить функціонал обробки сутностей. На цій сцені користувач може вибрати сторінки управління замовленнями, клієнтами, меню та персоналом. Кожна сторінка містить текстове поле пошуку, що виведе весь перелік відповідної сутності при залишанні її пустою, в іншому випадку виконає пошук по вказаному параметру. Також вони містять списки для відображення сутностей, текстові поля для вводу параметрів та кнопки додавання, редагування і видалення.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						69
Змін.	Арк.	№ докум.	Підп.	Дата.		

Сторінка управління клієнтами надає можливість реєструвати клієнтів завдяки введенню даних у текстові поля та натискання кнопки реєстрації. При виборі клієнту зі списку його дані виводяться у текстові поля для зручної зміни, при натисканні на кнопку змінити дані користувача будуть змінені. При виборі клієнту зі списку та натисканні кнопки видалення клієнта буде видалено.

Сторінка меню дозволяє вводити параметри меню та додавати його до програми при натисканні кнопки додавання. При виборі меню зі списку його дані виводяться у текстові поля для редагування і після натискання змінити, параметри будуть змінені. Також після вибору замовлення є можливість ввести інформацію про страву у текстові поля та додати до меню кнопкою. Вибрав меню зі списку всі його страви будуть виведені у список на сцені, також можна видалити його натисканням на кнопку видалення. Обрав страву зі списку можна змінити її параметри у текстових полях та зберегти зміни кнопкою зміни або видалити кнопкою видалення.

Сторінка управління персоналом дає можливість введення даних персоналу та реєстрації, відведеною на це, кнопкою. При виборі персоналу зі списку дані будуть виведені у текстові поля для зручного редагування і збереження змін кнопкою зміни даних. Обраний персонал можна видалити з програми кнопкою видалення.

Сторінка управління замовленнями дозволяє додавати замовлення з номером столу та кількістю клієнтів. Далі замовлення буде в активному стані і після ввімкнення режиму додавання і вибору замовлення в списку, на сцену виведеться список клієнтів та страв для додавання. Після обрання клієнту та натискання кнопки додавання клієнту до замовлення буде його додано. До обраного клієнта можна додати страву, вибравши її зі списку та натиснувши додати страву. Після вимкнення режиму додавання за таким самим алгоритмом можна видалити страву клієнта та самого клієнта кнопкою видалення. Обравши замовлення та натиснувши кнопку для розрахування буде відкрито діалогову сцену, де користувач може ввести чайові та обрати варіант

розбиття чеку та спосіб оплати. Після підтвердження введених даних замовлення розбивається та друкується чек, а саме замовлення з усіма зв'язками додається до бази даних.

Висновки по розділу

Даний розділ описує розгортання створеного програмного продукту. Описуються необхідні системні та програмні вимоги для функціонування програмного забезпечення. Також був наведений алгоритм роботи з програмою та детальний опис всіх можливостей взаємодії з нею для створення замовлень та обробки інших сутностей програми.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
						71
Змін.	Арк.	№ докум.	Підп.	Дата.		

ВИСНОВКИ

В результаті виконання дипломного проекту було пройдено повний життєвий цикл розробки додатку для прийому та обробки замовлень в ресторанному бізнесі. Програмне забезпечення відповідає всім функціональним вимогам та має допоміжний функціонал для полегшення виконання основного призначення.

Тема актуальна через автоматизацію бізнес процесів та потреби оптимізувати та пришвидшити обробки замовлень персоналом для підвищення продуктивності бізнесу.

В розділі «Аналізу вимог до ПЗ» були наведені вимоги до створюваного програмного забезпечення та опис і аналіз предметної області, а також аналіз конкурентів і відомих проектів. Було вивчено підходи до створення програми для обробки замовлень у ресторанах як: POS додатки, OMS. Досліджені різні рішення: OSPOS, WallacePOS, SambaPOS, що дають можливість прийому та обробки замовлень. А також проаналізовані відомі програмні продукти після чого було зроблено висновки, що всі продукти мають нагромаджений функціонал та складний інтерфейс.

В розділі були визначені варіанти використання програмного продукту. На основі них були визначені функціональні вимоги програми, що визначають головне призначення та нефункціональні вимоги, що описують потреби застосунку. Були визначені головні вимоги до модулів програми.

В результаті аналізу вимог було складено чітке розуміння мети розробки, етапи та вимоги до програмного забезпечення для повного функціонування.

В розділі «Моделювання та конструювання програмного забезпечення» було проаналізовано та змодельовано процеси, які користувач має пройти для прийому, обробки, редагування замовлень та маніпуляції с іншими потрібними сутностями.

Наступним кроком було розроблено архітектуру програмного продукту та розділення її на функціональні модулі, до яких увійшли: модуль сутностей, модуль роботи з базою даних, модуль сервісів та модуль інтерфейсу. Для розробки модулів була обрана мова Java версії 11 бо вона дозволяє легко перенести програму на пристрої з іншими операційними системами і тим самим розширити спектр використання програми. Також було використано високотехнологічний фреймворк Spring для поліпшення програми та реалізації інверсії залежностей. Для інтерфесу використовувалася сучасна технологія JavaFX з легкою побудовою інтерфейсу, а у якості бази даних використано MySQL через безкоштовність та наявності всього потрібного функціоналу.

Модуль сутностей відповідає за всі сутності якими оперує програма. Він виконує більш модельну роль ніж функціональну.

Модуль роботи з базою даних призначений для проведення операцій вставки, зміни, діставання та видалення всіх присутніх сутностей. Також він займається хешуванням паролів і шифруванням персональних даних користувачів та клієнтів.

Модуль сервісів призначений для відтворення логіки програми. Він оброблює дані, відправляє їх на відображення до інтерфейсу та взаємодіє з модулем бази даних. Також він зберігає об'єкти сутностей і виконує над ними необхідних операції, визначені в функціональних вимогах: додавання, зміну, видалення. На рівні з цим він виконує операції пошуку сутностей, авторизації для входу персоналу до системи, видалення страв або клієнтів зі стравами з замовлення і розрахунок замовлення. Однією з функцій також є розбиття замовлення на різні чеки по клієнтах та їх друк.

Модуль інтерфейсу відповідає за графічний шар програми. Цей модуль напряду взаємодіє з користувачем та оброблює його запити. Даний модуль відображає дані, отримані з модулю сервісів, передає введені користувачем дані до модулю сервісів, надсилає запити, а також переключає сцени.

Останнім кроком розділу було проаналізовано вимоги до безпеки даних. Через те що розроблене програмне забезпечення потребує зберігання даних персоналу та клієнтів, було вирішено шифрувати вразливу інформацію завдяки алгоритму AES-GCM. Паролі було вирішено хешувати з сіллю алгоритмом SHA3-256. Такі методи допоможуть зберегти персональні дані про несанкційованому доступі.

В розділі «Аналіз якості та конструювання програмного забезпечення» було виведено необхідні вимоги та критерії до якості програмного забезпечення, адже низька якість програми можуть негативно сказатися на її експлуатації.

Також в розділі було розроблено алгоритм тестування програмного продукту для виявлення всіх помилок і складався з: виявлення критеріїв оцінки, вибору підходу до тестування, визначення необхідних результатів, написання тест кейсів, проведення тестування, виправлення помилок та повтор тестування при виявленні недоліків. Було визначенні програмні та апаратні вимоги для проведення тестування. Ці всі кроки виконувалися для повного та комплексного тестування програми та виявлення всіх помилок в функціонуванні, для подальшого їх виправлення.

Останнім кроком тестування програмного продукту було написання тест-кейсів під всі функціональні вимоги та з виконанням критеріїв оцінки. Було перевірено кожний варіант використання програми користувачем та виправлено помилки для її коректної роботи. Застосунок був протестований на операційній системі Windows 10 та MacOS. Після виправлення помилок розроблений програмний продукт остаточно пройшов перевірку на якість, не було виявлено помилок впливаючих на функціонал.

В розділі «Впровадження та супровід програмного забезпечення» було визначено необхідні вимоги та кроки розгортання створеного програмного продукту на цільовій платформі. Було детально та поетапно описано роботу користувача з програмним забезпеченням та всі можливі функції та способи їх використання.

Перспективою програмного продукту є створення додаткових функцій: відстеження приготування, збір та аналіз даних, впровадження клієнт-серверної архітектури для взаємодії декількох дистанційних пристроїв з програмою, додавання можливостей онлайн бронювання та замовлення.

Розроблене програмне забезпечення націлене на використання в ресторанному бізнесі, тому

Таким чином було змодельовано, розроблено та перевірено на якість програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі. Програма відповідає всім сучасним технологічним нормам розробки забезпечення, таким як: SOLID принципи, простота інтерфейсу, безпека даних, ін'єкція залежностей, якість. Розроблений додаток повністю підходить для прийому та обробки замовлень в ресторанному бізнесі.

					КПІ.IT-8223.045430.02.81 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		75

ПЕРЕЛІК ПОСИЛАНЬ

1. Як використовувати фреймворк Spring <https://spring.io/guides>
2. Різні компоненти інтерфейсу JavaFX та способи використання <https://jenkov.com/tutorials/javafx/index.html>
3. Порівняння популярних POS систем <https://www.quicksprout.com/best-point-of-sales-systems-pos/>
4. Системи для обробки замовлень <https://www.nchannel.com/blog/top-5-order-management-systems/>
5. Використання Spring Application Context <https://www.baeldung.com/spring-application-context>
6. Описання POS систем https://en.wikipedia.org/wiki/Point_of_sale
7. ДСТУ з оформлення технічної документації https://science.kname.edu.ua/images/dok/derzhstandart_3008_2015.pdf
8. Запроси для взаємодії з БД https://www.w3schools.com/mysql/mysql_sql.asp
9. Open-source pos системи <https://opensource.com/tools/point-of-sale>
10. Допомога в вирішенні помилок <https://stackoverflow.com/>

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2022 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРИЙОМУ ТА ОБРОБКИ
ЗАМОЛВЕНЬ В РЕСТОРАННОМУ БІЗНЕСІ**

Опис програми

КПІ.IT-8223.045430.03.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Александер СЕЛЮК

Київ – 2022

ТЕКСТ ПРОГРАМНОГО КОДУ

Тексти програмного коду

Програмне забезпечення для прийому та обробки замовлень в ресторанному
бізнесі

(Найменування програми (документа))

Flash Drive

(Вид носія даних)

47 арк, 100,7 Мб

(Обсяг програми, арк., Мб)

Київ – 2022

					КПІ.ІТ-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		2

```

@ComponentScan()
public class Main extends Application {
    private Stage stage;
    ApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfig.class);
    private LoginController lcon;

    @Override
    public void start(Stage primaryStage) {
        try {
            stage = primaryStage;
            stage.setTitle("Order Processing App");
            initRootLayout();
            primaryStage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void initRootLayout() {
        FXMLLoader loader = new FXMLLoader();

        loader.setLocation(LoginController.class.getResource("loginScene.fxml"));
        try {
            stage.setScene(loader.load());
            lcon = loader.getController();
            lcon.setMain(this);
            lcon.setStaffService(context.getBean(StaffService.class));

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void initManagerLayout() {
        FXMLLoader loader = new FXMLLoader();

        loader.setLocation(ManagerController.class.getResource("managerScene.fxml"));
        try {
            stage.setScene(loader.load());
            ManagerController mCon = loader.getController();
            mCon.setManager(context.getBean(OrderService.class));
            mCon.setService(context.getBean(ClientService.class));
            mCon.setStaffService(context.getBean(StaffService.class));
            mCon.setMenuService(context.getBean(MenuService.class));
            mCon.setStaff(lcon.getStaff());
            mCon.setMain(this);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}

public class LoginController implements Initializable {
    private StaffService staffService;
    private Main main;
    private Staff staff;

    @FXML
    private TextField loginField;
    @FXML
    private TextField passwordField;

    @Override

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3


```

public void initialize(URL location, ResourceBundle resources) {

}

@FXML
private void loginButtonAction() {
    staff = staffService.login(loginField.getText(),
passwordField.getText());
    if(staff != null){
        main.initManagerLayout();
    }else {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Login error");
        alert.setHeaderText("Your credential are incorrect");
        alert.setContentText("Please repeat login");
        alert.showAndWait();
    }
    System.out.println("Button pressed");
}

public void setMain(Main main) {
    this.main = main;
}

public void setStaffService(StaffService staffService) {
    this.staffService = staffService;
}

public Staff getStaff() {
    return staff;
}

}

```

					КП.ІТ-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

```

public class ManagerController implements Initializable {
    private OrderService orderManager;
    private ClientService clientService;
    private StaffService staffService;
    private MenuService menuService;
    private Main main;
    private Staff staff;
    private Order orderInProcessing;

    public ManagerController() {
    }

    public void setMain(Main main) {
        this.main = main;
    }

    public void setService(ClientService service) {
        this.clientService = service;
    }

    public void setManager(OrderService orderService) {
        this.orderManager = orderService;
    }

    public void setStaffService(StaffService staffService) {
        this.staffService = staffService;
    }

    public void setMenuService(MenuService menuService) {
        this.menuService = menuService;
    }

    public void setStaff(Staff staff) {
        this.staff = staff;
        welcomeLabel.setText("Welcome, " + staff.getName());
        if (!staff.getPosition().equals("administrator")) {
            staffTab.setDisable(true);
            menuTab.setDisable(true);
        }
    }

    @FXML
    private Tab staffTab;
    @FXML
    private Tab menuTab;
    @FXML
    private Label welcomeLabel;
    @FXML
    private TabPane tabPane;
    @FXML
    private ListView<Order> orderList;
    @FXML
    private ListView<Client> clientOrderList;
    @FXML
    private ListView<Dish> dishOrderList;
    @FXML
    private TextField searchField;
    @FXML
    private Button searchButton;
    @FXML
    private TextField tableNumberField;
    @FXML
    private TextField numberOfClientsField;
    @FXML
    private DatePicker orderDateField;
    @FXML
    private ChoiceBox<String> clientBox;
    @FXML
    private Button addDishToClient;
    @FXML

```

```

private Button saveOrder;
@FXML
private Button editOrder;
@FXML
private Button checkout;
@FXML
private Button additionModeButton;

@FXML
private ListView<Client> clientList;
@FXML
private TextField clientSearch;
@FXML
private Button clientSearchButton;
@FXML
private TextField clientName;
@FXML
private TextField clientSurname;
@FXML
private TextField clientEmail;
@FXML
private TextField clientPhoneNumber;
@FXML
private TextField clientAddress;
@FXML
private DatePicker clientBirthDate;
@FXML
private DatePicker clientRegistrationDate;
@FXML
private Button registerClient;
@FXML
private Button editClient;
@FXML
private Button deleteClient;
@FXML
private Button addClientToOrder;
@FXML
private Button addDishToOrder;

@FXML
private ListView<Menu> menuList;
@FXML
private ListView<Dish> dishList;
@FXML
private TextField searchMenu;
@FXML
private TextField menuName;
@FXML
private TextArea menuDescription;
@FXML
private TextField menuType;
@FXML
private CheckBox activeMenu;
@FXML
private Button searchMenuButton;
@FXML
private DatePicker menuStartDate;
@FXML
private DatePicker menuEndDate;
@FXML
private Button addMenuButton;
@FXML
private Button editMenuButton;
@FXML
private Button deleteMenuButton;
@FXML
private TextField dishName;
@FXML
private TextField dishIngridients;
@FXML
private TextField dishDescription;

```

					КП.ІТ-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		2

```

@FXML
private TextField dishAlergens;
@FXML
private TextField dishPrice;
@FXML
private TextField dishType;
@FXML
private TextField dishCategory;
@FXML
private Button dishAddButton;
@FXML
private Button dishEditButton;
@FXML
private Button dishDeleteButton;

@FXML
private ListView<Staff> staffList;
@FXML
private TextField searchStaff;
@FXML
private TextField staffName;
@FXML
private TextField staffSurname;
@FXML
private TextField staffEmail;
@FXML
private TextField staffPhoneNumber;
@FXML
private TextField staffPassword;
@FXML
private TextField staffSalary;
@FXML
private TextField staffPosition;
@FXML
private DatePicker staffBirthDate;
@FXML
private Button addStaffButton;
@FXML
private Button editStaffButton;
@FXML
private Button deleteStaffButton;
@FXML
private Button searchStaffButton;

@Override
public void initialize(URL location, ResourceBundle resources) {

}

private int localId = 0;
private boolean additionMode;

@FXML
public void searchOrders() {
    updateOrderList(orderManager.getHistory());
    System.out.println("search pressed");
}

public void updateOrderList(List<Order> orders) {
    orderList.getItems().clear();
    orderList.getItems().addAll(orders);
}

@FXML
public void logOut() {
    main.initRootLayout();
}

@FXML
public void additionModeToggle() {
    additionMode = !additionMode;
}

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

```

        if (additionMode) {

            additionModeButton.setText("Disable additon mode");
        } else {
            additionModeButton.setText("Enable additon mode");
        }
    }

@FXML
public void saveOrderAction() {
    try {
        int tabNum = Integer.parseInt(tableNumberField.getText());
        int numOfCli =
Integer.parseInt(numberOfClientsField.getText());
        if (tabNum == 0 || numOfCli == 0) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Save error");
            alert.setHeaderText("Not all neccesary fields are
filled");

            alert.setContentText("Please input table number and
number of clients");

            alert.showAndWait();
        } else {
            LocalDateTime timeOrder = LocalDateTime.now();
            orderInProcessing = new OrderImpl(++localId, tabNum,
numOfCli, timeOrder);

            orderInProcessing.setStaff(staff);
            orderManager.addOrder(orderInProcessing);
            updateOrderList(orderManager.getAll());
        }
    } catch (NumberFormatException e) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Format error");
        alert.setHeaderText("Field contain wrong format");
        alert.setContentText("Please input only numbers");
        alert.showAndWait();
    }
}

@FXML
public void addClientToOrder() {
    int id = orderInProcessing.getId();
    orderManager.addClients(id,
clientOrderList.getSelectionModel().getSelectedItem());
}

@FXML
public void addDishesToOrder() {
    orderManager.addDishToClientOrder(orderInProcessing.getId(),

clientOrderList.getSelectionModel().getSelectedItem().getId(),

dishOrderList.getSelectionModel().getSelectedItem());
}

@FXML
public void deleteClientFromOrder() {
    int id = orderInProcessing.getId();
    orderManager.deleteClientFromOrder(id,
clientOrderList.getSelectionModel().getSelectedItem().getId());
    clientOrderUpdate();
    dishOrderList.getItems().clear();
}

@FXML
public void deleteDishesFromOrder() {
    orderManager.deleteDishFromClientOrder(orderInProcessing.getId(),

```

					КП.ІТ-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

```

        clientOrderList.getSelectionModel().getSelectedItem().getId(),
        dishOrderList.getSelectionModel().getSelectedItem());

        dishOrderListUpdate(orderManager.getDishesByClientAndOrder(orderInProcessing.g
        etId(),
        clientOrderList.getSelectionModel().getSelectedItem()));
    }

    @FXML
    public void chooseOrderAction() {
        Order order = orderList.getSelectionModel().getSelectedItem();
        orderInProcessing = order;
        if (!additionMode) {
            if (orderManager.findOrder(order.getId()) != null) {

                clientOrderListUpdate(orderManager.getClientsByOrder(order.getId()));

                dishOrderListUpdate(orderManager.getDishesByActiveOrder(order.getId()));

                numberOfClientsField.setText(String.valueOf(order.getNumberOfClients()));

                tableNumberField.setText(String.valueOf(order.getTableNumber()));
            } else {
                Order orderH =
                orderManager.findOrderHistory(order.getId());
                clientOrderListUpdate(new
                ArrayList<>(orderH.getDishesByClient().keySet()));

                dishOrderListUpdate(orderManager.getDishesByOrderHistory(order.getId()));
            }
        } else {
            clientOrderListUpdate(clientService.retrieveClients());
            dishOrderListUpdate(menuService.getAllDishes());
        }
    }

    public void dishOrderUpdate() {
        dishOrderList.getItems().clear();

        dishOrderList.getItems().addAll(orderManager.getDishesByActiveOrder(orderInPro
        cessing.getId()));
    }

    public void clientOrderUpdate() {
        clientOrderList.getItems().clear();

        clientOrderList.getItems().addAll(orderManager.getClientsByOrder(orderInProces
        sing.getId()));
    }

    public void dishOrderListUpdate(List<Dish> dishes) {
        dishOrderList.getItems().clear();
        dishOrderList.getItems().addAll(dishes);
    }

    public void clientOrderListUpdate(List<Client> clients) {
        clientOrderList.getItems().clear();
        clientOrderList.getItems().addAll(clients);
    }

    @FXML
    public void chooseClientOrderAction() {
        if (!additionMode) {

            dishOrderListUpdate(orderManager.getDishesByClientAndOrder(orderInProcessing.g
            etId(),
            clientOrderList.getSelectionModel().getSelectedItem()));
        }
    }

