ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ’ЯЗКУ ім. О.С. ПОПОВА

Навчально-науковий інститут інфокомунікацій та програмної інженерії

Кафедра комп’ютерних наук

**Пояснювальна записка**

до випускної роботи

бакалавра

на тему **РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ РЕСТОРАНУ**

Виконав: студент 4 курсу, групи ПІ-4.2.01

напряму підготовки

6.050103 «Програмна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кукол М.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_Вороной С.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Одеса – 2019 р.

**РЕФЕРАТ**

Текстова частина бакалаврської роботи: 35 с., 11 табл., 28 рис.,   
3 додатки.

Об’єкт дослідження – система автоматизації діяльності ресторану.

Метод дослідження – прикладний.

Мета роботи – розробка та програмна реалізація веб-орієнтованої системи автоматизації діяльності ресторану.

У роботі проаналізовано предметну область веб-орієнтованої системи автоматизації діяльності ресторану та використано такі технології як Python, Django, HTML, CSS, Bootstrap, SQL та PostgreSQL. Для даної системи були поставлені основні завдання, такі як створити, змінити та видалити замовлення, зберегти ці дані у базі даних та відображати їх коли це буде необхідно, зареєструвати постійних клієнтів, змінювати стоп-лист та/або чек-лист, друкувати чек, додавати знижку. Веб-додаток повинен легко масштабуватися для легкого та швидкого додавання додаткового функціоналу.

Ключові слова: РЕСТОРАН, ПОШУК СТРАВИ, ЗАМОВЛЕННЯ, ВЕБ-СЕРВІС, ЗНИЖКА, СТІЛ, СТОП-ЛИСТ, ЧЕК-ЛИСТ, ГІСТЬ, ПЕРСОНАЛ, ОФІЦІАНТ, АДМІНІСТРАТОР

Умови одержання випускної роботи бакалавра: за дозволом проректора з навчальної роботи ОНАЗ ім. О.С. Попова.

**Д О В І Д К А**

кафедри КН про виконану бакалаврську роботу

студента 4 курсу ННІ ІКПІ групи ПІ-4.2.01

Кукола Михайла Сергійовича

на тему Розробка веб-орієнтованої системи автоматизації діяльності ресторану

Висновок нормоконтролера \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (науковий ступінь, вчене звання, посада) | \_\_\_\_\_\_\_\_\_\_  (підпис, дата) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (і. б. прізвище) |

Висновок відповідального за наявність академічного плагіату \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| Відповідальна особа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (науковий ступінь, вчене звання, посада) | \_\_\_\_\_\_\_\_\_\_  (підпис, дата) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (і. б. прізвище) |

**Попередня експертиза (захист)**  \_\_\_\_\_\_\_\_\_\_ \_\_бакалаврської роботи\_ \_\_\_\_\_\_\_\_\_\_

(бакалаврської роботи чи магістерської роботи)

студ. \_\_\_\_\_\_\_\_Кукола М.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ проведена “\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_ 20***\_\_*** р.

(прізвище і.б.)

Висновки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (науковий ступінь, вчене звання, посада, прізвище і.б. )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (науковий ступінь, вчене звання, посада, прізвище і.б.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (науковий ступінь, вчене звання, посада, прізвище і.б.)

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ’ЯЗКУ ім. О.С. ПОПОВА

Навчально-науковий інститутінфокомунікацій та програмної інженерії\_\_\_\_\_\_\_\_\_

Кафедра\_\_\_\_\_\_комп’ютерних наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітньо-кваліфікаційний рівень\_\_\_\_\_бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Напрям підготовки\_\_\_\_\_6.050103 «Програмна інженерія»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

к.т.н., доц.