```

					КП.ІТ-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

```

    }

@FXML
public void chooseDishOrderAction() {

@FXML
public void editOrderAction() {
    try {
        int tabNum = Integer.parseInt(tableNumberField.getText());
        int numOfCli =
Integer.parseInt(numberOfClientsField.getText());
        if (tabNum == 0 || numOfCli == 0) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Edit error");
            alert.setHeaderText("Not all neccesary fields are
filled");

            alert.setContentText("Please input table number and
number of clients");

            alert.showAndWait();
        } else {
            Order order = new OrderImpl(tabNum, numOfCli);

            orderManager.changeOrder(orderList.getSelectionModel().getSelectedItem().getId
(), order);

            updateOrderList(orderManager.getAll());
        }
    } catch (NumberFormatException e) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Edit error");
        alert.setHeaderText("Salary must be an integer");
        alert.setContentText("Please input numbers and no symbols");
        alert.showAndWait();
    }
}

@FXML
public void checkoutOrderAction() {
    try {
        FXMLLoader loader = new FXMLLoader();

        loader.setLocation(DialogController.class.getResource("dialog.fxml"));
        DialogPane checkoutDialogPane = loader.load();
        DialogController dialogController = loader.getController();
        Dialog<String> dialog = new Dialog();
        dialog.setDialogPane(checkoutDialogPane);
        dialog.setTitle("Checkout dialog");
        dialog.showAndWait();
        int tips = dialogController.getTips();
        List<List<Integer>> splittings = new ArrayList<>();
        if (dialogController.getSplitting().equals("Whole")) {
            List<Integer> customers = new ArrayList<>();
            for (Client client :
orderManager.findOrder(orderInProcessing.getId()).getDishesByClient().keySet()) {
                customers.add(client.getId());
            }
            splittings.add(customers);
        } else if (dialogController.getSplitting().equals("50/50")) {
            List<Integer> customers1 = new ArrayList<>();
            List<Integer> customers2 = new ArrayList<>();
            int num = 1;
            for (Client client :
orderManager.findOrder(orderInProcessing.getId()).getDishesByClient().keySet()) {
                if (num <=
orderInProcessing.getNumberOfClients() / 2) {
                    customers1.add(client.getId());
                } else {
                    customers2.add(client.getId());
                }
            }
        }
    }
}

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

```

        num++;
    }
    splittings.add(customers1);
    splittings.add(customers2);
} else {
    for (Client client :
orderManager.findOrder(orderInProcessing.getId()).getDishesByClient().keySet()) {
        List<Integer> customers = new ArrayList<>();
        customers.add(client.getId());
        splittings.add(customers);
    }
}

    }
    orderManager.checkOut(orderInProcessing.getId(), tips,
splittings);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

@FXML
public void clientSearchAction() {
    Client client = null;
    String text = clientSearch.getText();
    if (text.contains("@")) {
        client = clientService.findByMail(text);
    } else {
        client = clientService.findByPhone(text);
    }
    System.out.println(clientSearch.getText());
    if (client != null) {
        showClientList(List.of(client));
    } else {
        showClients();
    }
}

public void showClientList(List<Client> clients) {
    clientList.getItems().clear();
    clientList.getItems().addAll(clients);
}

public void showClients() {
    showClientList(clientService.retrieveClients());
}

@FXML
public void clientListChooseAction() {
    Client client = clientList.getSelectionModel().getSelectedItem();
    System.out.println(client);
    clientName.setText(client.getName());
    clientSurname.setText(client.getSurname());
    clientEmail.setText(client.getEmail());
    clientPhoneNumber.setText(client.getPhoneNumber());
    clientAddress.setText(client.getAddress());
    clientBirthDate.setValue(client.getBirthDate());

    clientRegistrationDate.setValue(client.getRegisterDate().toLocalDate());
    // showClients();
}

@FXML
public void clientRegisterAction() {
    LocalDate birthDate = clientBirthDate.getValue();
    if (clientName.getText().isBlank() ||
clientSurname.getText().isBlank() || clientPhoneNumber.getText().isBlank()
        || birthDate == null) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Registration error");
        alert.setHeaderText("Not all neccessary fields are filled");
    }
}

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7


```

        alert.setContentText("Please input name, surname and phone
number");
        alert.showAndWait();
    } else {
        Client client = new ClientImpl(clientName.getText(),
clientSurname.getText(), clientEmail.getText(),
                                clientPhoneNumber.getText(), birthDate,
clientAddress.getText(), LocalDateTime.now());
        clientService.registerClient(client);
        showClients();
    }
}

@FXML
public void clientEditAction() {
    Client client = clientList.getSelectionModel().getSelectedItem();
    LocalDate birthDate = clientBirthDate.getValue();
    String name = clientName.getText().isBlank() ? client.getName() :
clientName.getText();
    String surname = clientSurname.getText().isBlank() ?
client.getSurname() : clientSurname.getText();
    String phone = clientPhoneNumber.getText().isBlank() ?
client.getPhoneNumber() : clientPhoneNumber.getText();

    Client clientedit = new ClientImpl(name, surname,
clientEmail.getText(), phone, birthDate,
                                clientAddress.getText());
    clientService.editClientData(client.getId(), clientedit);
    showClients();
}

@FXML
public void clientDeleteAction() {
    Client client = clientList.getSelectionModel().getSelectedItem();
    clientService.deleteClient(client);
    showClients();
}

public void showStaffList(List<Staff> staffs) {
    staffList.getItems().clear();
    staffList.getItems().addAll(staffs);
}

public void showStaff() {
    showStaffList(staffService.getStaff());
}

@FXML
public void staffListChooseAction() {
    Staff staff = staffList.getSelectionModel().getSelectedItem();
    System.out.println(staff);
    staffName.setText(staff.getName());
    staffSurname.setText(staff.getSurname());
    staffSalary.setText(String.valueOf(staff.getSalary()));
    staffPhoneNumber.setText(staff.getPhoneNumber());
    staffPosition.setText(staff.getPosition());
    staffBirthDate.setValue(staff.getBirthDate());
    staffPassword.clear();
    // showClients();
}

@FXML
public void staffAddAction() {
    try {
        LocalDate birthDate = staffBirthDate.getValue();
        if (staffName.getText().isBlank() ||
staffSurname.getText().isBlank() ||
                                staffPhoneNumber.getText().isBlank() ||
birthDate == null || staffPassword.getText().isBlank()) {
            Alert alert = new Alert(AlertType.INFORMATION);

```

					КП.ІТ-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		8

```

        alert.setTitle("Registration error");
        alert.setHeaderText("Not all necessary fields are
filled");

        alert.setContentText("Please input name, surname,
phone number and password");
        alert.showAndWait();
    } else {
        Staff staff = new StaffImpl(staffName.getText(),
staffSurname.getText(), staffPassword.getText(),

        Integer.parseInt(staffSalary.getText()), staffPhoneNumber.getText(),
staffBirthDate.getValue(),

                                staffPosition.getText());
        staffService.addStaff(staff);
        showStaff();
    }
} catch (NumberFormatException e) {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Edit error");
    alert.setHeaderText("Salary must be an integer");
    alert.setContentText("Please use correct format for salary");
    alert.showAndWait();
}

}

@FXML
public void staffEditAction() {
    try {
        Staff staffChosen =
staffList.getSelectionModel().getSelectedItem();
        if (staffName.getText().isBlank() ||
staffSurname.getText().isBlank()

                                || staffPhoneNumber.getText().isBlank() ||
staffBirthDate.getValue() == null

                                || staffPassword.getText().isBlank()) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Edit error");
            alert.setHeaderText("Not all necessary fields are
filled");

            alert.setContentText("Please input name, surname,
phone number and password");
            alert.showAndWait();
        } else {
            Staff staff = new StaffImpl(staffName.getText(),
staffSurname.getText(), staffPassword.getText(),

            Integer.parseInt(staffSalary.getText()), staffPhoneNumber.getText(),
staffBirthDate.getValue(),

                                staffPosition.getText());
            staff.setId(staffChosen.getId());
            staffService.editStaff(staffChosen.getId(), staff);
            showStaff();
        }
    } catch (NumberFormatException e) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Edit error");
        alert.setHeaderText("Salary must be an integer");
        alert.setContentText("Please use correct format for salary");
        alert.showAndWait();
    }
}

@FXML
public void staffDeleteAction() {
    Staff staff = staffList.getSelectionModel().getSelectedItem();
    staffService.deleteStaff(staff);
    showStaff();
}

@FXML
public void staffSearchAction() {

```

```

        String text = searchStaff.getText();
        Staff staff;
        if (text.matches("[0-9]+")) {
            System.out.println("phone");
            staff = staffService.findStaffByPhone(text);
        } else {
            staff = staffService.findStaffBySurname(text);
        }
        System.out.println(text);
        if (staff != null) {
            showStaffList(List.of(staff));
        } else {
            showStaff();
        }
    }

    public void showMenuList(List<Menu> menus) {
        menuList.getItems().clear();
        menuList.getItems().addAll(menus);
    }

    public void showMenus() {
        showMenuList(menuService.getAllMenus());
    }

    @FXML
    public void menuSearchAction() {
        Menu menu = menuService.findMenu(searchMenu.getText());
        if (menu != null) {
            showMenuList(List.of(menu));
        } else {
            showMenus();
        }
    }

    @FXML
    public void addMenuAction() {
        if (menuName.getText().isBlank()) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Addition error");
            alert.setHeaderText("Not all neccesary fields are filled");
            alert.setContentText("Please input name");
            alert.showAndWait();
        } else {
            Menu menu = new MenuImpl(menuName.getText(),
            menuDescription.getText(), menuType.getText(),
            activeMenu.isSelected(),
            menuStartDate.getValue(), menuEndDate.getValue());
            menuService.addMenu(menu);
            showMenus();
        }
    }

    @FXML
    public void editMenuAction() {
        Menu menu = menuList.getSelectionModel().getSelectedItem();
        if (menuName.getText().isBlank()) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Edit error");
            alert.setHeaderText("Not all neccesary fields are filled");
            alert.setContentText("Please input name");
            alert.showAndWait();
        } else {
            Menu editedMenu = new MenuImpl(menuName.getText(),
            menuDescription.getText(), menuType.getText(),
            activeMenu.isSelected(),
            menuStartDate.getValue(), menuEndDate.getValue());
            editedMenu.setId(menu.getId());
            menuService.editMenu(menu.getId(), editedMenu);
        }
        showMenus();
    }

```

					КПІ.IT-8223.045430.03.13	Арк.
						10
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

    }

@FXML
public void deleteMenuAction() {
    Menu menu = menuList.getSelectionModel().getSelectedItem();
    menuService.removeMenu(menu.getId());
    showMenus();
}

@FXML
public void chooseMenuAction() {
    Menu menu = menuList.getSelectionModel().getSelectedItem();
    menuName.setText(menu.getName());
    menuDescription.setText(menu.getDescription());
    menuType.setText(menu.getType());
    activeMenu.setSelected(menu.getActive());
    menuStartDate.setValue(menu.getStartDate());
    menuEndDate.setValue(menu.getEndDate());
    showDishesList(menu.getDishes());
}

@FXML
public void addDishAction() {
    try {
        Menu menu = menuList.getSelectionModel().getSelectedItem();
        if (dishName.getText().isBlank()) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Addition error");
            alert.setHeaderText("Not all necessary fields are
filled");

            alert.setContentText("Please input name");
            alert.showAndWait();
        } else {
            Dish dish = new DishImpl(dishName.getText(),
dishIngridients.getText(), dishDescription.getText(),
dishAlergens.getText(),
Integer.parseInt(dishPrice.getText()), dishType.getText(),
dishCategory.getText());
            menuService.addDishToMenu(menu.getId(), dish);

            showDishesList(menuService.findMenu(menu.getId()).getDishes());
        }
    } catch (NumberFormatException e) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Add error");
        alert.setHeaderText("Price must be an integer");
        alert.setContentText("Please use correct format for price");
        alert.showAndWait();
    }
}

@FXML
public void editDishAction() {
    try {
        Dish dish = dishList.getSelectionModel().getSelectedItem();
        if (dishName.getText().isBlank()) {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Addition error");
            alert.setHeaderText("Not all necessary fields are
filled");

            alert.setContentText("Please input name");
            alert.showAndWait();
        } else {
            Dish dishedit = new DishImpl(dishName.getText(),
dishIngridients.getText(), dishDescription.getText(),
dishAlergens.getText(),
Integer.parseInt(dishPrice.getText()), dishType.getText(),
dishCategory.getText());
            dishedit.setId(dish.getId());
            menuService.editDish(dishedit);
            showDishesList(

```

```

        menuService.findMenu(menuList.getSelectionModel().getSelectedItem().getId()).getDishes();
    }
    } catch (NumberFormatException e) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Edit error");
        alert.setHeaderText("Price must be an integer");
        alert.setContentText("Please use correct format for price");
        alert.showAndWait();
    }
}

@FXML
public void deleteDishAction() {
    Dish dish = dishList.getSelectionModel().getSelectedItem();
    menuService.removeDish(dish.getId());

    showDishesList(menuService.findMenu(menuList.getSelectionModel().getSelectedItem().getId()).getDishes());
}

@FXML
public void chooseDishAction() {
    Menu menu = menuList.getSelectionModel().getSelectedItem();
    Dish dish = dishList.getSelectionModel().getSelectedItem();
    if (dish != null) {
        dishName.setText(dish.getName());
        dishDescription.setText(dish.getDescription());
        dishType.setText(dish.getType());
        dishPrice.setText(String.valueOf(dish.getPrice()));
        dishCategory.setText(dish.getCategory());
        dishIngridients.setText(dish.getIngridients());
        dishAlergens.setText(dish.getAlergens());
    } else {
        dishName.clear();
        dishDescription.clear();
        dishType.clear();
        dishPrice.clear();
        dishCategory.clear();
        dishIngridients.clear();
        dishAlergens.clear();
    }
}

public void showDishes() {
    showDishesList(menuService.getAllDishes());
}

public void showDishesList(List<Dish> dishes) {
    dishList.getItems().clear();
    dishList.getItems().addAll(dishes);
}

}

public class ClientImpl implements Client {
    private int id;
    private String name;
    private String surname;
    private String email;
    private String phoneNumber;
    private LocalDate birthDate;
    private String address;
    private LocalDateTime registerDate;

    public ClientImpl(int id, String name, String surname, String email, String
phoneNumber, LocalDate birthDate, String address,
                    LocalDateTime registerDate) {

```

					КПІ.IT-8223.045430.03.13	Арк.
						12
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        super();
        this.id = id;
        this.name = name;
        this.surname = surname;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
        this.address = address;
        this.registerDate = registerDate;
    }

    public ClientImpl(String name, String surname, String email, String
phoneNumber, LocalDate birthDate, String address,
        LocalDateTime registerDate) {
        super();
        this.name = name;
        this.surname = surname;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
        this.address = address;
        this.registerDate = registerDate;
    }

    public ClientImpl(String name, String surname, String email, String
phoneNumber, LocalDate birthDate, String address) {
        super();
        this.name = name;
        this.surname = surname;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
        this.address = address;
    }

    public ClientImpl() {
    }

    public ClientImpl(int clientNumber) {
        super();
        this.id = clientNumber;
        this.name = String.valueOf(clientNumber);
        this.surname = String.valueOf(clientNumber);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    @JsonIgnore

```

```

public LocalDate getBirthDate() {
    return birthDate;
}
public void setBithDate(LocalDate bithDate) {
    this.birthDate = bithDate;
}
public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}
@JsonIgnore
public LocalDateTime getRegisterDate() {
    return registerDate;
}
public void setRegisterDate(LocalDateTime registerDate) {
    this.registerDate = registerDate;
}

public String getEmail() {
    return email;
}
@JsonGetter("birthDate")
public String getBirthDateString() {
    return birthDate.toString();
}
@JsonGetter("registerDate")
public String getRegisterDateString() {
    return registerDate.toString();
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((email == null) ? 0 : email.hashCode());
    result = prime * result + ((name == null) ? 0 : name.hashCode());
    result = prime * result + ((phoneNumber == null) ? 0 :
phoneNumber.hashCode());
    result = prime * result + ((surname == null) ? 0 :
surname.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    ClientImpl other = (ClientImpl) obj;
    if (email == null) {
        if (other.email != null)
            return false;
    } else if (!email.equals(other.email))
        return false;
    if (name == null) {
        if (other.name != null)
            return false;
    } else if (!name.equals(other.name))
        return false;
    if (phoneNumber == null) {
        if (other.phoneNumber != null)
            return false;
    } else if (!phoneNumber.equals(other.phoneNumber))
        return false;
    if (surname == null) {
        if (other.surname != null)

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		14

```

        return false;
    } else if (!surname.equals(other.surname))
        return false;
    return true;
}

@Override
public String toString() {
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy");
    return "Client:" + " name: " + name + ", surname: " + surname + ",
email: " + email
        + ", phoneNumber: " + phoneNumber + ", birthDate: "
+ birthDate.format(format) + ", address: " + address
        + ", registerDate: " + registerDate + ";";
}

@Override
public void setBirthDate(LocalDate birthDate) {
    this.birthDate = birthDate;
}

@Override
public void setEmail(String email) {
    this.email = email;
}
}

public class DishImpl implements Dish {
    private int id;
    private String name;
    private String ingredients;
    private String description;
    private String alergens;
    private int price;
    private String type;
    private String category;

    public DishImpl(int id, String name, String ingredients, String description,
String alergens, int price,
        String type, String category) {
        super();
        this.id = id;
        this.name = name;
        this.ingredients = ingredients;
        this.description = description;
        this.alergens = alergens;
        this.price = price;
        this.type = type;
        this.category = category;
    }

    public DishImpl(String name, String ingredients, String description, String
alergens, int price,
        String type, String category) {
        this.name = name;
        this.ingredients = ingredients;
        this.description = description;
        this.alergens = alergens;
        this.price = price;
        this.type = type;
        this.category = category;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {

```

					КПІ.IT-8223.045430.03.13	Арк.
						15
Змін.	Арк.	№ докум.	Підп.	Дата.		


```

        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getIngridients() {
        return ingridients;
    }
    public void setIngridients(String ingridients) {
        this.ingridients = ingridients;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getAlergens() {
        return alergens;
    }
    public void setAlergens(String alergens) {
        this.alergens = alergens;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }

    @Override
    public String toString() {
        return "Dish: " + "name: " + name + ", ingridients: " + ingridients +
            ", description: " + description
            + ", alergens: " + alergens + ", price: " + price +
            ", type: " + type + ", category: " + category + ";";
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((alergens == null) ? 0 :
alergens.hashCode());
        result = prime * result + ((category == null) ? 0 :
category.hashCode());
        result = prime * result + ((description == null) ? 0 :
description.hashCode());
        result = prime * result + ((ingridients == null) ? 0 :
ingridients.hashCode());
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        result = prime * result + price;
        result = prime * result + ((type == null) ? 0 : type.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		16

```

        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    DishImpl other = (DishImpl) obj;
    if (alergens == null) {
        if (other.alergens != null)
            return false;
    } else if (!alergens.equals(other.alergens))
        return false;
    if (category == null) {
        if (other.category != null)
            return false;
    } else if (!category.equals(other.category))
        return false;
    if (description == null) {
        if (other.description != null)
            return false;
    } else if (!description.equals(other.description))
        return false;
    if (ingridients == null) {
        if (other.ingridients != null)
            return false;
    } else if (!ingridients.equals(other.ingridients))
        return false;
    if (name == null) {
        if (other.name != null)
            return false;
    } else if (!name.equals(other.name))
        return false;
    if (price != other.price)
        return false;
    if (type == null) {
        if (other.type != null)
            return false;
    } else if (!type.equals(other.type))
        return false;
    return true;
}

}

public class MenuImpl implements Menu {
    private int id;
    private String name;
    private String description;
    private String type;
    private boolean active;
    private LocalDate startDate;
    private LocalDate endDate;
    private List<Dish> dishes;

    public MenuImpl(int id, String name, String description, String type, boolean
active, LocalDate startDate,
        LocalDate endDate) {
        this.id = id;
        this.name = name;
        this.description = description;
        this.type = type;
        this.active = active;
        this.startDate = startDate;
        this.endDate = endDate;
        this.dishes = new ArrayList<Dish>();
    }

    public MenuImpl(String name, String description, String type, boolean active,
LocalDate startDate,

```

					КПІ.IT-8223.045430.03.13	Арк.
						17
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        LocalDate endDate) {
            this.name = name;
            this.description = description;
            this.type = type;
            this.active = active;
            this.startDate = startDate;
            this.endDate = endDate;
            this.dishes = new ArrayList<Dish>();
        }

        public int getId() {
            return id;
        }
        public void setId(int id) {
            this.id = id;
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getDescription() {
            return description;
        }
        public void setDescription(String description) {
            this.description = description;
        }
        public String getType() {
            return type;
        }
        public void setType(String type) {
            this.type = type;
        }
        public boolean getActive() {
            return active;
        }
        public void setActive(boolean active) {
            this.active = active;
        }
        public LocalDate getStartDate() {
            return startDate;
        }
        public void setStartDate(LocalDate startDate) {
            this.startDate = startDate;
        }
        public LocalDate getEndDate() {
            return endDate;
        }
        public void setEndDate(LocalDate endDate) {
            this.endDate = endDate;
        }

        public List<Dish> getDishes(){
            return dishes;
        }

        @Override
        public void setDishes(List<Dish> dishes) {
            this.dishes = dishes;
        }

        @Override
        public int hashCode() {
            final int prime = 31;
            int result = 1;
            result = prime * result + ((description == null) ? 0 :
description.hashCode());
            result = prime * result + ((endDate == null) ? 0 :
endDate.hashCode());
            result = prime * result + ((name == null) ? 0 : name.hashCode());

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		18

```

        result = prime * result + ((startDate == null) ? 0 :
startDate.hashCode());
        result = prime * result + ((type == null) ? 0 : type.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        MenuImpl other = (MenuImpl) obj;
        if (description == null) {
            if (other.description != null)
                return false;
        } else if (!description.equals(other.description))
            return false;
        if (endDate == null) {
            if (other.endDate != null)
                return false;
        } else if (!endDate.equals(other.endDate))
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        if (startDate == null) {
            if (other.startDate != null)
                return false;
        } else if (!startDate.equals(other.startDate))
            return false;
        if (type == null) {
            if (other.type != null)
                return false;
        } else if (!type.equals(other.type))
            return false;
        return true;
    }

    @Override
    public String toString() {
        DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        return "Menu: " + "name: " + name + ", description: " + description +
", type: " + type + ", active: "
+ active + ", startDate: " +
startDate.format(format) + ", endDate: " + endDate.format(format) + ";";
    }

}

public class StaffImpl implements Staff{
    private int id;
    private String name;
    private String surname;
    private String password;
    private int salary;
    private String phoneNumber;
    private LocalDate birthDate;
    private String position;

    public StaffImpl(int id, String name, String surname, int salary, String
phoneNumber, LocalDate birthDate,
String position) {
        super();
        this.id = id;
        this.name = name;

```

					КПІ.IT-8223.045430.03.13	Арк.
						19
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        this.surname = surname;
        this.salary = salary;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
        this.position = position;
    }

    public StaffImpl(int id, String name, String surname, String password, int
salary, String phoneNumber, LocalDate birthDate,
        String position) {
        super();
        this.id = id;
        this.name = name;
        this.surname = surname;
        this.salary = salary;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
        this.position = position;
        this.password = password;
    }

    public StaffImpl(String name, String surname, String password, int salary,
String phoneNumber, LocalDate birthDate,
        String position) {
        this.name = name;
        this.surname = surname;
        this.salary = salary;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
        this.position = position;
        this.password = password;
    }

    @Override
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Override
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    @Override
    public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    @Override
    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

```

					КПІ.IT-8223.045430.03.13	Арк.
						20
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

    }

    @Override
    public LocalDate getBirthDate() {
        return birthDate;
    }

    public void setBirthDate(LocalDate birthDate) {
        this.birthDate = birthDate;
    }

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    @Override
    public void setPassword(String password) {
        this.password = password;
    }

    }

    @Override
    public String getPassword() {
        return password;
    }

    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((birthDate == null) ? 0 :
        birthDate.hashCode());
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        result = prime * result + ((phoneNumber == null) ? 0 :
        phoneNumber.hashCode());
        result = prime * result + ((position == null) ? 0 :
        position.hashCode());
        result = prime * result + salary;
        result = prime * result + ((surname == null) ? 0 :
        surname.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        StaffImpl other = (StaffImpl) obj;
        if (birthDate == null) {
            if (other.birthDate != null)
                return false;
        } else if (!birthDate.equals(other.birthDate))
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        if (phoneNumber == null) {
            if (other.phoneNumber != null)
                return false;
        } else if (!phoneNumber.equals(other.phoneNumber))
            return false;
        if (position == null) {
            if (other.position != null)

```

```

        return false;
    } else if (!position.equals(other.position))
        return false;
    if (salary != other.salary)
        return false;
    if (surname == null) {
        if (other.surname != null)
            return false;
    } else if (!surname.equals(other.surname))
        return false;
    return true;
}

@Override
public String toString() {
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy");
    return "Staff: " + "name: " + name + ", surname: " + surname + ",
password: " + password
        + ", salary: " + salary + ", phoneNumber: " +
phoneNumber + ", birthDate: " + birthDate.format(format) + ", position: "
        + position + ";";
}

}

@Component
public class AESHandler implements EncryptionHandler{

    public AESHandler() {

    }

    private static final String KEY_LOCATION = "C:/Users/alexs/eclipse-
workspace/RestaurantClient/resources/notakey.conf";
    private static final int BLOCK_SIZE = 128;

    @Override
    public String encrypt(String data, byte[] iv) {
        String encrypted = null;
        byte[] keyBytes = decryptKeyKMS(takeKeyOut());
        encrypted = encryptAES(data, getKey(keyBytes), iv);
        return encrypted;
    }

    @Override
    public String decrypt(String data, byte[] iv) {
        String decrypted = null;
        byte[] keyBytes = decryptKeyKMS(takeKeyOut());
        decrypted = decryptAES(data, getKey(keyBytes), iv);
        return decrypted;
    }

    public byte[] decryptKeyKMS(byte[] baseKey) {
        byte[] encKey = Base64.getDecoder().decode(baseKey);
        return encKey;
    }

    public byte[] takeKeyOut() {
        byte[] key = null;
        try {
            key = Files.readAllBytes(Path.of(KEY_LOCATION));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return key;
    }

    @Override
    public SecretKey getKey(byte[] key) {

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		22

```

        return new SecretKeySpec(key, 0, key.length, "AES");
    }

    @Override
    public SecretKey generateKey(int length) throws NoSuchAlgorithmException {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(length);
        SecretKey key = keyGenerator.generateKey();
        return key;
    }

    @Override
    public byte[] generateVector() {
        return generateBytes(BLOCK_SIZE);
    }

    @Override
    public byte[] generateBytes(int n) {
        SecureRandom random = new SecureRandom();
        byte[] bytes = new byte[n];
        random.nextBytes(bytes);
        return bytes;
    }

    private String encryptAES(String data, SecretKey key, byte[] iv) {
        try {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            GCMParameterSpec gcmParameterSpec = new GCMParameterSpec(16 *
8, iv);

            cipher.init(Cipher.ENCRYPT_MODE, key, gcmParameterSpec);
            byte[] ciphered = cipher.doFinal(data.getBytes());
            return Base64.getEncoder().encodeToString(ciphered);
        } catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException
                | IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        } catch (InvalidAlgorithmParameterException e) {
            e.printStackTrace();
        }
        return null;
    }

    private String decryptAES(String data, SecretKey key, byte[] iv) {
        try {
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            GCMParameterSpec gcmParameterSpec = new GCMParameterSpec(16 *
8, iv);

            cipher.init(Cipher.DECRYPT_MODE, key, gcmParameterSpec);
            byte[] deciphered =
cipher.doFinal(Base64.getDecoder().decode(data));
            return new String(deciphered);
        } catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException
                | IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        } catch (InvalidAlgorithmParameterException e) {
            e.printStackTrace();
        }
        return null;
    }

    @Component
    public class Sha3256Hasher implements HashHandler{

        @Override

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		23


```

        public String encrypt(String text, byte[] salt) {
            MessageDigest md = null;
            String result = null;
            if (text != null) {
                try {
                    md = MessageDigest.getInstance("SHA3-256");
                } catch (NoSuchAlgorithmException e) {
                    e.printStackTrace();
                }
                if (md != null) {
                    md.update(salt);
                    md.update(text.getBytes(StandardCharsets.UTF_8));
                    result = String.format("%064x", new BigInteger(1,
md.digest())));
                }
            }
            return result;
        }

        public byte[] generateSalt() {
            SecureRandom random = new SecureRandom();
            byte[] salt = new byte[32];
            random.nextBytes(salt);
            return salt;
        }
    }

    @Component
    public class Sha3256Hasher implements HashHandler{

        @Override
        public String encrypt(String text, byte[] salt) {
            MessageDigest md = null;
            String result = null;
            if (text != null) {
                try {
                    md = MessageDigest.getInstance("SHA3-256");
                } catch (NoSuchAlgorithmException e) {
                    e.printStackTrace();
                }
                if (md != null) {
                    md.update(salt);
                    md.update(text.getBytes(StandardCharsets.UTF_8));
                    result = String.format("%064x", new BigInteger(1,
md.digest())));
                }
            }
            return result;
        }

        public byte[] generateSalt() {
            SecureRandom random = new SecureRandom();
            byte[] salt = new byte[32];
            random.nextBytes(salt);
            return salt;
        }
    }

    @Component
    public class StaffDAO implements DAO<Staff> {
        private static final String SELECT_STAFF = "SELECT * FROM staff WHERE id=?";
        private static final String SELECT_ALL_STAFF = "SELECT * FROM staff";
        private static final String LOGIN_STAFF = "SELECT * FROM staff WHERE
phone_number = ? AND password = ?";
        private static final String SELECT_STAFF_IV = "SELECT iv FROM staff WHERE
id=?";
        private static final String SELECT_STAFF_SALT = "SELECT salt FROM staff WHERE
phone_number=?";
    }

```

```

        private static final String SELECT_STAFF_IV_BY_PHONE = "SELECT iv FROM staff
WHERE phone_number=?";
        private static final String DELETE_STAFF = "DELETE FROM staff WHERE id=?";
        private static final String INSERT_STAFF = "INSERT INTO staff (name, surname,
password, salt, phone_number, birth_date, salary, position, iv) values(?, ?, ?, ?, ?,
?, ?, ?, ?)";
        private static final String UPDATE_STAFF = "UPDATE staff SET name = ?, surname
= ?, password = ?, salt = ?, phone_number = ?, birth_date = ?, salary = ?, position =
? WHERE id = ?";
        private static final String UPDATE_ORDERS = "UPDATE orders SET staff_id = 1
WHERE staff_id = ?";
        private static final String UPDATE_STAFF_NO_PASS = "UPDATE staff SET name = ?,
surname = ?, phone_number = ?, birth_date = ?, salary = ?, position = ? WHERE id = ?";
        private static final Logger LOGGER =
Logger.getLogger(StaffDAO.class.getName());

        private final JdbcTemplate jdbcTemplate;
        private final EncryptionHandler encryptor;
        private final HashHandler hasher;

        public StaffDAO(JdbcTemplate jdbcTemplate, EncryptionHandler ecnryptor,
HashHandler hasher) {
            this.jdbcTemplate = jdbcTemplate;
            this.encryptor = ecnryptor;
            this.hasher = hasher;
        }

        @Override
        public void add(Staff t) {
            LOGGER.log(Level.INFO, "Starting Staff insert");
            byte[] salt = hasher.generateSalt();
            String saltHash = Base64.getEncoder().encodeToString(salt);
            String hashedPass = hasher.encrypt(t.getPassword(), salt);
            byte[] iv = encryptor.generateVector();
            DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd");
            jdbcTemplate.update(INSERT_STAFF, encryptor.encrypt(t.getName(), iv),
encryptor.encrypt(t.getSurname(), iv), hashedPass, saltHash, t.getPhoneNumber(),
            encryptor.encrypt(t.getBirthDate().format(form),
iv), encryptor.encrypt(String.valueOf(t.getSalary()), iv),
encryptor.encrypt(t.getPosition(), iv), Base64.getEncoder().encodeToString(iv));
            LOGGER.log(Level.INFO, "Successful Staff insert");
        }

        @Override
        public void delete(Staff t) {
            delete(t.getId());
        }

        @Override
        public void delete(int id) {
            LOGGER.log(Level.INFO, "Starting Staff deletion");
            jdbcTemplate.update(UPDATE_ORDERS, id);
            jdbcTemplate.update(DELETE_STAFF, id);
            LOGGER.log(Level.INFO, "Successful Staff deletion");
        }

        @Override
        public Staff getById(int id) {
            LOGGER.log(Level.INFO, "Starting Staff extraction");
            byte[] iv = getIv(id);
            Staff staff = jdbcTemplate.query(SELECT_STAFF, new
PreparedStatementSetter() {

                @Override
                public void setValues(PreparedStatement ps) throws
SQLException {

                    ps.setInt(1, id);

                }

            }

```

```

        }, new ResultSetExtractor<Staff>() {

            @Override
            public Staff extractData(ResultSet rs) throws SQLException,
DataAccessException {
                return new StaffImpl(rs.getInt("id"),
encryptor.decrypt(rs.getString("name"), iv),

                encryptor.decrypt(rs.getString("surname"), iv),

                Integer.parseInt(encryptor.decrypt(String.valueOf(rs.getInt("salary")), iv)),
rs.getString("phone_number"),
                LocalDate.parse(encryptor.decrypt(rs.getString("birth_date"), iv)),

                encryptor.decrypt(rs.getString("position"), iv));
            }

        });
        LOGGER.log(Level.INFO, "Successful Staff extraction");
        return staff;
    }

    public byte[] getIv(int id) {
        System.out.println(id);
        return Base64.getDecoder().decode(jdbcTemplate.query(SELECT_STAFF_IV,
new PreparedStatementSetter() {

            @Override
            public void setValues(PreparedStatement ps) throws
SQLException {
                ps.setInt(1, id);
            }

        }, new ResultSetExtractor<String>() {

            @Override
            public String extractData(ResultSet rs) throws SQLException,
DataAccessException {
                rs.next();
                return rs.getString(1);
            }

        }

        ));
    }

    public String getIv(String phone) {
        return jdbcTemplate.query(SELECT_STAFF_IV_BY_PHONE, new
PreparedStatementSetter() {

            @Override
            public void setValues(PreparedStatement ps) throws
SQLException {
                ps.setString(1, phone);
            }

        }, new ResultSetExtractor<String>() {

            @Override
            public String extractData(ResultSet rs) throws SQLException,
DataAccessException {
                String result = null;
                if(rs.next()) {
                    result = rs.getString("iv");
                }
                return result;
            }

        }

        ));
    }
}

```

```

@Override
public List<Staff> getAll() {
    LOGGER.log(Level.INFO, "Starting all staff extraction");
    List<Staff> staff = jdbcTemplate.query(SELECT_ALL_STAFF, new
StaffRowMapper(encryptor));
    LOGGER.log(Level.INFO, "Successful all staff extraction");
    return staff;
}

@Override
public void update(Staff t) {
    LOGGER.log(Level.INFO, "Starting staff update");
    byte[] iv = getIv(t.getId());
    DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    System.out.println(t.getPassword());
    if(t.getPassword() != null) {
        byte[] salt = hasher.generateSalt();
        String saltHash = Base64.getEncoder().encodeToString(salt);
        String hashedPass = hasher.encrypt(t.getPassword(), salt);
        jdbcTemplate.update(UPDATE_STAFF, encryptor.encrypt(t.getName(), iv),
encryptor.encrypt(t.getSurname(), iv), hashedPass, saltHash, t.getPhoneNumber(),
encryptor.encrypt(t.getBirthDate().format(form),
iv), encryptor.encrypt(String.valueOf(t.getSalary()), iv),
encryptor.encrypt(t.getPosition(), iv), t.getId());
    } else {
        jdbcTemplate.update(UPDATE_STAFF_NO_PASS,
encryptor.encrypt(t.getName(), iv), encryptor.encrypt(t.getSurname(), iv),
t.getPhoneNumber(),

encryptor.encrypt(t.getBirthDate().format(form), iv),
encryptor.encrypt(String.valueOf(t.getSalary()), iv),
encryptor.encrypt(t.getPosition(), iv), t.getId());
    }
    LOGGER.log(Level.INFO, "Successful staff update");
}

@Override
public Staff login(String phone, String password) {
    Staff staff = null;
    String ivS = getIv(phone);
    byte[] iv;
    if(ivS != null) {
        iv = Base64.getDecoder().decode(ivS);
    } else {
        return null;
    }
    byte[] salt =
Base64.getDecoder().decode(jdbcTemplate.query(SELECT_STAFF_SALT, new
PreparedStatementSetter() {

        @Override
        public void setValues(PreparedStatement ps) throws
SQLException {

            ps.setString(1, phone);

        }

    }, new ResultSetExtractor<String>() {

        @Override
        public String extractData(ResultSet rs) throws SQLException,
DataAccessException {

            rs.next();
            return rs.getString("salt");

        }

    }));
    staff = jdbcTemplate.query(LOGIN_STAFF, new
PreparedStatementSetter() {

        @Override

```

```

        public void setValues(PreparedStatement ps) throws
SQLException {
            ps.setString(1, phone);
            ps.setString(2, hasher.encrypt(password, salt));
        }
    }, new ResultSetExtractor<Staff>() {

        @Override
        public Staff extractData(ResultSet rs) throws SQLException,
DataAccessException {
            Staff staff = null;
            if(rs.next()) {
                staff = new StaffImpl(rs.getInt("id"),
encryptor.decrypt(rs.getString("name"), iv),

                encryptor.decrypt(rs.getString("surname"), iv),

                Integer.parseInt(encryptor.decrypt(rs.getString("salary"), iv)),
rs.getString("phone_number"),
LocalDate.parse(encryptor.decrypt(rs.getString("birth_date"), iv)),

                encryptor.decrypt(rs.getString("position"), iv));
            }
            return staff;
        }

    });
    System.out.println(hasher.encrypt(password, salt));
    return staff;
}

}

```

```

@Component
public class OrderDAO implements DAO<Order> {
    private static final String SELECT_ORDER = "SELECT * FROM orders WHERE id =
?";

    private static final String SELECT_ALL_ORDERS = "SELECT * FROM orders ";
    private static final String SELECT_STAFF_ORDER = "SELECT st.* FROM orders JOIN
staff st on orders.staff_id = st.id WHERE orders.id = ?";
    private static final String DELETE_ORDER = "DELETE * FROM orders WHERE id=?";
    private static final String INSERT_ORDER = "INSERT INTO orders (table_number,
number_of_clients, order_date, staff_id) values(?, ?, ?, ?)";
    private static final String INSERT_CLIENT_TO_ORDER = "INSERT INTO
clients_to_orders (order_id, client_id, dish_id) values(?, ?, ?)";
    private static final String SELECT_CLIENT_BY_ORDER = "SELECT DISTINCT c.* FROM
clients c JOIN clients_to_orders cto ON c.id = cto.client_id JOIN orders o ON
cto.order_id = o.id WHERE o.id = ?";
    private static final String SELECT_ORDER_BY_CLIENT = "SELECT DISTINCT o.* FROM
clients c JOIN clients_to_orders cto ON c.id = cto.client_id JOIN orders o ON
cto.order_id = o.id WHERE c.id = ?";
    private static final String SELECT_DIDHES_BY_CLIENT = "SELECT d.* FROM clients
c JOIN clients_to_orders cto ON c.id = cto.client_id JOIN orders o ON cto.order_id =
o.id JOIN dishes d ON d.id = cto.dish_id WHERE c.id = ?";
    private static final String SELECT_DISHES_BY_CLIENT_IN_ORDER = "SELECT d.*
FROM clients c JOIN clients_to_orders cto ON c.id = cto.client_id JOIN orders o ON
cto.order_id = o.id JOIN dishes d ON d.id = cto.dish_id WHERE o.id = ? AND c.id = ?";
    private static final String UPDATE_ORDER = "UPDATE orders SET table_number =
?, number_of_clients = ? WHERE id = ?";
    private static final String DELETE_CLIENT_TO_ORDER = "DELETE * FROM
clients_to_orders WHERE order_id = ? AND client_id = ?";
    private static final String UPDATE_CLIENT_TO_ORDER = "UPDATE clients_to_orders
SET dish_id = ? WHERE order_id = ? AND client_id = ?";
    private static final String SELECT_ID = "SELECT MAX(id) FROM orders";
    private static final Logger LOGGER =
Logger.getLogger(OrderDAO.class.getName());
}

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		28

```

private final JdbcTemplate jdbcTemplate;

//@Autowired
public OrderDAO(final JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
}

@Override
public void add(Order t) {
    LOGGER.log(Level.INFO, "Starting order insert");
    jdbcTemplate.update(INSERT_ORDER, t.getTableNumber(),
t.getNumberOfClients(), t.getOrderDate().toString(),
t.getStaff().getId());
    LOGGER.log(Level.INFO, "Starting order clients dishes insert");
    addDishesByClient(t);
    LOGGER.log(Level.INFO, "Successful order insert");
}

public void addDishesByClient(Order t) {
    int id = jdbcTemplate.query(SELECT_ID, new
ResultSetExtractor<Integer>() {

        @Override
        public Integer extractData(ResultSet rs) throws SQLException,
DataAccessException {
            rs.next();
            return rs.getInt(1);
        }
    });
    System.out.println(id);
    for (Map.Entry<Client, List<Dish>> clientToDishes :
t.getDishesByClient().entrySet()) {
        for (Dish dish : clientToDishes.getValue()) {
            jdbcTemplate.update(INSERT_CLIENT_TO_ORDER, id,
clientToDishes.getKey().getId(), dish.getId());
        }
    }
}