\_\_\_\_\_\_\_\_\_\_С.М. Вороной

“ \_\_\_\_ ” \_\_\_\_\_\_\_\_ 20\_\_ року

**З А В Д А Н Н Я**

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Куколу Михайлу Сергійовичу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема роботи Розробка веб-орієнтованої системи автоматизації діяльності ресторану \_\_\_\_\_\_\_\_

керівник роботи Вороний Сергій Михайлович, доцент кафедри КН\_\_\_\_\_\_\_\_\_

затверджені наказом закладу вищої освіти від 27.11.2018 р. № 01-08-280

2. Строк подання студентом роботи \_\_\_\_\_\_06.06.2019\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_документація по створенню веб-додатків, веб-ресурси з даними про CRM-системи, веб-ресурси з описом сучасних технологій розробки веб-додатків, база даних\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1) Поняття CRM-систем та їх актуальність \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) Розробка функціональних вимог до системи автоматизації діяльності ресторану \_\_\_\_\_\_\_\_\_\_\_

3) Засоби розробки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4) Розробка бази даних \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5) Розробка програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6) Опис функціювання та тестування системи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Слайд 1 – Загальна характеристика роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Слайд 2 – Результат аналізу конкурентів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Слайд 3 – Розробка Use-Case діаграми \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Слайд 4 – Обґрунтування обраних засобів розробки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Слайд 5 – Схема бази даних \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Слайд 6 – Реалізований функціонал веб-системи ч.1\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Слайд 7 – Реалізований функціонал веб-системи ч.2\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  Консультанта | Підпис, дата | |
| завдання  видав | Завдання  Прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів бакалаврської роботи | Строк виконання етапів роботи | Примітка |
| 1 | Проведення аналізу предметної області, виявлення задач | 27.11 – 27.12 |  |
| 2 | Аналіз конкурентів, виявлення найважливіших завдань | 08.01 – 08.02 |  |
| 3 | Вибір ПЗ для розробки, створення системи, згідно задач | 09.02 – 09.03 |  |
| 4 | Тестування розробленої системи | 10.03 – 10.04 |  |
| 5 | Оформлення пояснювальної записки | 10.04 – 10.05 |  |
| 6 | Підготовка презентації та доповіді | 10.05 – 06.06 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_М.С. Кукол\_\_\_\_

( підпис )

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ С.М. Вороной \_\_\_\_\_\_

( підпис )

ЗМІСТ

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ 10](#_Toc11321778)

[ВСТУП 11](#_Toc11321779)

[1 ПОНЯТТЯ CRM-СИСТЕМ ТА ЇХ АКТУАЛЬНІСТЬ 12](#_Toc11321780)

[1.1 Поняття CRM-систем 12](#_Toc11321781)

[1.2 Класифікація CRM-систем 12](#_Toc11321782)

[1.3 Аналіз існуючих систем автоматизації діяльності закладів громадського харчування 13](#_Toc11321783)

[1.4 Постановка задачі 17](#_Toc11321784)

[2 РОЗРОБКА ФУНКЦІОНАЛЬНИХ ВИМОГ ДО СИСТЕМИ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ РЕСТОРАНУ 18](#_Toc11321785)

[2.1 Ієрархія користувачів системи 18](#_Toc11321786)

[2.2 Розробка Use-Case діаграм 19](#_Toc11321787)

[3 ЗАСОБИ РОЗРОБКИ 22](#_Toc11321788)

[3.1 Система управління базами даних 22](#_Toc11321789)

[3.2 Фреймворк 23](#_Toc11321790)

[3.3 Засоби розробки Front-End частини 27](#_Toc11321791)

[4 РОЗРОБКА БАЗИ ДАНИХ 29](#_Toc11321792)

[4.1 Сутності предметної області 29](#_Toc11321793)

[4.2 Розробка діаграми сутність-зв'язок 31](#_Toc11321794)

[4.3 Фізична реалізація бази даних засобами PostgreSQL 32](#_Toc11321795)

[5 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 35](#_Toc11321796)

[5.1 Розробка Front-End частини 35](#_Toc11321797)

[5.2 MVC або MVT 36](#_Toc11321798)

[5.3 Розробка діаграми розгортки системи 37](#_Toc11321799)

[5.4 Програмна реалізація модулів системи 38](#_Toc11321800)

[6 ОПИС ФУНКЦІЮВАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ 41](#_Toc11321801)