@Override
public void delete(Order t) {
    // TODO Auto-generated method stub
}

@Override
public void delete(int id) {
    // TODO Auto-generated method stub
}

@Override
public Order getById(int id) {
    LOGGER.log(Level.INFO, "Starting order extraction");
    Order order = jdbcTemplate.queryForObject(SELECT_ORDER, new
OrderRowMapper(), id);
    if (order != null) {
        order.setStaff(selectStaffById(id));
        order.setDishesByClient(selectDishByClientById(id));
    }
    LOGGER.log(Level.INFO, "Successful order extraction");
    return null;
}

public Staff selectStaffById(int id) {
    LOGGER.log(Level.INFO, "Starting staff from order extraction");
    List<Staff> staff = jdbcTemplate.query(SELECT_STAFF_ORDER, new
StaffRowMapper(new AESHandler()), id);
    LOGGER.log(Level.INFO, "Successful staff from order extraction");
}

```

```

        return staff.get(0);
    }

    public Map<Client, List<Dish>> selectDishByClientById(int id) {
        LOGGER.log(Level.INFO, "Starting dish by client in order
extraction");
        Map<Client, List<Dish>> dishesByClient = new HashMap<>();
        List<Client> clientsByOrder =
jdbcTemplate.query(SELECT_CLIENT_BY_ORDER, new PreparedStatementSetter() {

            @Override
            public void setValues(PreparedStatement ps) throws
SQLException {

                ps.setInt(1, id);

            }
        }, new ClientRowMapper(new AESHandler()));
        for (Client client : clientsByOrder) {
            List<Dish> dishesByClients =
jdbcTemplate.query(SELECT_DISHES_BY_CLIENT_IN_ORDER,
                    new PreparedStatementSetter() {

                        @Override
                        public void
setValues(PreparedStatement ps) throws SQLException {

                            ps.setInt(1, id);
                            ps.setInt(2,
client.getId());

                        }
                    }, new DishRowMapper());
            dishesByClient.put(client, dishesByClients);
        }
        LOGGER.log(Level.INFO, "Successful dish by client in order
extraction");
        return dishesByClient;
    }

    @Override
    public List<Order> getAll() {
        LOGGER.log(Level.INFO, "Starting all orders extraction");
        List<Order> orders = jdbcTemplate.query(SELECT_ALL_ORDERS, new
OrderRowMapper());
        for (Order order : orders) {
            order.setStaff(selectStaffById(order.getId()));

            order.setDishesByClient(selectDishByClientById(order.getId()));
        }
        LOGGER.log(Level.INFO, "Successful all orders extraction");
        return orders;
    }

    @Override
    public void update(Order t) {
        LOGGER.log(Level.INFO, "Starting Order update");
        jdbcTemplate.update(UPDATE_ORDER, t.getTableNumber(),
t.getNumberOfClients(), t.getId());
        for (Map.Entry<Client, List<Dish>> entry :
t.getDishesByClient().entrySet()) {
            jdbcTemplate.update(DELETE_CLIENT_TO_ORDER,
t.getId(), entry.getKey().getId());
        }
        addDishesByClient(t);
        LOGGER.log(Level.INFO, "Successful Order update");
    }

    @Override
    public Order login(String login, String password) {
        // TODO Auto-generated method stub
        return null;
    }

```

```

}

@Component
public class MenuDAO implements DAO<Menu> {
    private static final String SELECT_ALL_MENUS = "SELECT * FROM menus";
    private static final String SELECT_MENU_BY_ID = "SELECT * FROM menus WHERE id
= ?";
    private static final String DELETE_MENU = "DELETE FROM menus WHERE id = ?";
    private static final String DISH_BY_MENU = "SELECT d.* FROM menus m JOIN
dishes_to_menus dtm on dtm.menu_id = m.id JOIN dishes d ON dtm.dish_id = d.id WHERE
m.id = ?";
    private static final String DELETE_DISH_TO_MENU = "DELETE FROM dishes_to_menus
WHERE menu_id=?";
    private static final String INSERT_MENU = "INSERT INTO menus (name,
description, type, active, start_date, end_date) values(?, ?, ?, ?, ?, ?)";
    private static final String INSERT_DISH_TO_MENU = "INSERT INTO dishes_to_menus
(menu_id, dish_id) values(?, ?)";
    private static final String UPDATE_MENU = "UPDATE menus SET name = ?,
description = ?, type = ?, active = ?, start_date = ?, end_date = ? WHERE id = ?";
    private static final Logger LOGGER =
Logger.getLogger(MenuDAO.class.getName());

    private final JdbcTemplate jdbcTemplate;

    public MenuDAO(JdbcTemplate jdbcTemplate) {
        super();
        this.jdbcTemplate = jdbcTemplate;
    }

    @Override
    public void add(Menu t) {
        LOGGER.log(Level.INFO, "Starting menu insert");
        jdbcTemplate.update(INSERT_MENU, t.getName(), t.getDescription(),
t.getType(), t.getActive(), t.getStartDate(), t.getEndDate()
        );
        LOGGER.log(Level.INFO, "Starting menu dishes insert");
        insertDishesToMenu(t);
        LOGGER.log(Level.INFO, "Successful menu insert");
    }

    @Override
    public void delete(Menu t) {
        LOGGER.log(Level.INFO, "Starting menu delete");
        jdbcTemplate.update(DELETE_DISH_TO_MENU, t.getId());
        jdbcTemplate.update(DELETE_MENU, t.getId());
        LOGGER.log(Level.INFO, "Successful menu delete");
    }

    private void insertDishesToMenu(Menu t) {
        for (Dish dish : t.getDishes()) {
            jdbcTemplate.update(INSERT_DISH_TO_MENU, t.getId(),
dish.getId());
        }
    }

    @Override
    public void delete(int id) {
        // TODO Auto-generated method stub
    }

    @Override
    public Menu getById(int id) {
        LOGGER.log(Level.INFO, "Starting menu extraction by id");
        Menu menu = jdbcTemplate.queryForObject(SELECT_MENU_BY_ID, new
MenuRowMapper(), id);
    }

```



```

        List<Dish> dishes = getDishesByMenuId(id);
        if(menu != null) menu.setDishes(dishes);
        LOGGER.log(Level.INFO, "Successful menu extraction by id");
        return menu;
    }

    public List<Dish> getDishesByMenuId(int id){
        LOGGER.log(Level.INFO, "Dish by menu extraction by id");
        List<Dish> dishes = jdbcTemplate.query(DISH_BY_MENU, new
PreparedStatementSetter() {

            @Override
            public void setValues(PreparedStatement ps) throws
SQLException {

                ps.setInt(1, id);

            }}, new DishRowMapper());
        LOGGER.log(Level.INFO, "Successful dish by menu extraction by id");
        return dishes;
    }

    @Override
    public List<Menu> getAll() {
        List<Menu> allMenus = jdbcTemplate.query(SELECT_ALL_MENUS, new
MenuRowMapper());
        for (Menu menu : allMenus) {
            menu.setDishes(getDishesByMenuId(menu.getId()));
        }
        return allMenus;
    }

    @Override
    public void update(Menu t) {
        LOGGER.log(Level.INFO, "Starting Menu update");
        jdbcTemplate.update(UPDATE_MENU, t.getName(), t.getDescription(),
t.getType(), t.getActive(), t.getStartDate(), t.getEndDate(), t.getId());
        jdbcTemplate.update(DELETE_DISH_TO_MENU, t.getId());
        insertDishesToMenu(t);
        LOGGER.log(Level.INFO, "Successful Menu update");
    }

    @Override
    public Menu login(String login, String password) {
        // TODO Auto-generated method stub
        return null;
    }
}

@Component
public class DishDAO implements DAO<Dish> {
    private static final String SELECT_DISH = "SELECT * FROM dishes WHERE id = ?";
    private static final String SELECT_ALL_DISHES = "SELECT * FROM dishes";
    private static final String SELECT_DISH_NAME = "SELECT * FROM dishes WHERE
name = ?";
    private static final String DELETE_DISH_TO_MENU = "DELETE FROM dishes_to_menus
WHERE dish_id = ?";
    private static final String DELETE_DISH = "DELETE FROM dishes WHERE id = ?";
    private static final String INSERT_DISH = "INSERT INTO dishes (name,
ingridients, description, alergens, price, type, category) values(?, ?, ?, ?, ?, ?,
?)";

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		32

```

        private static final String UPDATE_DISH = "UPDATE dishes SET name = ?,
ingridients = ?, description = ?, alergens = ?, price = ?, type = ?, category = ?
WHERE id = ?";

        private static final Logger LOGGER =
Logger.getLogger(DishDAO.class.getName());

        private final JdbcTemplate jdbcTemplate;

        public DishDAO(JdbcTemplate jdbcTemplate) {
            super();
            this.jdbcTemplate = jdbcTemplate;
        }

        @Override
        public void add(Dish t) {
            LOGGER.log(Level.INFO, "Starting dish insert");
            jdbcTemplate.update(INSERT_DISH, t.getName(), t.getIngridients(),
t.getDescription(), t.getAlergens(), t.getPrice(), t.getType(), t.getCategory()
);
            LOGGER.log(Level.INFO, "Successful dish insert");
        }

        @Override
        public void delete(Dish t) {
            LOGGER.log(Level.INFO, "Starting Dish to Menu connection deletion");
            jdbcTemplate.update(DELETE_DISH_TO_MENU, t.getId());
            LOGGER.log(Level.INFO, "Successfull Dish to Menu connection
deletion");

            LOGGER.log(Level.INFO, "Starting Dish deletion");
            jdbcTemplate.update(DELETE_DISH, t.getId());
            LOGGER.log(Level.INFO, "Successfull Dish deletion");
        }

        @Override
        public void delete(int id) {
        }

        @Override
        public Dish getById(int id) {
            LOGGER.log(Level.INFO, "Starting dish extraction");
            Dish dish = jdbcTemplate.query(SELECT_DISH, new
PreparedStatementSetter() {

                @Override
                public void setValues(PreparedStatement ps) throws
SQLException {

                    ps.setInt(1, id);

                }
            }, new ResultSetExtractor<Dish>() {

                @Override
                public Dish extractData(ResultSet rs) throws SQLException,
DataAccessException {

                    return new DishImpl(rs.getInt("id"),
rs.getString("name"), rs.getString("ingridients"),
rs.getString("description"),
rs.getString("alergens"), rs.getInt("price"),
rs.getString("type"),
rs.getString("category"));
                }

            });
            LOGGER.log(Level.INFO, "Successful dish extraction");
            return dish;
        }

        @Override
        public List<Dish> getAll() {

```

```

        LOGGER.log(Level.INFO, "Starting all dish extraction");
        List<Dish> dishes = jdbcTemplate.query(SELECT_ALL_DISHES, new
DishRowMapper());
        LOGGER.log(Level.INFO, "Successful all dish extraction");
        return dishes;
    }

    @Override
    public void update(Dish t) {
        LOGGER.log(Level.INFO, "Starting Dish update");
        jdbcTemplate.update(UPDATE_DISH, t.getName(), t.getIngridients(),
t.getDescription(), t.getAlergens(), t.getPrice(), t.getType(), t.getCategory(),
t.getId());
        LOGGER.log(Level.INFO, "Successful Dish update");
    }

    @Override
    public Dish login(String login, String password) {
        // TODO Auto-generated method stub
        return null;
    }
}

@Component
public class ClientDAO implements DAO<Client> {
    private static final String SELECT_CLIENT = "SELECT * FROM clients WHERE
id=?";
    private static final String SELECT_CLIENT_IV = "SELECT iv FROM clients WHERE
id=?";
    private static final String SELECT_ALL_CLIENTS = "SELECT * FROM clients";
    private static final String SELECT_ALL_IV = "SELECT iv FROM clients";
    private static final String DELETE_CLIENT = "DELETE FROM clients WHERE id=?";
    private static final String INSERT_CLIENT = "INSERT INTO clients (name,
surname, email, phone_number, birth_date, address, register_date, iv) values(?, ?, ?,
?, ?, ?, ?, ?)";
    private static final String UPDATE_CLIENT = "UPDATE clients SET name = ?,
surname = ?, email = ?, phone_number = ?, birth_date = ?, address = ? WHERE id = ?";
    private static final Logger LOGGER =
Logger.getLogger(ClientDAO.class.getName());

    private final JdbcTemplate jdbcTemplate;
    private final EncryptionHandler encryptor;

    public ClientDAO(JdbcTemplate jdbcTemplate, EncryptionHandler ecnryptor) {
        super();
        this.jdbcTemplate = jdbcTemplate;
        this.encryptor = ecnryptor;
    }

    @Override
    public void add(Client t) {
        LOGGER.log(Level.INFO, "Starting Client insert");
        byte[] iv = encryptor.generateVector();
        DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        DateTimeFormatter formD = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        jdbcTemplate.update(INSERT_CLIENT, encryptor.encrypt(t.getName(),
iv), encryptor.encrypt(t.getSurname(), iv),
            encryptor.encrypt(t.getEmail(), iv),
            encryptor.encrypt(t.getPhoneNumber(), iv),

```

```

        encryptor.encrypt(t.getBirthDate().format(formD).toString(), iv),
        encryptor.encrypt(t.getAddress(), iv),

        encryptor.encrypt(t.getRegisterDate().format(form).toString(), iv),
        Base64.getEncoder().encode(iv));
        LOGGER.log(Level.INFO, "Successfull Client insert");
    }

    @Override
    public void delete(Client t) {
        LOGGER.log(Level.INFO, "Starting Client deletion");
        jdbcTemplate.update(DELETE_CLIENT, t.getId());
        LOGGER.log(Level.INFO, "Successfull Client deletion");
    }

    @Override
    public void delete(int id) {
        // TODO Auto-generated method stub
    }

    @Override
    public Client getById(int id) {
        LOGGER.log(Level.INFO, "Starting Client extraction");
        byte[] iv = getIv(id);
        Client client = jdbcTemplate.query(SELECT_CLIENT, new
        PreparedStatementSetter() {

            @Override
            public void setValues(PreparedStatement ps) throws
            SQLException {

                ps.setInt(1, id);

            }
        }, new ResultSetExtractor<Client>() {

            @Override
            public Client extractData(ResultSet rs) throws SQLException,
            DataAccessException {

                return new ClientImpl(rs.getInt("id"),
                encryptor.decrypt(rs.getString("name"), iv),

                encryptor.decrypt(rs.getString("surname"), iv),
                encryptor.decrypt(rs.getString("email"), iv),

                encryptor.decrypt(rs.getString("phone_number"), iv),

                LocalDate.parse(encryptor.decrypt(rs.getString("birth_date"), iv)),

                encryptor.decrypt(rs.getString("address"), iv),

                LocalDateTime.parse(encryptor.decrypt(rs.getString("register_date"), iv)));
            }

        });
        LOGGER.log(Level.INFO, "Successfull Client extraction");
        return client;
    }

    @Override
    public List<Client> getAll() {
        LOGGER.log(Level.INFO, "Starting all clients extraction");
        List<Client> clients = jdbcTemplate.query(SELECT_ALL_CLIENTS, new
        ClientRowMapper(encryptor));
        LOGGER.log(Level.INFO, "Successfull all clients extraction");
        return clients;
    }

    private byte[] getIv(int id) {

```

					КПІ.IT-8223.045430.03.13	Арк.
						35
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        System.out.println("idi " + id);
        String iv = jdbcTemplate.query(SELECT_CLIENT_IV, new
PreparedStatementSetter() {

            @Override
            public void setValues(PreparedStatement ps) throws
SQLException {

                ps.setInt(1, id);

            }

        }, new ResultSetExtractor<String>() {

            @Override
            public String extractData(ResultSet rs) throws SQLException,
DataAccessException {

                rs.next();
                return rs.getString("iv");

            }

        });
        if(!iv.equals(1)) {
            return Base64.getDecoder().decode(iv);
        }

        return null;

    }

    @Override
    public void update(Client t) {
        LOGGER.log(Level.INFO, "Starting staff update");
        byte[] iv = getIv(t.getId());
        DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        if(iv != null) {
            jdbcTemplate.update(UPDATE_CLIENT, encryptor.encrypt(t.getName(),
iv), encryptor.encrypt(t.getSurname(), iv),
                                encryptor.encrypt(t.getEmail(), iv),
                                encryptor.encrypt(t.getPhoneNumber(), iv),

                                encryptor.encrypt(t.getBirthDate().format(form).toString(), iv),
                                encryptor.encrypt(t.getAddress(), iv),
                                t.getId());
        } else {
            jdbcTemplate.update(UPDATE_CLIENT, t.getName(),
t.getSurname(),
                                t.getEmail(), t.getPhoneNumber(),
                                t.getBirthDate().format(form).toString(),
                                t.getAddress(),
                                t.getId());
        }
        LOGGER.log(Level.INFO, "Successfull staff update");
    }

    @Override
    public Client login(String login, String password) {
        // TODO Auto-generated method stub
        return null;
    }

}

public class StaffRowMapper implements RowMapper<Staff> {
    private EncryptionHandler encryptor;

    public StaffRowMapper(EncryptionHandler encryptor) {
        super();
        this.encryptor = encryptor;
    }

    @Override
    public Staff mapRow(ResultSet rs, int rowNum) throws SQLException {

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		36

```

        if (!rs.getString("iv").equals("1")) {
            byte[] iv = Base64.getDecoder().decode(rs.getString("iv"));
            return new StaffImpl(rs.getInt("id"),
                encryptor.decrypt(rs.getString("name"), iv),
                    encryptor.decrypt(rs.getString("surname"), iv),

                Integer.parseInt(encryptor.decrypt(rs.getString("salary"), iv)),
                rs.getString("phone_number"),
                LocalDate.parse(encryptor.decrypt(rs.getString("birth_date"), iv)),
                    encryptor.decrypt(rs.getString("position"), iv));
        } else {
            return new StaffImpl(rs.getInt("id"), rs.getString("name"),
                rs.getString("surname"), rs.getInt("salary"),
                    rs.getString("phone_number"),
                LocalDate.parse(rs.getString("birth_date")),
                    rs.getString("position"));
        }
    }
}

```

```

public class MenuRowMapper implements RowMapper<Menu>{

    @Override
    public Menu mapRow(ResultSet rs, int rowNum) throws SQLException {
        DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        Menu menu = new MenuImpl(rs.getInt("id"), rs.getString("name"),
            rs.getString("description"),
                rs.getString("type"), rs.getBoolean("active"),
                LocalDate.parse(rs.getString("start_date"), form),
                    LocalDate.parse(rs.getString("end_date")));
        return menu;
    }
}

```

```

public class DishRowMapper implements RowMapper<Dish> {

    @Override
    public Dish mapRow(ResultSet rs, int rowNum) throws SQLException {
        Dish dish = new DishImpl(rs.getInt("id"), rs.getString("name"),
            rs.getString("ingridients"),
                rs.getString("description"),
            rs.getString("alergens"), rs.getInt("price"),
                rs.getString("type"), rs.getString("category"));
        return dish;
    }
}

```

```

public class ClientRowMapper implements RowMapper<Client> {
    private EncryptionHandler encryptor;

    public ClientRowMapper(EncryptionHandler encryptor) {
        super();
        this.encryptor = encryptor;
    }

    @Override
    public Client mapRow(ResultSet rs, int rowNum) throws SQLException {
        DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

```

					КПІ.IT-8223.045430.03.13	Арк.
3мін.	Арк.	№ докум.	Підп.	Дата.		37

```

        DateTimeFormatter formD = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        if (!rs.getString("iv").equals("1")) {
            byte[] iv = Base64.getDecoder().decode(rs.getString("iv"));
            return new ClientImpl(rs.getInt("id"),
                encryptor.decrypt(rs.getString("name"), iv),
                encryptor.decrypt(rs.getString("surname"),
                    iv), encryptor.decrypt(rs.getString("email"), iv),

                encryptor.decrypt(rs.getString("phone_number"), iv),

                LocalDate.parse(encryptor.decrypt(rs.getString("birth_date"), iv), formD),
                encryptor.decrypt(rs.getString("address"),
                    iv),

                LocalDateTime.parse(encryptor.decrypt(rs.getString("register_date"), iv),
                    form));
        } else {
            return new ClientImpl(rs.getInt("id"), rs.getString("name"),
                rs.getString("surname"), rs.getString("email"),
                rs.getString("phone_number"),
                LocalDate.parse(rs.getString("birth_date"), formD),
                rs.getString("address"),
                LocalDateTime.parse(rs.getString("register_date"), form));
        }
    }
}

```

@Service

public class BillService {

//private Map<Dish, Integer> dishAmount;

```

public void printBill(String billForm) {
    System.out.println(billForm);
}

```

```

public String formBill(Order order, Map<Dish, Integer> dishAmount) {
    String DATE_FORMATTER = "yyyy-MM-dd HH:mm:ss";
    DateTimeFormatter formatter =
DateTimeFormatter.ofPattern(DATE_FORMATTER);
    String line = "-----";
    String bill = "          Restaurant Name \n\n\n" +
LocalDateTime.now().format(formatter) + "\n" + "Staff: "
+ order.getStaff().getName() + "\n" + line + "\n
Table: " + order.getTableNumber() + " ".repeat(15)
+ "Guests: " + order.getNumberOfClients() + " \n" +
line + "\n";

    int spaceNum = 25;
    String name;
    int amount;
    int sum;
    int total = 0;
    int price;
    for (Map.Entry<Dish, Integer> dishes : dishAmount.entrySet()) {
        name = dishes.getKey().getName();
        amount = dishes.getValue();
        price = dishes.getKey().getPrice();
        sum = price * amount;
        total += sum;
        bill += amount + " " + name + " ".repeat(spaceNum -
name.length() >= 0 ? spaceNum - name.length() : 0) + price
+ " ".repeat(5 -
String.valueOf(sum).length()) + sum + "\n";
    }
    bill += line + "\n" + " ".repeat(15) + "Total: " + total + "\n" + "
".repeat(15) + "Payment type: Card";
    return bill;
}

```

					КПІ.IT-8223.045430.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		38

```

    }

    public void createBill() {

    }

    public void splitBill(Order order, List<List<Integer>> split) {
        Map<Dish, Integer> dishAmount = new HashMap<>();
        System.out.println(order.getDishesByClient());
        int num = 0;
        for (int i = 0; i < split.size(); i++) {
            for (Map.Entry<Client, List<Dish>> orders :
order.getDishesByClient().entrySet()) {
                if (split.get(i).contains(orders.getKey().getId()))
                {
                    for (Dish dish : orders.getValue()) {
                        dishAmount.merge(dish, 1,
Integer::sum);
                    }
                }
                num++;
            }
        }
        System.out.println(dishAmount);
        printBill(formBill(order, dishAmount));
        dishAmount.clear();
        num = 0;
    }
}
}

```

```

@Service
public class ClientService {
    private List<Client> clients;
    @Qualifier
    private DAO<Client> clientDAO;

    public ClientService(DAO<Client> clientDAO) {
        this.clientDAO = clientDAO;
        clients = new ArrayList<>();
        loadClients();
    }

    public void editClientData(int id, Client client) {
        Client editClient = findById(id);
        editClient.setName(client.getName());
        editClient.setSurname(client.getSurname());
        editClient.setEmail(client.getEmail());
        editClient.setPhoneNumber(client.getPhoneNumber());
        editClient.setAddress(client.getAddress());
        editClient.setBirthDate(client.getBirthDate());
        clientDAO.update(editClient);
    }

    public void deleteClient(int id) {
        Client deleteClient = findById(id);
        clients.remove(deleteClient);
        clientDAO.delete(id);
    }

    public void deleteClient(Client client) {
        loadClients();
        Client deleteClient = findByMail(client.getEmail());
        clients.remove(deleteClient);
        clientDAO.delete(deleteClient.getId());
    }

    public Client findById(int id) {

```



```

        Client returnClient = null;
        for (Client client : clients) {
            if (client.getId() == id)
                returnClient = client;
        }
        return returnClient;
    }

    public Client findByMail(String email) {
        Client returnClient = null;
        for (Client client : clients) {
            if (client.getEmail().equals(email))
                returnClient = client;
        }
        return returnClient;
    }

    public Client findByPhone(String phone) {
        Client returnClient = null;
        for (Client client : clients) {
            if (client.getPhoneNumber().equals(phone))
                returnClient = client;
        }
        return returnClient;
    }

    public void registerClient(Client client) {
        clients.add(client);
        clientDAO.add(client);
        loadClients();
    }

    public Client retrieveClient(String email) {
        return findByMail(email);
    }

    public Client retrieveClientPhone(String phone) {
        return findByPhone(phone);
    }

    public List<Client> retrieveClients() {
        return clients;
    }

    public void loadClients() {
        clients = clientDAO.getAll();
    }
}

@Service
public class MenuService {
    private List<Menu> menus;
    private List<Dish> dishesFromMenus;
    private final DAO<Menu> menuDAO;
    private final DAO<Dish> dishDAO;

    public MenuService(DAO<Menu> menuDAO, DAO<Dish> dishDAO) {
        this.menuDAO = menuDAO;
        this.dishDAO = dishDAO;
        menus = new ArrayList<>();
        dishesFromMenus = new ArrayList<>();
        loadMenus();
        loadDishes();
        // getAllDishesByMenus();
    }
}

```

```

    }

    public void addMenu(Menu menu) {
        menuDAO.add(menu);
        loadMenus();
    }

    public void editMenu(int id, Menu menu) {
        Menu menuToEdit = findMenu(id);
        menuToEdit.setName(menu.getName());
        menuToEdit.setDescription(menu.getDescription());
        menuToEdit.setActive(menu.getActive());
        menuToEdit.setType(menu.getType());
        menuToEdit.setStartDate(menu.getStartDate());
        menuToEdit.setEndDate(menu.getEndDate());
        menuDAO.update(menuToEdit);
    }

    public void editMenu(String name, Menu menu) {
        Menu menuToEdit = findMenu(name);
        menuToEdit.setName(menu.getName());
        menuToEdit.setDescription(menu.getDescription());
        menuToEdit.setActive(menu.getActive());
        menuToEdit.setType(menu.getType());
        menuToEdit.setStartDate(menu.getStartDate());
        menuToEdit.setEndDate(menu.getEndDate());
        menuDAO.update(menuToEdit);
    }

    public Menu findMenu(int id) {
        Menu menuToReturn = null;
        for (Menu menu : menus) {
            if (menu.getId() == id)
                menuToReturn = menu;
        }
        return menuToReturn;
    }

    public Menu findMenu(String name) {
        Menu menuToReturn = null;
        for (Menu menu : menus) {
            if (menu.getName().equals(name))
                menuToReturn = menu;
        }
        return menuToReturn;
    }

    public List<Menu> getAllMenus() {
        return menus;
    }

    public void removeMenu(int id) {
        Menu menuToDelete = findMenu(id);
        menus.remove(menuToDelete);
        for (Dish dish : menuToDelete.getDishes()) {
            removeDish(dish.getId());
        }
        menuDAO.delete(menuToDelete);
    }

```

```

        loadMenus();
    }

    public void removeMenu(String name) {
        Menu menuToDelete = findMenu(name);
        menuDAO.delete(menuToDelete.getId());
    }

    public void loadMenus() {
        menus = menuDAO.getAll();
    }

    public void addDishToMenu(int id, Dish dish) {
        Menu menuToAdd = findMenu(id);
        dishDAO.add(dish);
        loadDishes();
        System.out.println(dishesFromMenus);
        Dish dishLoaded = findDish(dish.getName());
        System.out.println(dishLoaded.getId());
        menuToAdd.getDishes().add(dishLoaded);
        menuDAO.update(menuToAdd);
        loadMenus();
    }

    public void removeDishInMenu(Dish dish) {
        for (Menu menu : menus) {
            for (Dish d : menu.getDishes()) {
                if (d.getId() == dish.getId()) {
                    menu.getDishes().remove(d);
                }
            }
        }
    }

    public Dish findDishInMenus(Dish dish) {
        Dish dishRetun = null;
        for (Menu menu : menus) {
            for (Dish d : menu.getDishes()) {
                if (d.getId() == dish.getId()) {
                    dishRetun = d;
                }
            }
        }
        return dishRetun;
    }

    public void addDishToMenu(int id, int dishId) {
        Menu menuToAdd = findMenu(id);
        menuToAdd.getDishes().add(dishDAO.getById(dishId));

        // finish addition
    }

    public void addDish(Dish dish) {
        dishDAO.add(dish);
    }

```

```

public void editDish(Dish dish) {
    Dish toEdit = findDishInMenus(dish);
    toEdit.setAlergens(dish.getAlergens());
    toEdit.setCategory(dish.getCategory());
    toEdit.setName(dish.getName());
    toEdit.setPrice(dish.getPrice());
    toEdit.setDescription(dish.getDescription());
    toEdit.setType(dish.getType());
    toEdit.setIngridients(dish.getIngridients());
    dishDAO.update(toEdit);
}

public void removeDish(int id) {
    Dish toRemove = findDish(id);
    System.out.println(toRemove);
    dishesFromMenus.remove(toRemove);
    // removeDishInMenu(toRemove);
    dishDAO.delete(toRemove);
    loadMenus();
}

public void addDish(int id, int dishId) {
    Menu menuToAdd = findMenu(id);
    menuToAdd.getDishes().add(dishDAO.getById(dishId));
}

public Dish findDish(int id) {
    getAllDishesByMenus();
    Dish dishToReturn = null;
    for (Dish dish : dishesFromMenus) {
        if (dish.getId() == id)
            dishToReturn = dish;
    }
    return dishToReturn;
}

public Dish findDish(String name) {
    Dish dishToReturn = null;
    for (Dish dish : dishesFromMenus) {
        if (dish.getName().equals(name))
            dishToReturn = dish;
    }
    return dishToReturn;
}

public void getAllDishesByMenus() {
    for (Menu menu : menus) {
        dishesFromMenus.addAll(menu.getDishes());
    }
}

public List<Dish> getAllDishes() {
    return dishesFromMenus;
}

public void loadDishes() {
    dishesFromMenus = dishDAO.getAll();
}

```

```

    }

}

@Service
public class OrderManager implements OrderService {
    private List<Order> historyOfOrders;
    private List<Order> activeOrders;
    @Qualifier("orderDAO")
    private final DAO<Order> orderDao;
    @Autowired
    private BillService billService;

    public OrderManager(DAO<Order> orderDao) {
        this.orderDao = orderDao;
        loadHistory();
        activeOrders = new ArrayList<>();
    }

    public BillService getBillService() {
        return billService;
    }

    public void setBillService(BillService billService) {
        this.billService = billService;
    }

    @Override
    public void addOrder(Order order) {
        activeOrders.add(order);
        // orderDao.add(order);
    }

    @Override
    public void changeOrder(int id, Order order) {
        Order orderToChange = findOrder(id);

        orderToChange.setNumberOfClients(order.getNumberOfClients());
        orderToChange.setTableNumber(order.getTableNumber());
    }

    @Override
    public void addClients(int id, Client client) {
        Order orderToChange = findOrder(id);
        if (client.equals(null)) {
            orderToChange.addClient(new
ClientImpl(orderToChange.getNumberOfClients() + 1));
        } else {
            //
            orderToChange.setNumberOfClients(orderToChange.getNumberOfClients() +
1);
            orderToChange.addClient(client);
        }
    }
}

```

```

    }

    @Override
    public void addDishToClientOrder(int id, int clientId, Dish
dish) {
        Order order = findOrder(id);
        Client client = findClientInOrderByID(order, clientId);
        if (order.getDishesByClient().containsKey(client)) {
            order.getDishesByClient().get(client).add(dish);
        }
    }

    @Override
    public void deleteDishFromClientOrder(int id, int clientId,
Dish dish) {
        Order order = findOrder(id);
        Client client = findClientInOrderByID(order, clientId);
        if (order.getDishesByClient().containsKey(client)) {

            order.getDishesByClient().get(client).remove(dish);
        }
    }

    @Override
    public void deleteClientFromOrder(int id, int clientId) {
        Order order = findOrder(id);
        Client client = findClientInOrderByID(order, clientId);
        if (order.getDishesByClient().containsKey(client)) {
            order.getDishesByClient().remove(client);
        }
    }

    public void addDishToClientOrder(int id, Client client, Dish
dish) {
        Order orderToChange = findOrder(id);

    }

    public List<Client> getClientsByOrder(int orderId) {
        Order order = findOrder(orderId);
        return new
ArrayList<>(order.getDishesByClient().keySet());
    }

    public List<Dish> getDishesByActiveOrder(int orderId) {
        return getDishesByOrder(findOrder(orderId));
    }

    public List<Dish> getDishesByOrder(Order order) {
        List<Dish> dishes = new ArrayList<>();
        for (List<Dish> dishH :
order.getDishesByClient().values()) {
            dishes.addAll(dishH);
        }
        return dishes;
    }
}

```

```

    public List<Dish> getDishesByClientAndOrder(int orderId, Client
client) {
        Order order = findOrder(orderId);
        if (order == null) {
            order = findOrderHistory(orderId);
        }
        return order.getDishesByClient().get(client);
    }

    public List<Dish> getDishesByOrderHistory(int orderId) {
        return getDishesByOrder(findOrderHistory(orderId));
    }

    public Client findClientInOrderByID(Order order, int clientId)
{
        Client client = null;
        for (Entry<Client, List<Dish>> clientToDishes :
order.getDishesByClient().entrySet()) {
            if (clientToDishes.getKey().getId() == clientId) {
                client = clientToDishes.getKey();
            }
        }
        return client;
    }

    public Order findOrder(int id) {
        Order returnOrder = null;
        for (Order order : activeOrders) {
            if (order.getId() == id)
                returnOrder = order;
        }
        return returnOrder;
    }

    public Order findOrderHistory(int id) {
        Order returnOrder = null;
        for (Order order : historyOfOrders) {
            if (order.getId() == id)
                returnOrder = order;
        }
        return returnOrder;
    }

    @Override
    public void completeOrder() {
        // TODO Auto-generated method stub
    }

    @Override
    public void addClients() {
        // TODO Auto-generated method stub
    }

    @Override
    public void splitBill() {
        // TODO Auto-generated method stub
    }

```

```

    }

    @Override
    public void sendForCooking() {
        // TODO Auto-generated method stub

    }

    @Override
    public void checkOut(int id, int tips, List<List<Integer>>
splitting) {
        Order orderToCheckout = findOrder(id);
        orderToCheckout.setTips(tips);
        billService.splitBill(orderToCheckout, splitting);
        orderDao.add(orderToCheckout);
        activeOrders.remove(orderToCheckout);

    }

    @Override
    public int generateId() {
        return historyOfOrders.get(historyOfOrders.size() -
1).getId() + 1;

    }

    @Override
    public List<Order> getAll() {
        return activeOrders;

    }

    @Override
    public List<Order> getHistory() {
        loadHistory();
        return historyOfOrders;

    }

    @Override
    public List<Order> findClientOrder(Client client) {
        List<Order> orders = new ArrayList<>();
        for (Order o : historyOfOrders) {
            if (o.getDishesByClient().containsKey(client))
                orders.add(o);

        }
        return orders;

    }

    @Override
    public List<Order> findOrderByDate(LocalDate date) {
        List<Order> orders = new ArrayList<>();
        for (Order o : historyOfOrders) {
            if
(o.getOrderDateTime().toLocalDate().equals(date))
                orders.add(o);

        }
        return orders;

    }

```



```

        public void loadHistory() {
            historyOfOrders = orderDao.getAll();
        }

    }

@Service
public class StaffService {
    private List<Staff> staffs;
    private Staff loggedIn;
    private final DAO<Staff> staffDAO;

    public StaffService(DAO<Staff> staffDAO) {
        super();
        this.staffDAO = staffDAO;
        staffs = new ArrayList<>();
        loadStaff();
    }

    public void editStaff(int id, Staff staff) {
        Staff staffToEdit = findStaffById(id);
        staffToEdit.setName(staff.getName());
        staffToEdit.setSurname(staff.getSurname());
        staffToEdit.setPhoneNumber(staff.getPhoneNumber());
        staffToEdit.setBirthDate(staff.getBirthDate());
        staffToEdit.setSalary(staff.getSalary());
        staffToEdit.setPosition(staff.getPosition());
        staffToEdit.setPassword(staff.getPassword());
        staffDAO.update(staffToEdit);
        //loadStaff();
    }

    public Staff login(String phone, String password) {
        Staff staff = staffDAO.login(phone, password);
        loggedIn = staff;
        return staff;
    }

    public void promote(int id, String position) {
        Staff staffToPromote = findStaffById(id);
        staffToPromote.setPosition(position);
        staffDAO.update(staffToPromote);
    }

    public Staff findStaffById(int id) {
        Staff staffToReturn = null;
        for (Staff staff : staffs) {
            if(staff.getId() == id) staffToReturn = staff;
        }
        return staffToReturn;
    }
}

```

```

    public Staff findStaffByPhone(String phone) {
        Staff staffToReturn = null;
        for (Staff staff : staffs) {
            if(staff.getPhoneNumber().equals(phone))
staffToReturn = staff;
        }
        return staffToReturn;
    }

    public Staff findStaffBySurname(String surname) {
        Staff staffToReturn = null;
        for (Staff staff : staffs) {
            if(staff.getSurname().equals(surname) ||
staff.getName().equals(surname)) staffToReturn = staff;
        }
        return staffToReturn;
    }

    public void addStaff(Staff staff) {
        //staffs.add(staff);
        staffDAO.add(staff);
        loadStaff();
    }

    public void deleteStaff(Staff staff) {
        //loadStaff();
        Staff staffToDel = findStaffById(staff.getId());
        staffs.remove(staffToDel);
        staffDAO.delete(staffToDel.getId());
    }

    public List<Staff> getStaff(){
        return staffs;
    }

    public void loadStaff() {
        staffs = staffDAO.getAll();
    }

}

```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2022 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРИЙОМУ ТА ОБРОБКИ
ЗАМОЛВЕНЬ В РЕСТОРАННОМУ БІЗНЕСІ**

Програма та методика тестування

КПІ.IT-8223.045430.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:

_____ Валерій НОВІНСЬКИЙ

Виконавець:

_____ Александер СЕЛЮК

Київ – 2022

ЗМІСТ

1 Об'єкт випробувань	3
2 Мета тестування	4
3 Методи тестування.....	5
4 Засоби та порядок тестування.....	6

					КП.ІТ-8223.045430.04.51	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є програмне забезпечення, що допомагає керувати життєвим циклом замовлень в ресторанному бізнесі. Додаток створений на мові Java з використанням фреймворку Spring та технології JavaFX для інтерфейсу.

					КП.ІТ-8223.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка програмного забезпечення на якість і буде включати такі вимоги:

- перевірка правильності роботи програмного забезпечення у відповідно до функціональних вимог;
- знаходження проблем та недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу;
- перевірка обробки невірного введення даних користувачем;
- перевірка обробки непризначених дій користувача.

					КПІ.ІТ-8223.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється уся документація, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – перевіряється програма на безпомилкове виконання всіх реалізованих функцій.
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на безлічі тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми в роботі програми;
- тестування «білої скриньки» – об'єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним.

					КП.ІТ-8223.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується вручну, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності в користуванні. Для того, щоб перевірити працездатність та відмовостійкість додатку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на операційних системах сімейства Windows та Unix.
- тестування обробки введення некоректних даних та виводу повідомлення про помилку;
- тестування збереження цілісності даних при виконанні операцій з базою даних.
- тестування інтерфейсу користувача;
- тестування зручності використання.

					КПІ.ІТ-8223.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2022 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРИЙОМУ ТА ОБРОБКИ
ЗАМОЛВЕНЬ В РЕСТОРАННОМУ БІЗНЕСІ**

Керівництво користувача

КПІ.IT-8223.045430.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:

_____ Валерій НОВІНСЬКИЙ

Виконавець:

_____ Селюк Александер

Київ – 2022

ЗМІСТ

1 Загальні відомості	3
2 Підготовка до роботи.....	4
2.1 Системні вимоги для коректної роботи	4
2.2 Завантаження застосунку.....	4
2.3 Перевірка коректної роботи.....	4
3 Робота із застосунком.....	5

					КПІ.ІТ-8223.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		2

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програма для прийому та обробки замовлень створене для пришвидшення та допомоги управління замовленнями в ресторанному бізнесі. Додаток створений для простої та швидкої роботи замовленнями та має такі функції:

- прийом замовлень;
- змінення замовлення;
- видалення замовлень;
- додавання клієнтів до замовлень;
- додавання страв до замовлень;
- реєстрація клієнтів;
- зміна клієнтів;
- видалення клієнтів;
- реєстрація персоналу;
- зміна персоналу;
- видалення персоналу;
- додавання меню;
- змінення меню;
- видалення меню;
- додавання страв;
- додавання страв до меню;
- видалення страв з меню;
- зміна страв;
- розрахунок замовлень.

					КПІ.ІТ-8223.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність комп'ютера з операційною системою Windows 10, MacOS або Linux;
- встановлена віртуальна машина Java версії 11 або вище;
- для встановлення додатку на пристрої повинно бути не менше 100 МБ вільної пам'яті;
- мінімальна кількість оперативної пам'яті становить 2 гігабайти;
- на пристрої повинна бути встановлена та налаштована база даних. MySQL або MariaDB.

2.2 Завантаження застосунку

Застосунок запускається безпосередньо відкриттям .jar архіву під назвою «Restaurant.jar» тому завантаження не потребує.

2.3 Перевірка коректної роботи

По завершенню встановлення додатка на робочому столі мобільного пристрою повинна відобразитись іконка даного застосунку. У разі, якщо дана іконка не з'явилась, то встановлення відбулось не успішно. Інакше користувач має змогу запустити додаток, клацнувши на його іконку. Після натискання повинна відобразитись початкова сторінка застосунку.

					КПІ.IT-8223.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 РОБОТА ІЗ ЗАСТОСУНКОМ

Після успішного запуску користувачу буде відображено сцену авторизації(рис. 3.1) з двома полями для вводу логіну та пароллю.