[6.1 Опис функціювання користувача «Повар» 41](#_Toc11321802)

[6.2 Опис функціювання користувача «Офіціант» 42](#_Toc11321803)

[6.3 Опис функціювання користувача «Адміністратор» 43](#_Toc11321804)

[ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ 45](#_Toc11321805)

[ПЕРЕЛІК ПОСИЛАНЬ 46](#_Toc11321806)

[Додаток А](#_Toc11321807) [ПЕРЕЛІК КОПІЙ ДЕМОНСТРАЦІЙНОГО МАТЕРІАЛУ 48](#_Toc11321808)

[Додаток Б](#_Toc11321809) [СПИСОК РОЗВ’ЯЗУВАНИХ ЗАВДАНЬ 52](#_Toc11321810)

[Додаток В](#_Toc11321811) [ПРОГРАМНИЙ КОД МОДУЛІВ 53](#_Toc11321812)

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

1. MVC – Model-View-Controller
2. БД – База даних
3. SQL – Structured Query Language
4. HTML – HyperText Markup Language
5. CSS – Cascading Style Sheets
6. RoR – Ruby on Rails
7. CRM – Customer Relationship Management
8. S – strength
9. W – weakness
10. DOM – Document Object Model
11. MVT – Model-View-Template
12. JSX – JavaScript XML
13. XML – Extensible Markup Language

# ВСТУП

Актуальність даної роботи полягає в тому, що створення системи автоматизації діяльності ресторану дозволить персоналу використовувати свій смартфон чи комп’ютер для створення замовлення клієнтами. Зараз майже усі ресторани перейшли на електронну систему стеження за замовленнями, але вони використовують тільки стаціонарні комп’ютери для реалізації. На даний момент мобільні технології розвиваются швидше за стаціонарні, та набирають високу популярність. У цій же роботі система розрахована для її використання на мобільних пристроях персоналу, а саме на смартфонах та електронних планшетах.

Об'єктом дослідження в даній роботі є система діяльності ресторану.

Предметом дослідження є функціонал, який має реалізувати дана система.

Метою даної роботи є проектування і реалізація інформаційної системи стоврення, зміни замовлень клієнтами ресторану та стеження за цими діями адміністратором чи директором.

Для досягнення даної мети необхідно вирішити такі завдання:

* виконати аналіз предметної області, скласти завдання, які необхідно вирішити;
* розробити архітектуру системи;
* розробити базу даних;
* реалізувати користувацький інтерфейс.

# 1 ПОНЯТТЯ CRM-СИСТЕМ ТА ЇХ АКТУАЛЬНІСТЬ

## 1.1 Поняття CRM-систем

CRM-система – система управління взаємовідносин з клієнтами. CRM-система – це програмне забезпечення прикладного рівня для організацій, передбачене для автоматизації взаємодії з клієнтами. Її основними функціями є покращення обслуговування клієнтів шляхом зберігання інформації про них, зберігання взаємодій з ними, автоматизація аналізу результатів, встановлення та покращення бізнес-процесів та оптимізація маркетингу.

CRM-системи впроваджуються насамперед для збільшення ступеню задоволених клієнтів за рахунок аналізу накопленої інформації про клієнтську поведінку, налаштування інструментів маркетингу, регулювання тарифної політики. Завдяки застосуванню автоматизованої централізованої обробки даних з’являється можливість ефективно і з мінімальною участю співробітників враховувати індивідуальні потреби замовників, а за рахунок оперативності обробки – здійснювати раннє виявлення ризиків і потенційних можливостей.

За даними на 03.04.2018 – більше половини, а саме – 61%, українських компаній замість CRM-систем використовують Microsoft Excel та ще 3% ведуть облік клієнтів на папері [1].

## 1.2 Класифікація CRM-систем

CRM-системи класифікуються за призначенням та за рівнем обробки інформації. За призначенням: керування продажами – інформаційні системи, що використовуються у CRM-маркетингу та CRM-менеджменті, що допомагають автоматизувати деякі продажні функції та функції управління збутом; керування маркетингом – використовуються для автоматизації маркетингових процесів підприємства, перенесення поточних бізнес-процесів компанії в область цифрових сервісів з метою економії трудових і часових ресурсів; керування клієнтським обслуговуванням і колл-центрами – системи з опрацювання звернень абонентів, фіксація і подальша робота зі зверненнями клієнтів.