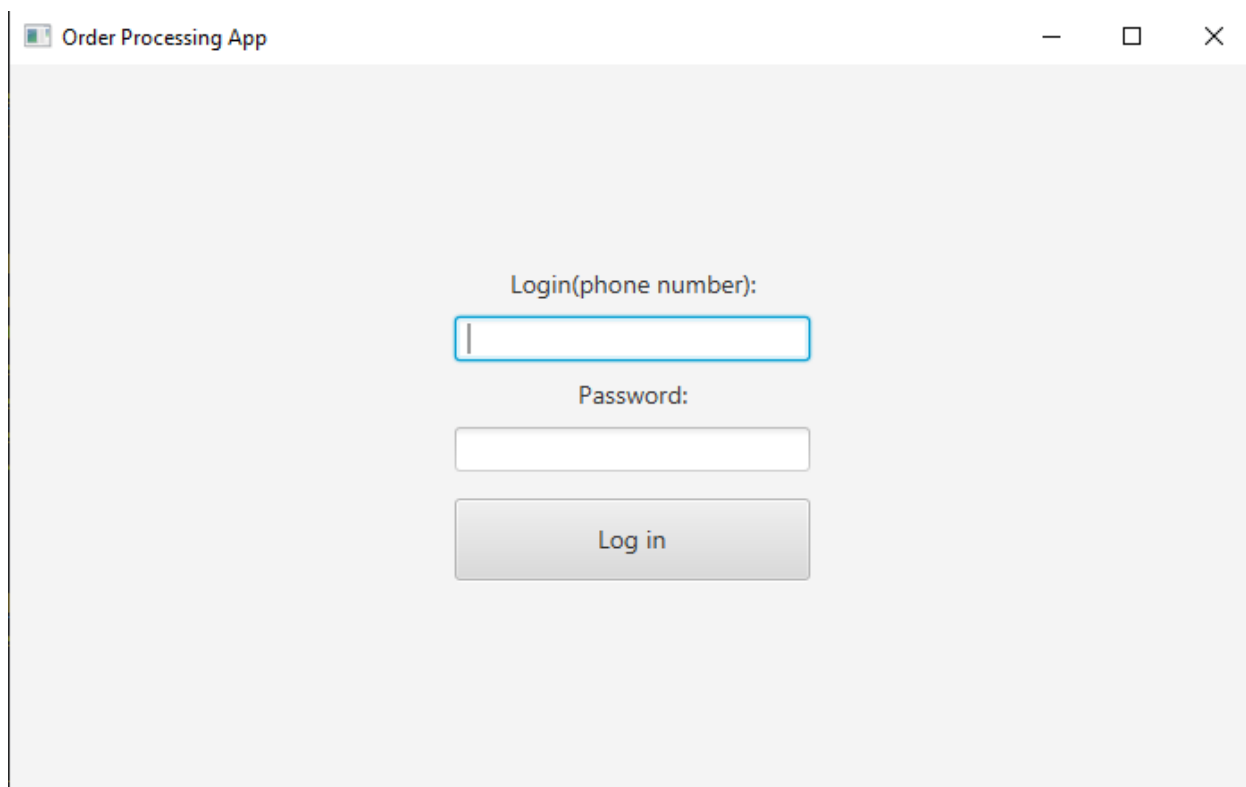


Рисунок 3.1 – Сцена авторизації

Від користувача потребується ввід номера телефону та пароллю за якими було виконано реєстрацію. При не співпадінні даних з збереженими у базі виведеться вікно помилки(рис. 3.2). Та прохання повторити ввід даних.

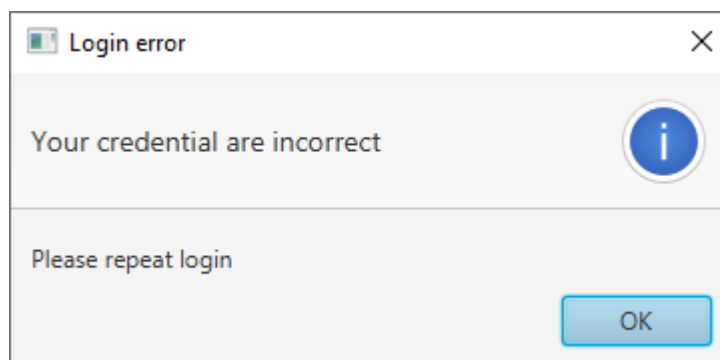


Рисунок 3.2 – Вивід помилки про не співпадіння даних

При коректному вводі даних буде закрита форма авторизації та відображена форма управління з доступними вкладками управління меню та персоналом якщо користувач адміністратор(рис 3.3), та з недоступними якщо користувач офіціант(рис. 3.4).

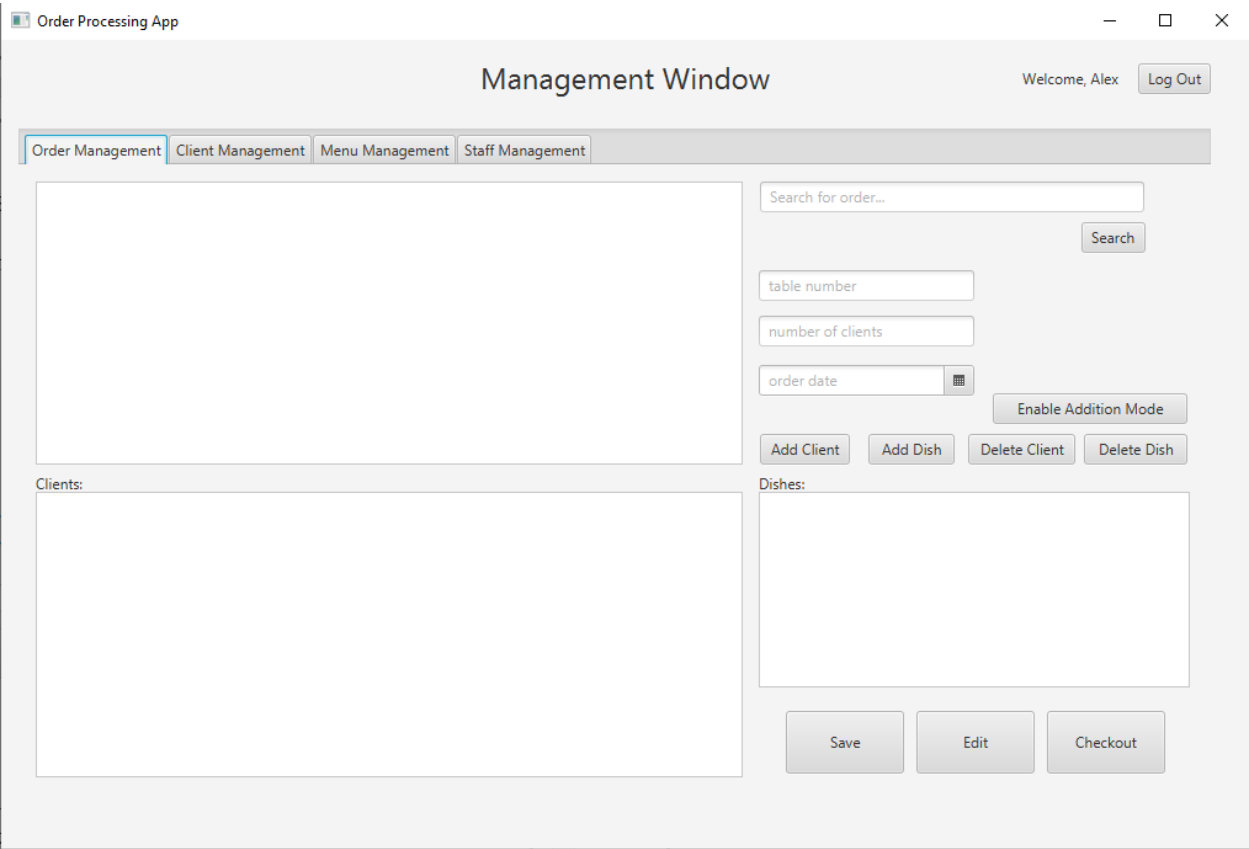


Рисунок 3.3 – Форма управління для адміністратора

Рисунок 3.4 – Форма управління для офіціанта

Після входу в правому верхньому куту є кнопка виходу з акаунту, після натискання її користувач повернеться на сцену авторизації(рис. 3.1)

Для створення та управління клієнтами користувач має натиснути на сторінку роботи з клієнтами після чого програма переключиться на це вікно(рис 3.5). На ньому можна створювати, редагувати та видаляти клієнтів.

Рисунок 3.5 – Сторінка управління клієнтам.

Для пошуку клієнта по телефону або за електронною поштою слід ввести ці дані у текстове поле пошуку у правій верхній частині вікна та натиснути на кнопку пошуку. Якщо поле буде пустим виведуться усі користувачі. Для реєстрації треба ввести всі дані користувача крім дати

реєстрації і натиснути на кнопку «Register»(рис. 3.6). Клієнт зареєстрований.

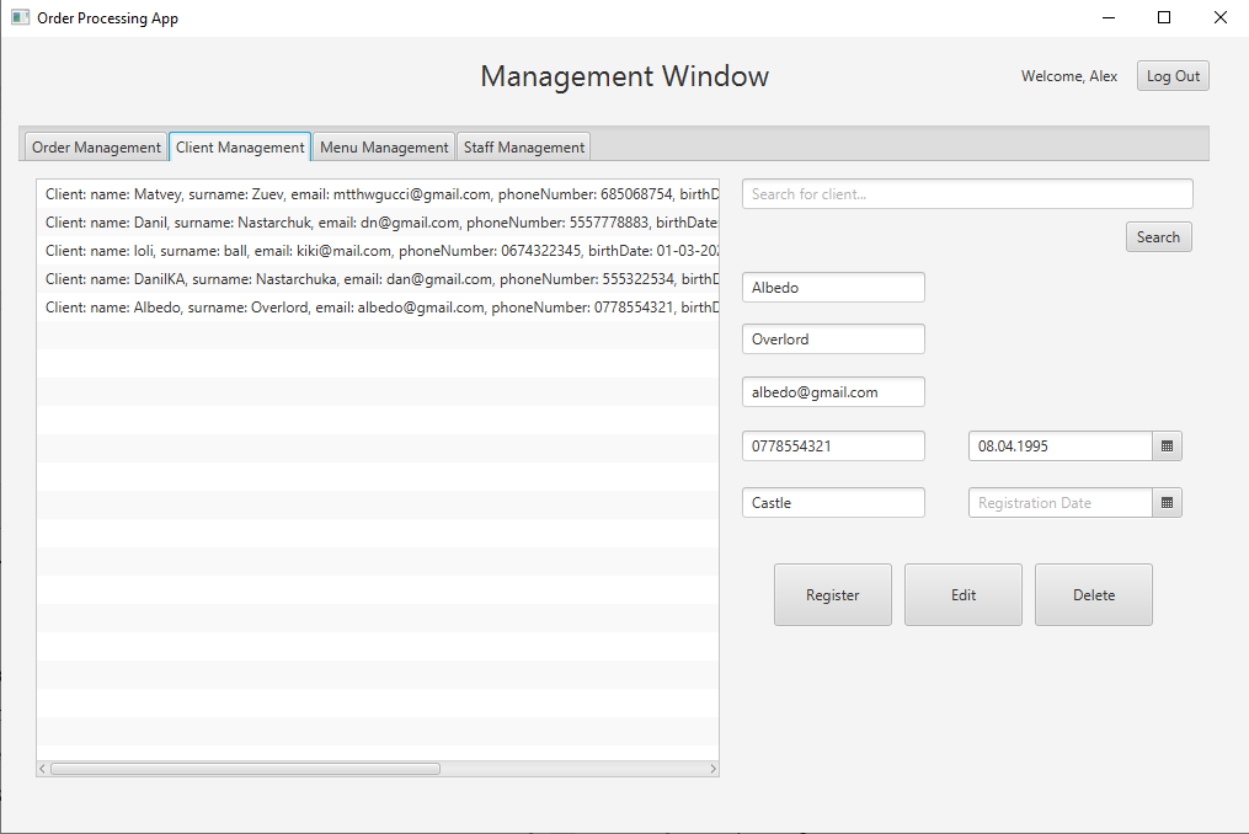


Рисунок 3.6 – Додавання клієнта.

Для зміни користувача слід знайти його в списку та вибрати, після чого дані виведуться у текстові поля, далі треба змінити дані та натиснути кнопку зміни. Для видалення користувача слід вибрати його та натиснути кнопку видалення.

Для менеджменту персоналом треба натиснути на вікно персоналу, сторінка переключиться для управління персоналом(рис. 3.7)

Рисунок 3.7 – Сторінка управління персоналом.

Для пошуку персоналу по телефону або за фамілією слід ввести ці дані у текстове поле пошуку у правій верхній частині вікна та натиснути на кнопку шукати, якщо персонал не буде знайдено, виведуться увесь персонал. Для додавання персоналу треба ввести усі дані до належних для цього полів і натиснути кнопку реєстрації(рис. 3.8).

Order Processing App

Management Window

Welcome, Alex Log Out

Order Management Client Management Menu Management Staff Management

Staff: name: Steven, surname: Armstrong, password: null, salary: 1900, phoneNumber: 005467584, birthDate: 10.01.1997

Staff: name: Jotaro, surname: Kujo, password: null, salary: 12000, phoneNumber: 1122334455, birthDate: 02-06-2000

Staff: name: Joline, surname: Kujo, password: null, salary: 12000, phoneNumber: 1223334444, birthDate: 02-06-2000

Staff: name: Dio, surname: Branlol, password: null, salary: 12000, phoneNumber: 1223334456, birthDate: 02-06-2000

Staff: name: Albedo, surname: Overlord, password: null, salary: 100001, phoneNumber: 23333344444, birthDate: 02-06-2000

Staff: name: Alex, surname: Seljuk, password: null, salary: 120000, phoneNumber: 0688630116, birthDate: 02-06-2000

Staff: name: lol, surname: loool, password: null, salary: 123, phoneNumber: 56744345, birthDate: 02-06-2000

Staff: name: Biba, surname: Boba, password: null, salary: 1, phoneNumber: 12345678, birthDate: 24-11-2000

Staff: name: Catrine, surname: Sonch, password: null, salary: 18000, phoneNumber: +380684567636, birthDate: 10.01.1997

Search for staff...

Search

Catrine

Sonch

Som123456@

+380684567636

waiter

18000

10.01.1997

Register

Edit

Delete

Рисунок 3.8 –Додавання персоналу.

Для зміни персоналу треба вибрати його зі списку, після чого його дані виведуться в поля для зміни, після натискання на кнопку редагування інформацію про персонал буде змінено. Для видалення достатньо вибрати персонал зі списку та натиснути на кнопку видалення.

Для управління меню та стравами треба натиснути на вкладку менеджменту меню, після цього сцену буде змінено(рис. 3.9)

Management Window

Welcome, Alex Log Out

Order Management

Client Management

Menu Management

Staff Management

Search for menu...

Search

Name

Description

Type

☐ Active

Start date

End date

Add Menu

Edit Menu

Delete Menu

Dishes:

Name

Ingridients

Price

Description

Type

Alergens

Category

Add dish

Edit dish

Delete dish

Рисунок 3.9 – Сторінка управління меню та стравами.

Для пошуку меню слід ввести назві у текстове поле пошуку у правій верхній частині вікна та натиснути на кнопку шукати. Для додавання меню потрібно ввести його дані та натиснути на кнопку «Add Menu»(рис. 3.10)

Рисунок 3.10 – Додавання меню.

Вибравши меню зі списку можна ввести дані у поля та натиснувши «Edit Menu» змінити його параметри. Для видалення меню достатньо вибрати меню зі списку та натиснути на кнопку «Delete Menu», при цьому всі страви, асоційовані з ним, будуть також видалені.

Для перегляду списку страв меню достатньо вибрати його у списку та всі його страви будуть відображені на вікні. Для додавання страви треба вибрати меню в списку та ввести її дані у нижні поля сторінки і після цього натиснути кнопку «Add dish»(рис. 3.11).

Order Processing App

Management Window

Welcome, Alex [Log Out](#)

Order Management Client Management Menu Management Staff Management

Menu: name: summer, description: lolol, type: summer vibes, active: true, startDate: 01-06-2022, endDate: 30-05-2022

Menu: name: Special Dining, description: For best guests, type: from 6pm, active: true, startDate: 01-06-2022, endDate: 30-05-2022

Menu: name: Best menu, description: yummy yummy, type: only on saturdays, active: true, startDate: 30-05-2022, endDate: 01-09-2022

Search for menu... [Search](#)

Best menu

yummy yummy

only on saturdays ☒ Active

30.05.2022 01.09.2022

[Add Menu](#) [Edit Menu](#) [Delete Menu](#)

Dishes:

Dish: name: Pinapple cake, ingridients: Pinaple, cake, description: very tasty pinaple cake with little pieces

Pinapple cake

Pinaple, cake 199

very tasty pinaple cake wi Cakes

HGNMB Desserts

[Add dish](#) [Edit dish](#) [Delete dish](#)

Рисунок 3.11 – Додавання страви до меню.

Страва буде додана та відображена у списку страв. Для редагування страви слід вибрати меню зі списку та потім обрати потрібну страву зі списку нижче. Дані обраної страви виведуться в тестові поля для редагування і після натискання кнопки «Edit dish» збережуться зроблені зміни. Для видалення страв потрібно обрати меню та його страву і натиснути кнопку «Delete dish» після чого страву буде видалена.

Для обробки замовлень користувачу потрібно вибрати вкладку «Order management», після натискання відобразиться сторінка управління замовленнями(рис. 3.12).

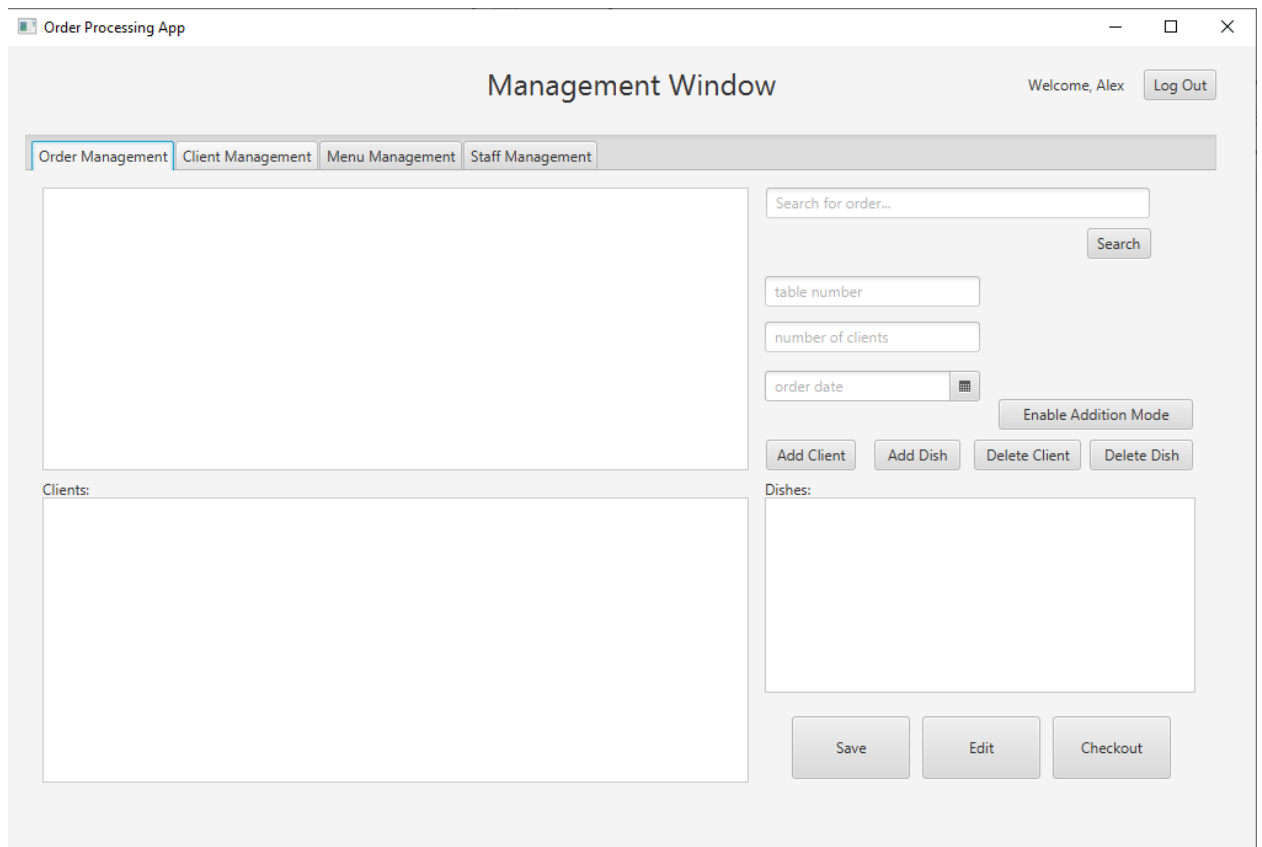


Рисунок 3.12 – Сторінка управління замовленнями

Для пошуку в історії замовлень достатньо ввести ключове слово у строку пошуку та натиснути кнопку «Search». Якщо поле буде пустим, виведеться уся історія замовлень. Для додавання замовлення треба ввести номер столу та кількість клієнтів у поля «table number» та «number of clients» відповідно та натиснути кнопку «Save». Після цього буде додано відображено нове, активне замовлення у списку замовлень (рис. 3.13)

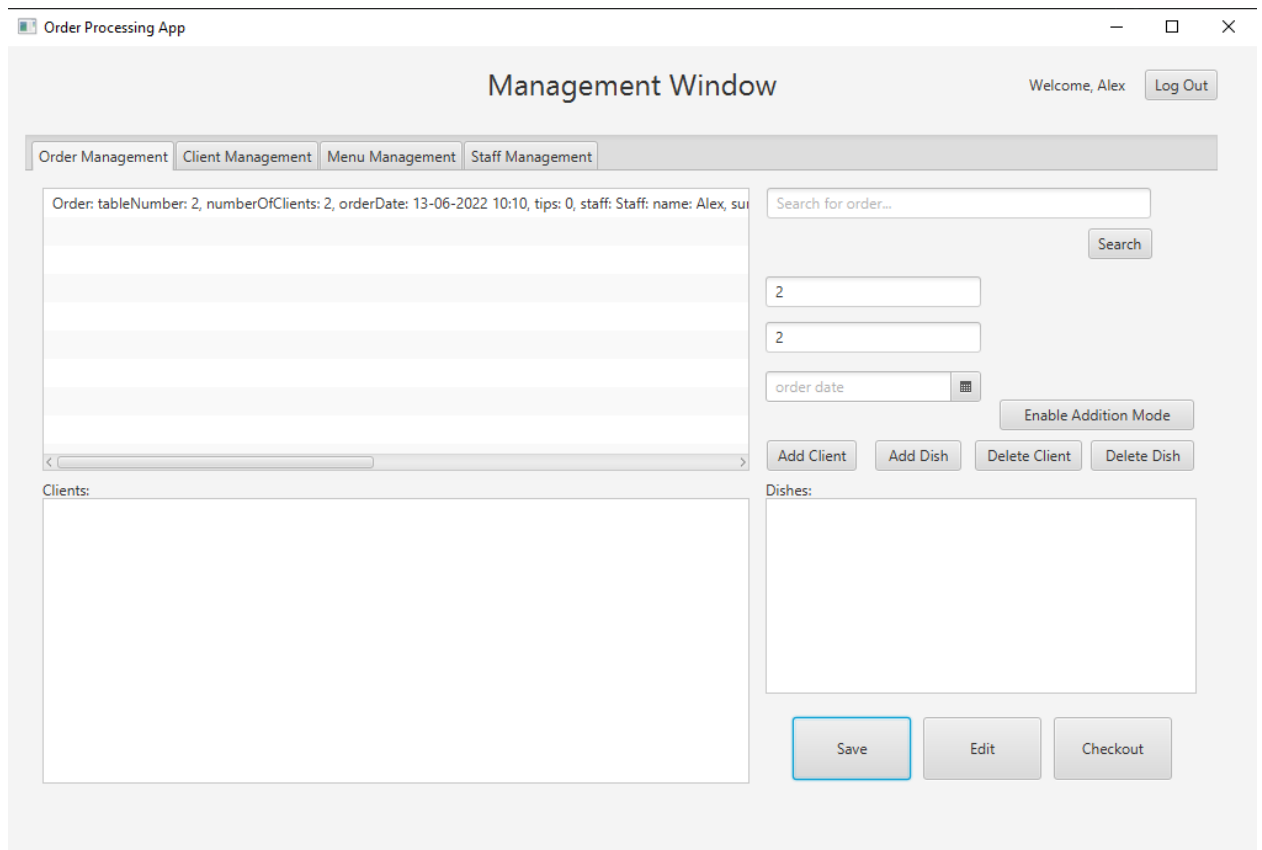


Рисунок 3.13 – Додане та відображене замовлення

Для редагування параметрів замовлення потрібно вибрати його зі списку замовлень ,після чого його параметри виведуться у поля для зміни(рис. 3.14) І після натискання кнопки «Edit» замовлення буде змінено та відображено з новими параметрами.

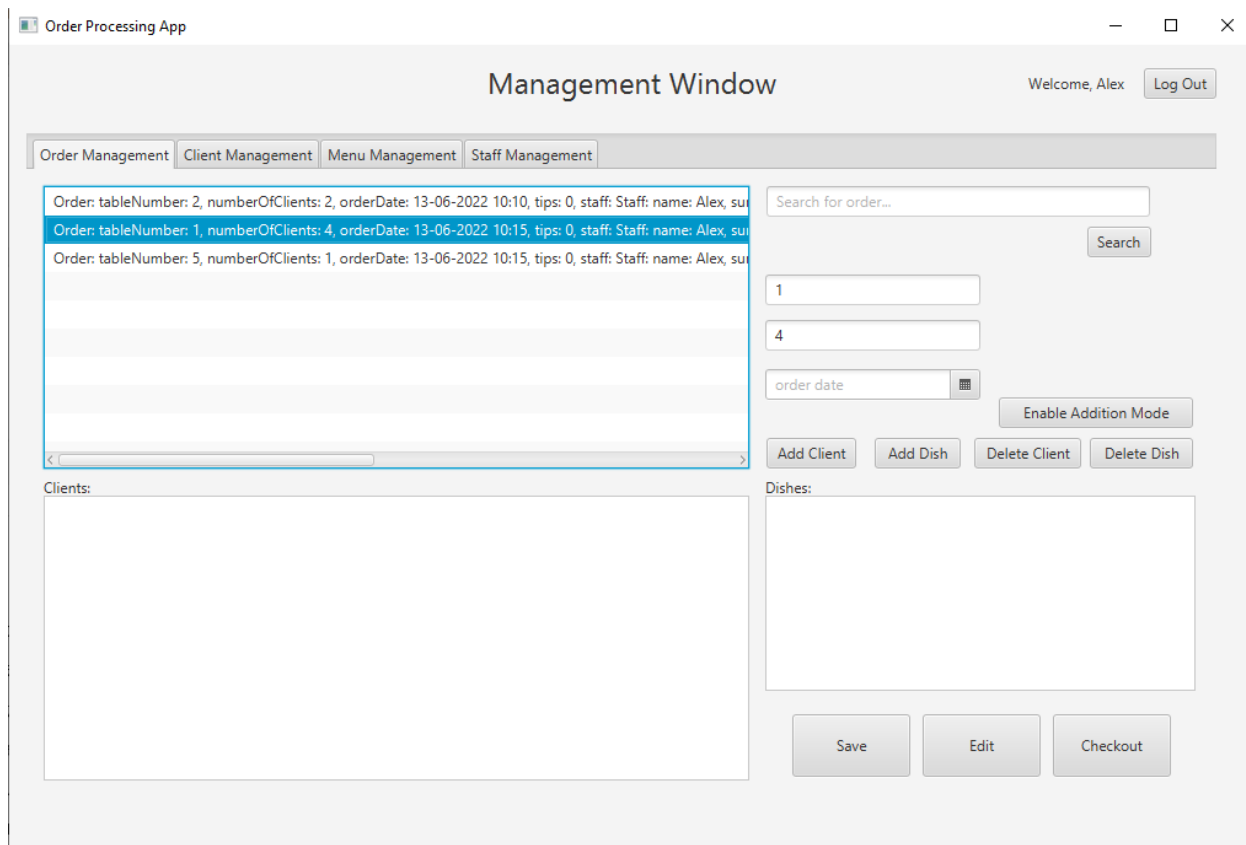


Рисунок 3.14 – Вибране замовлення для редагування

Для додавання клієнтів і їх страв потрібно натиснути на кнопку «Enable Addition Mode» і вибрати замовлення зі списку. Після чого відобразяться доступні клієнти та страви для додавання (рис. 3.15)

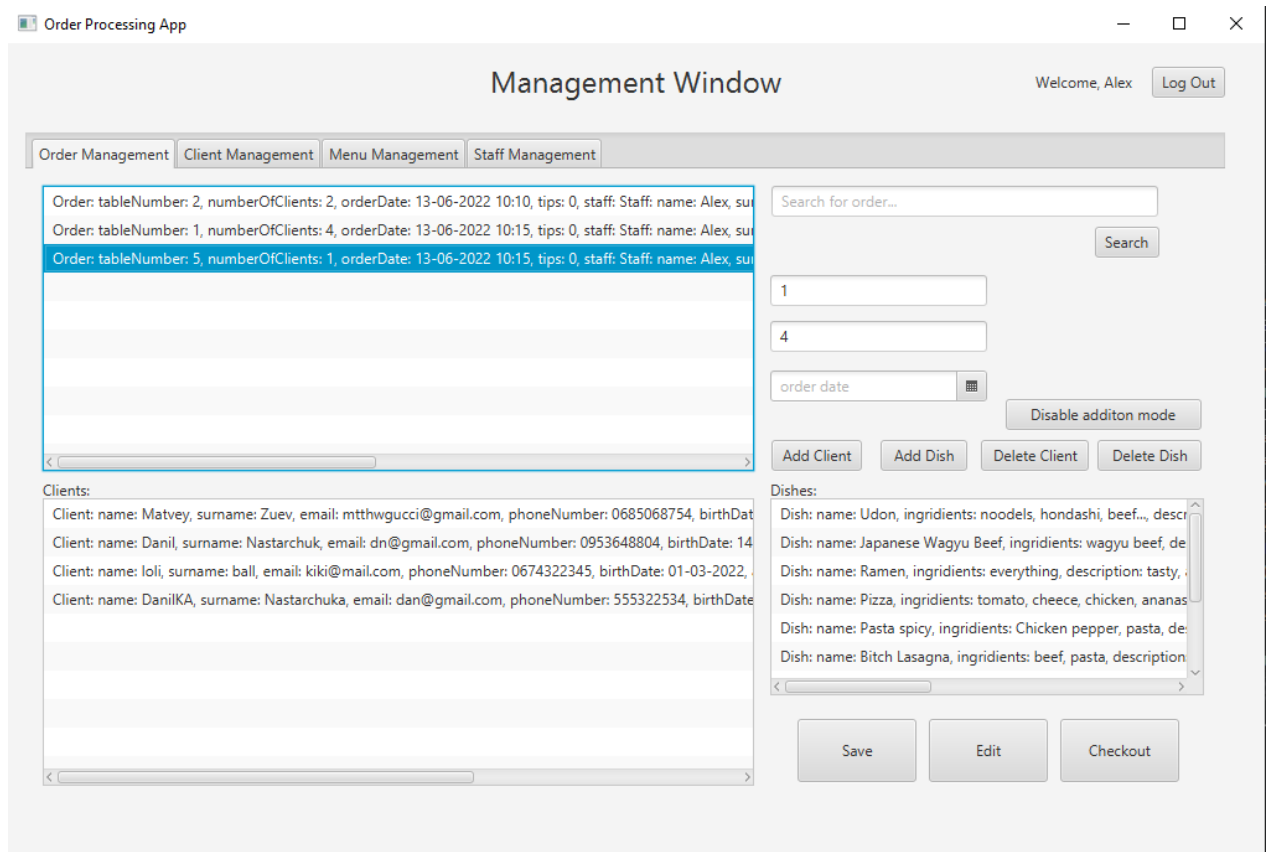


Рисунок 3.15 – Вибране замовлення в режимі додавання

Щоб додати клієнта потрібно вибрати його зі списку та натиснути кнопку «Add Client» після цього клієнта буде додано, а для додавання страв до обраного клієнту треба вибрати страву зі списку та натиснути кнопку «Add Dish» після чого страву буде додано. Щоб переглянути клієнтів або страви в замовленні треба вийти із режиму додавання натисканням кнопки «Disable addition mode» та вибрати замовлення зі списку, після чого відобразиться список клієнтів в замовленні та список всіх страв(рис.3.16). При виборі клієнта можна побачити список замовлених ним страв(рис. 3.17).

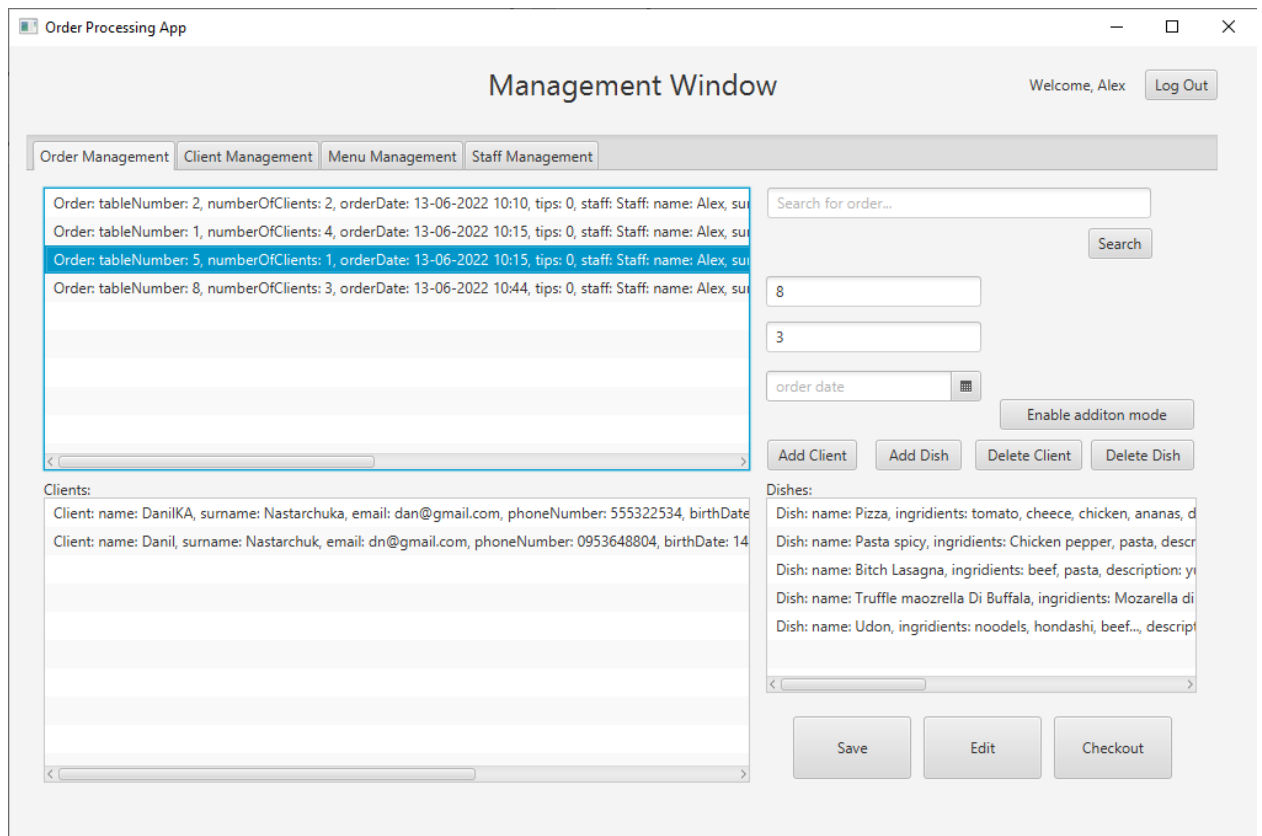


Рисунок 3.16 – Вибране замовлення для перегляду

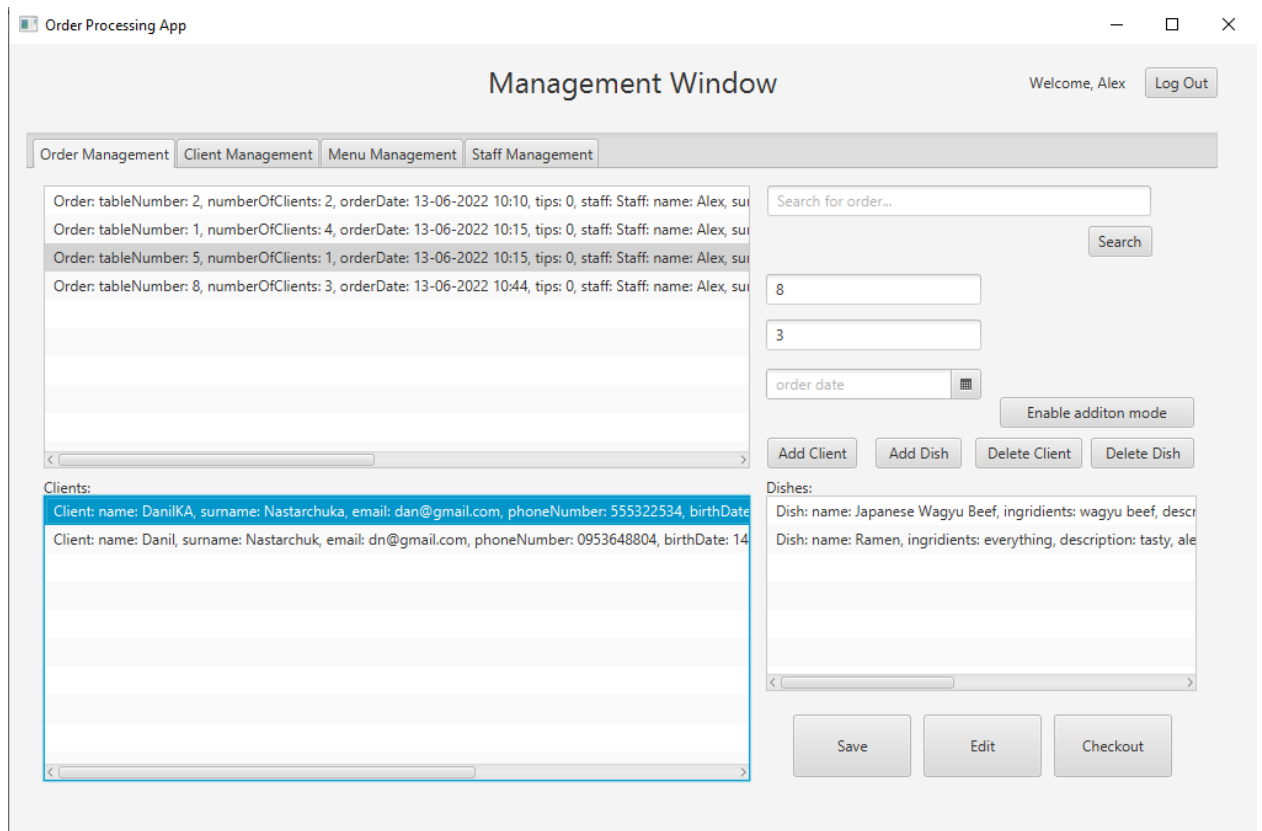


Рисунок 3.17 – Замовлення та клієнт вибрані для перегляду

Для видалення страви обраного клієнта треба вибрати її в списку замовлених клієнтом страв та натиснути кнопку «Delete Dish», а для видалення клієнта з усіма його замовленими стравами, натиснути кнопку «Delete Client»

Щоб розрахувати замовлення треба вибрати його зі списку та натиснути на кнопку «Checkout», після чого з'явиться діалогове вікно для заповнення даними(рис. 3.18).