## 1.3 Аналіз існуючих систем автоматизації діяльності закладів громадського харчування

На сьогоднішній день існує безліч CRM-систем для автоматизації діяльності закладів громадського харчування, але серед них можна виділити найкращі: найоптимізованіші, найшвидші, найвигідніші та маючі кращі користувальницькі інтерфейси.

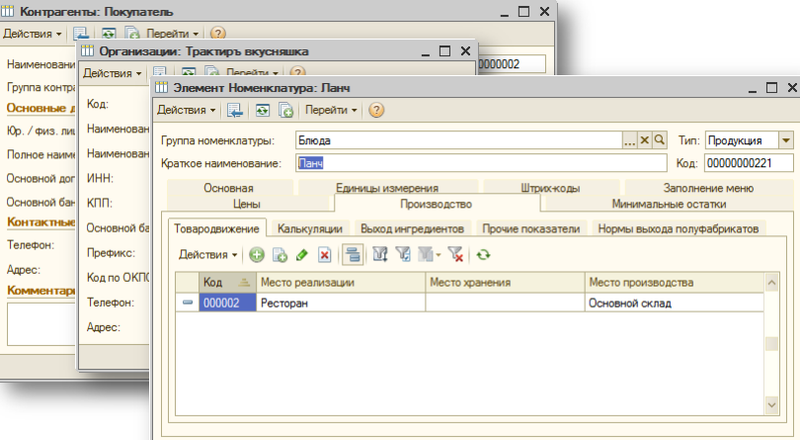
«Трактиръ» – одна з найпопулярніших CRM-систем у Росії, розробка групи компаній «СофтБаланс»[2]. Розроблюється з 1997 року, компанії мають більш 10 000 корпоративних проектів на застарілій платформі «1С:Предприятие». Інтерфейс користувача має хоч і старий, але приємний для очей вигляд, стилізований під систему управління базами даних Microsoft Access. Інтерфейс «Трактиръ» зображено на рисунку 1.1.

Рисунок 1.1 – Користувальницький інтерфейс «Трактиръ»

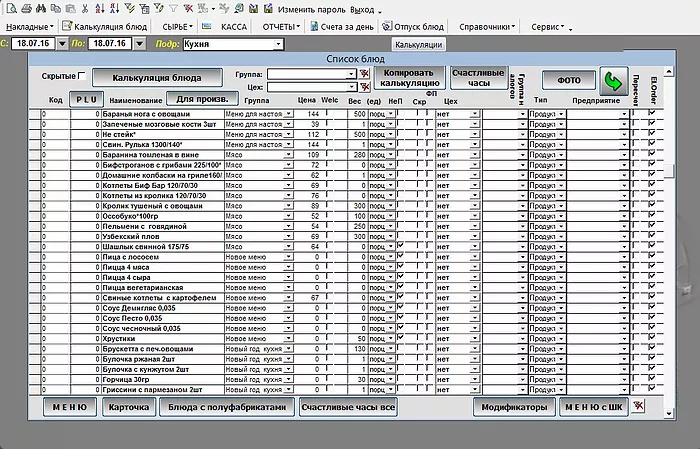
«BarBo$$» – найпопулярніша CRM-система для ресторанів, готелів та фітнес-клубів і спортзалів в Україні, Республіці Молдові та у Придністровській Молдавській Республіці [3]. У 90% випадків при вході у заклад загального харчування в Україні клієнт може побачити від одного до декількох пристроїв, працюючих на системі «BarBo$$». Вона має модуль бухгалтера, модуль офіціанта та мобільний-додаток «BarBo$$ Dashboard», що дозволяє отримувати десятки різних звітів закладів, а також отримувати онлайн-статистику закладу (кількість клієнтів, поточна виручка та ін.). Користувальницький інтерфейс має не самий сучасний вигляд, але зручний для користувачів. Над ним була проведена чимала робота UI/UX-дизайнерів. Інтерфейс користувача можна побачити на рисунках 1.2 та 1.3.

Рисунок 1.2 – Інтерфейс модуля бухгалтера системи «BarBo$$»

Рисунок 1.3 – Інтерфейс модуля офіціанта системи «BarBo$$»

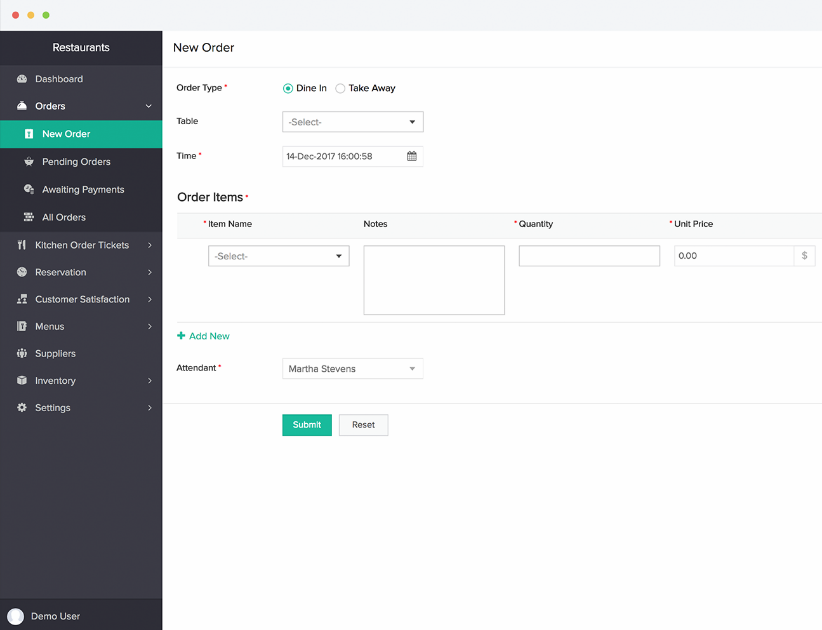
«Zoho CRM» – одна за найпопулярніших CRM-систем у країнах Європи [4]. Вона має сучасний користувальницький інтерфейс, мобільний додаток та сучасну оптимізацію. Багатоканальна підтримка у «Zoho CRM» дозволяє користувачеві спілкуватися з людьми по телефону, через чат, електронну пошту, соціальні мережі та, навіть, особисто. Вона має онлайн-статистику відвідувачів та аналітику електронної пошти, щоб узнати, що бачать їх клієнти, та знайти можливості для взаємодії. Великим недоліком «Zoho CRM» є відсутність інтернаціональності – система має лише англійську мову. На рисунку 1.4 зображено користувальницький інтерфейс «Zoho CRM».

Рисунок 1.4 – Інтерфейс користувача «Zoho CRM»

Таблиця 1.1 – Оцінка критеріїв функціональності досліджуваних сервісів

| **Функціонал** | **Розглянуті сервіси** | | |
| --- | --- | --- | --- |
| BarBo$$ | Zoho CRM | Трактиръ |
| S | Перекидання окремих елементів на столі іншому столові (користувачеві) | Примітки для персоналу | Прийом бронювання столу через сайт |
| W | Система знижок | – | Чек-лист та стоп-лист |
| Додавання окремого елемента столові | + | + | + |
| Перекидання окремих елементів на столі іншому столові (користувачеві) | + | – | + |
| Скасування замовлення | + | + | – |
| Прийом бронювання столу через сайт | – | – | + |
| Врахування непередбачених обставин | – | – | + |
| Примітки для персоналу | – | + | – |
| Система знижок | – | + | + |
| Чек-лист | + | + | – |
| Стоп-лист | + | + | – |
| Додавання та видалення окремих атрибутів страви для столу | + | + | + |
| Вказівка виносу окремих страв | + | – | – |
| Рейтинг персоналу | – | – | + |