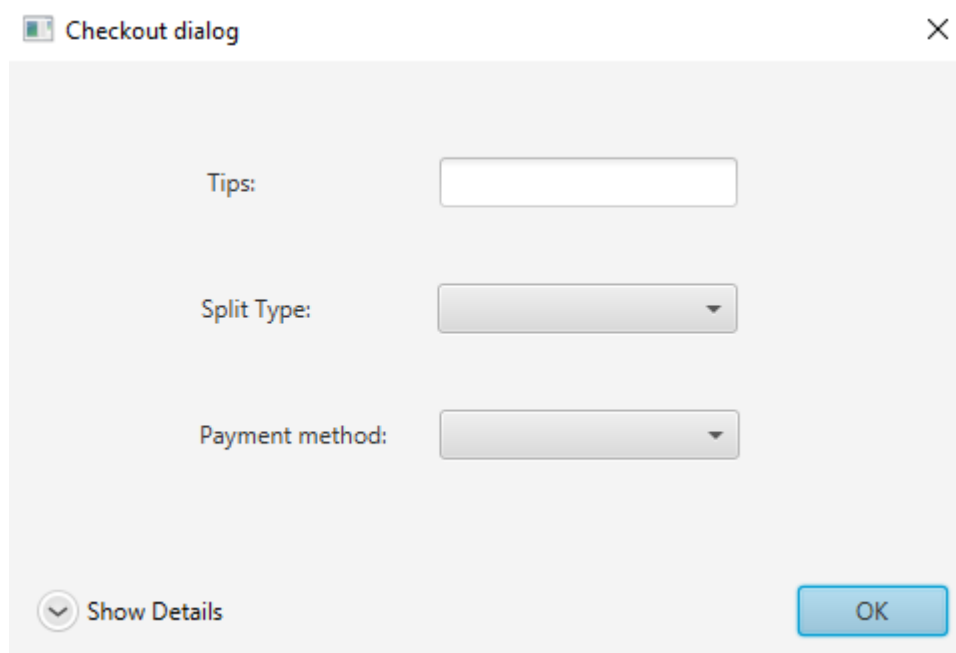


Рисунок 3.18 – Діалогове вікно розрахунку

Після вводу кількості чайових, способу поділення та способу оплати, замовлення буде розділене за вказаними параметрами та буде сформовано один або декілька чеків для друку. Після цього замовлення буде вважатися завершеним.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2022 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ ТА ОБРОБКИ
ЗАМОЛВЕНЬ В РЕСТОРАННОМУ БІЗНЕСІ**

Графічні матеріали

КПІ.IT-8223.045430.07.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Валерій НОВІНСЬКИЙ

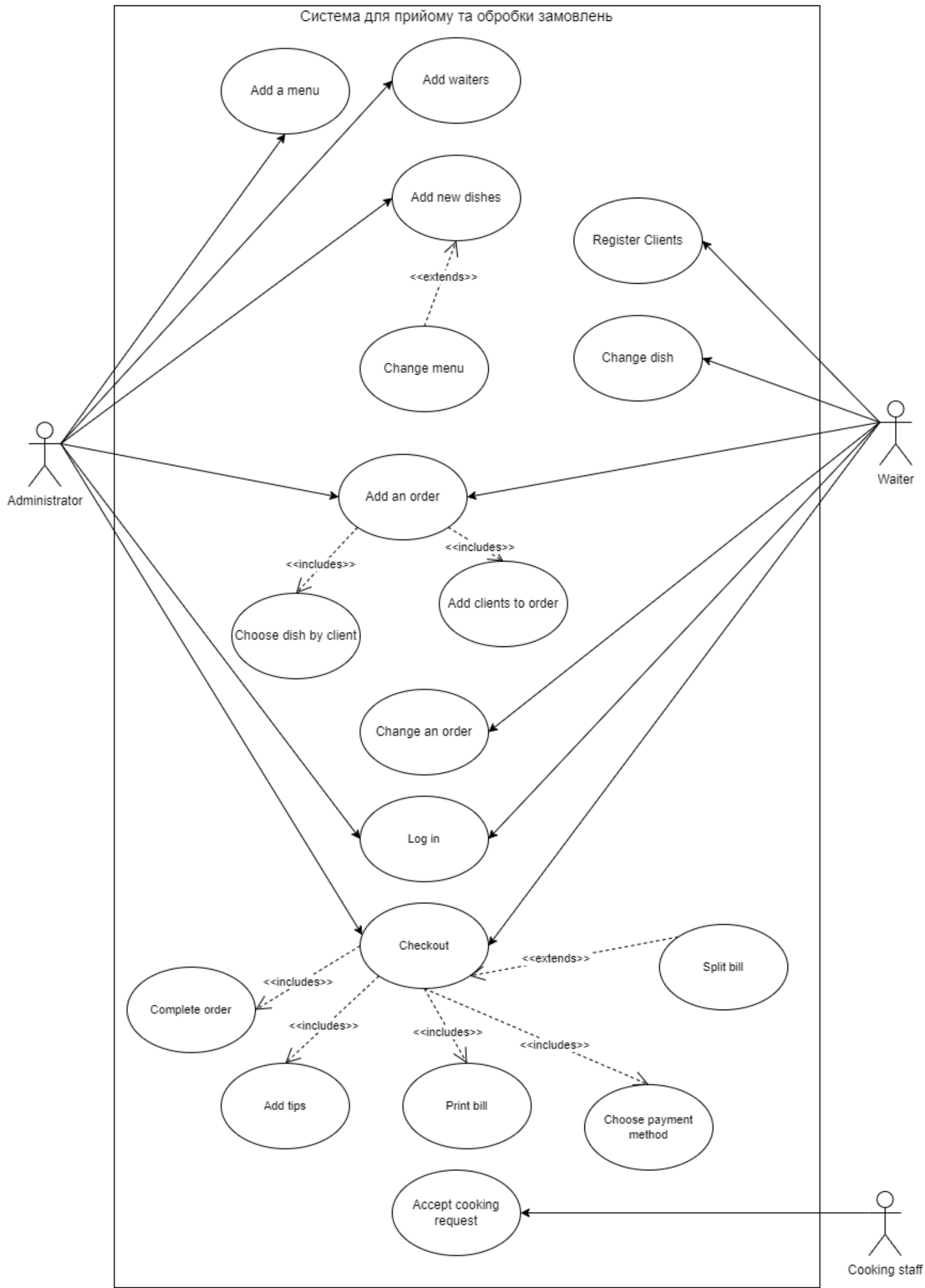
Нормоконтроль:

_____ Валерій НОВІНСЬКИЙ

Виконавець:

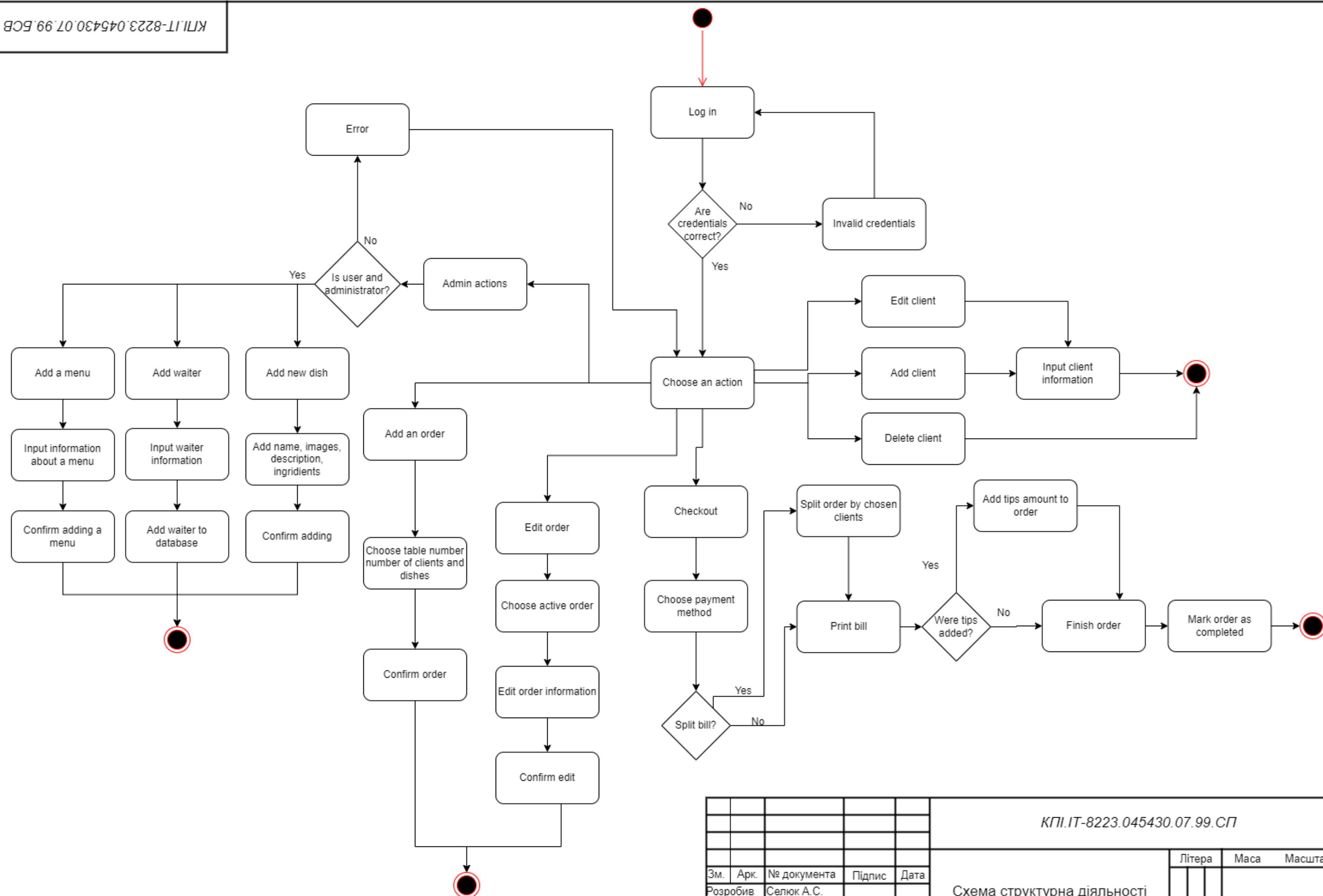
_____ Селюк Александер

Київ – 2022

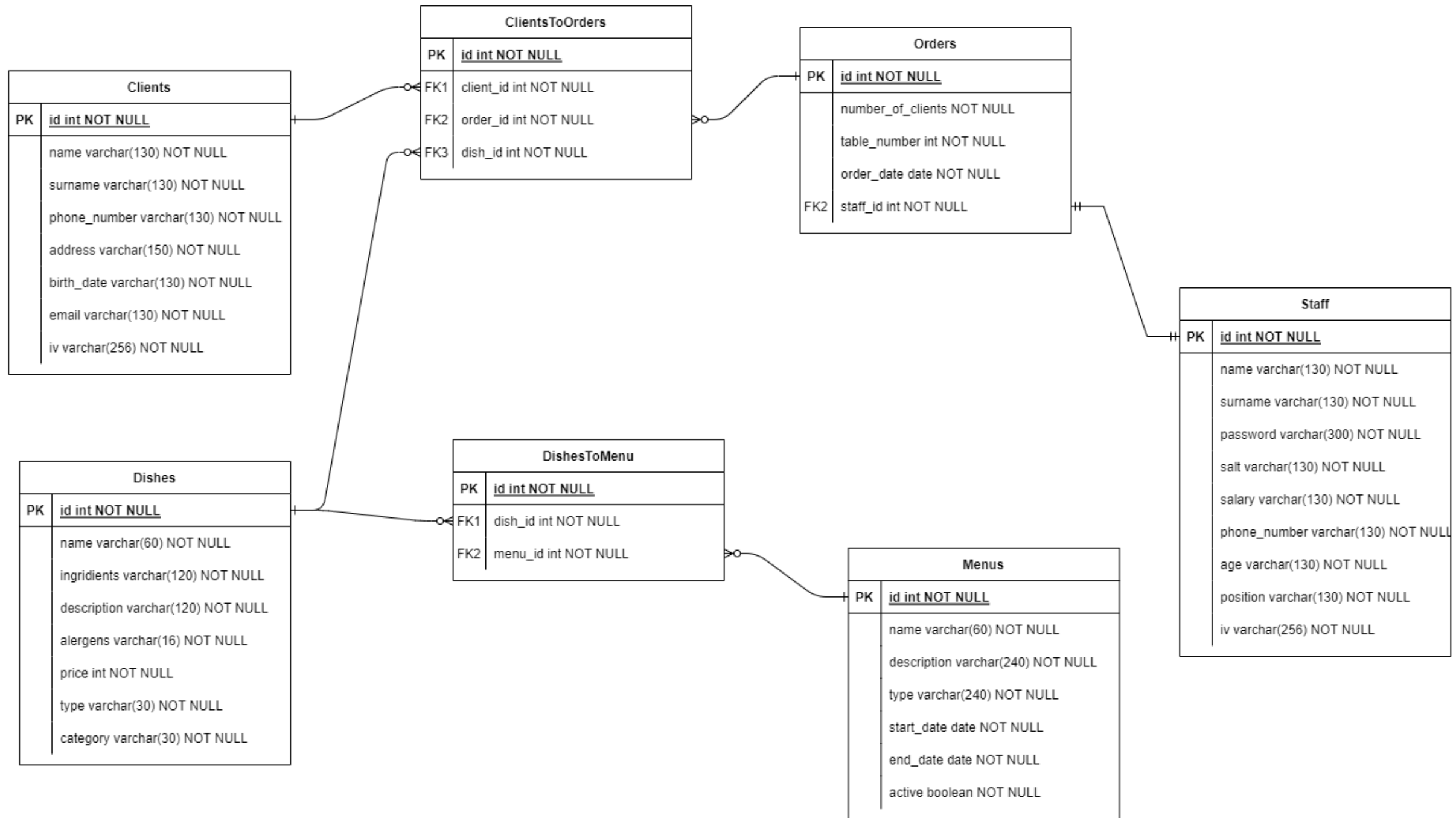


Інв. № підп.	Дата підп.	Інв. № взаєм. підп.	Інв. № дубл.	Підп. дата

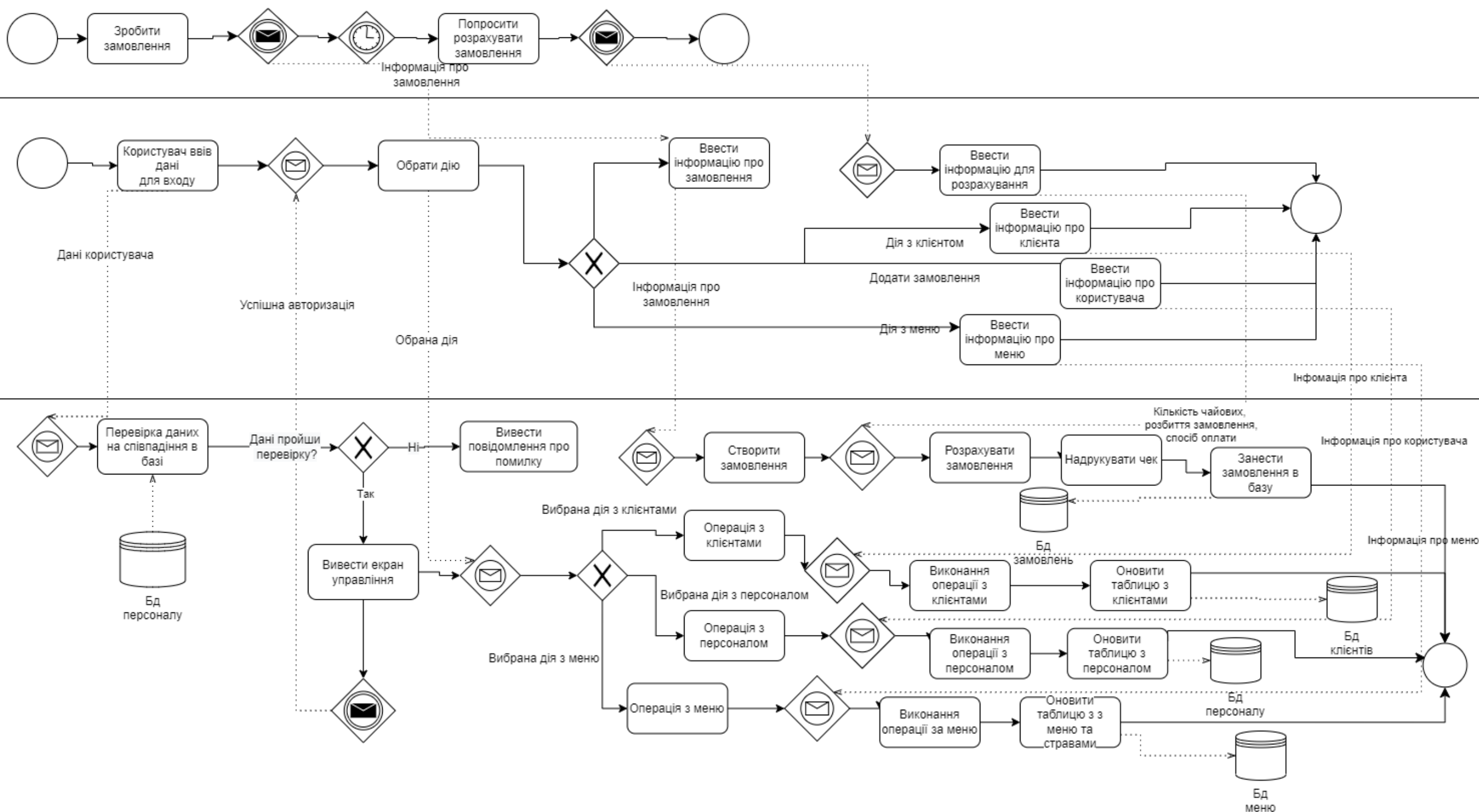
КП.ІТ-8223.045430.06.99.СВВ					Лит.			Арк.	Аркуші
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використання			1	
Розроб.	Селюк А.С.								
Перев.	Новінський В.П.								
Т. Кон.					Аркуш			Аркуші	
Н. Кон.	Ліщук К.І.				Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-82	
Затв.	Новінський В.П.								



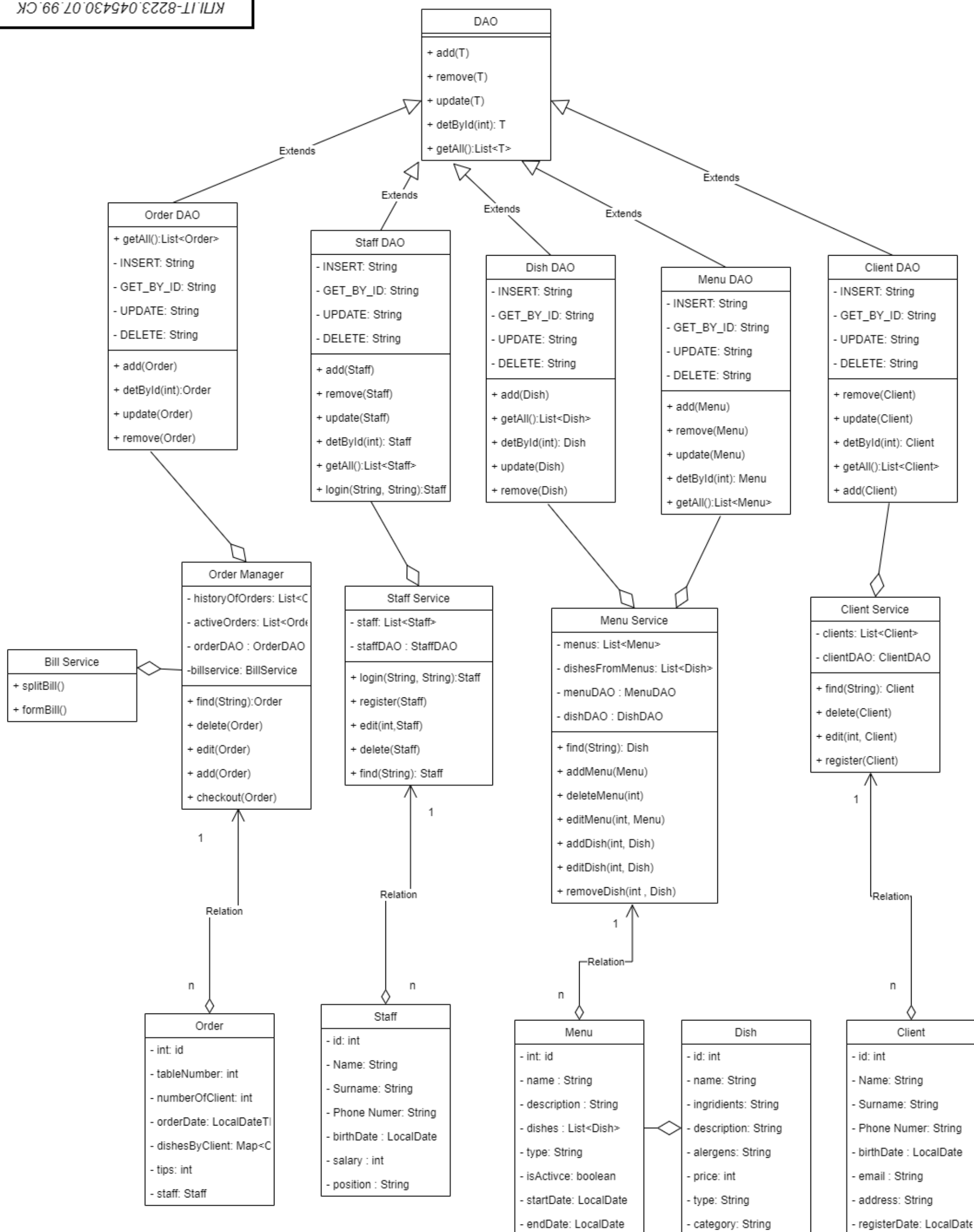
					КПІ.ІТ-8223.045430.07.99.СП				
					Схема структурна діяльності	Літера		Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив		Селюк А.С.							
Перевірив		Новінський В.П.							
Т. кон.						Аркуш	Аркушів		
					Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-82			
Н. кон.		Ліщук К.І.							
Затвердив		Новінський В.П.							



					КПІ.ІТ-8223.045430.08.99.СБД				
					Схема бази даних				
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив		Селюк А.С.							
Перевірів		Новінський В.П.							
Т. кон.									
					Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі			Літера	
Н. кон.		Ліщук К.І.						Маса	Масштаб
Затвердив		Новінський В.П.						Аркуш	Аркушів
					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-82				



					КПІ.ІТ-8223.045430.09.99.СБП				
					Схема бізнес-процесів програмного забезпечення	Літера	Маса	Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив	Селюк А.С.								
Перевірів	Новінський В.П.								
Т. кон.						Аркуш	Аркушів		
					Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі				
Н. кон.	Ліщук К.І.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-82				
Затвердив	Новінський В.П.								



					КПІ.ІТ-8223.045430.10.99.СК				
					Схема структурна класів програмного забезпечення	Лит.		Арк.	Аркуші
Зм.	Арк.	№ докум.	Підп.	Дата					1
Розроб.	Селюк А.С.								
Перев.	Новінський В.П.								
Т. Кон.									
					Програмне забезпечення для прийому та обробки замовлень в ресторанному бізнесі	Аркуш		Аркуші	
Н. Кон.	Ліщук К.І.					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-82			
Затв.	Новінський В.П.								