## 1.4 Постановка задачі

Основною задачею проекту є розробка веб-додатку автоматизації діяльності ресторану. Вона поділена на декілька підзадач, що мають бути виконані по черзі:

1. обрати тип бази даних: реляційна чи нереляційна;
2. обрати найкращу та найоптимізованішу систему управління базами даних для даного проекту;
3. обрати мову програмування та фреймворк для розробки веб-додатку;
4. обрати засоби для розробки візуальної частини (інтерфейсу користувача);
5. розробити базу даних (виділити сутності, розробити діаграму сутність-зв'язок);
6. розробити програмне забезпечення (інтерфейс користувача, діаграму розгортки, діаграму моделі проектування, модулі системи);
7. описати функціювання системи для кожного з користувачів;
8. зробити висновки і рекомендації та описати їх.

Серед основних задач, що має виконувати веб-додаток виділено такі:

1. створювання замовлення та його модифікація (додавання страв, атрибутів до страв, їх видалення);
2. додавання, видалення та модифікація даних постійних клієнтів;
3. прийом передзамовлення, створеного клієнтом, на певні дату і час;
4. модифікація даних користувачів системи (офіціантів, поварів та адміністраторів) та їх реєстрація чи видалення;
5. розрахунок столу (друк чеку, внесення знижки чи додаткової плати);
6. стеження офіціантів за їх чайовими за необхідністю (модифікація суми та розподіл між кількістю персон);
7. перегляд замовлень поварами и зазначення страв готовими до виносу.

# 2 **РОЗРОБКА ФУНКЦІОНАЛЬНИХ ВИМОГ ДО СИСТЕМИ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ РЕСТОРАНУ**

## 2.1 Ієрархія користувачів системи

В результаті дослідження предметної області та аналізу конкурентів були виділені такі користувачі системи:

1. Базовий користувач. Узагальнений тип користувача, що об’єднує інших користувачів за спільними функціями. Базовий користувач може модифікувати чек-лист та стоп-лист (додавати страви та видаляти їх) і переглядати замовлення;
2. Адміністратор. Одноразово зареєстрований користувач при створенні системи. Адміністратор має усі ті функції, що і базовий користувач. Він може модифікувати користувачів та постійних клієнтів (створювати нових, видаляти існуючих, модифікувати їх дані), видаляти страву або набір страв з існуючого замовлення, розраховувати стіл (друкувати чек, вказувати постійного клієнта та додавати додаткової плати, наприклад, за обслуговування) та приймати передзамовлення (додавати набір страв до столу на указані дату і час та резервування столу);
3. Повар. Зареєстрований адміністратором користувач. Повар має усі ті функції, що і базовий користувач. Також він може позначити страву як готову до виносу;
4. Офіціант. Зареєстрований адміністратором користувач. Має усі ті функції, що і базовий користувач. Офіціант також може: створювати замовлення (додавати набір страв до столу, зазначати номер виносу для набору страв, зазначати кількість посуду для набору страв), стежити за своїми та/або чужими чайовими (модифікувати суму, ділити між зазначеною кількістю офіціантів), розраховувати стіл та приймати передзамовлення.

Повний список розв’язуваних завдань у вигляді таблиці можна продивитися у Додатку Б.

## 2.2 Розробка Use-Case діаграм

У зв’язку з тим, що у системі передбачено чотири типи користувачів, були розроблені окремі Use-Case діаграми [5] для кожного з них.

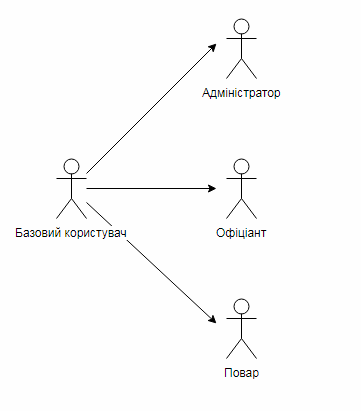
Use-Case діаграма ієрархії користувачів зображена на рисунку 2.1.

Рисунок 2.1 – Use-Case діаграма ієрархії користувачів

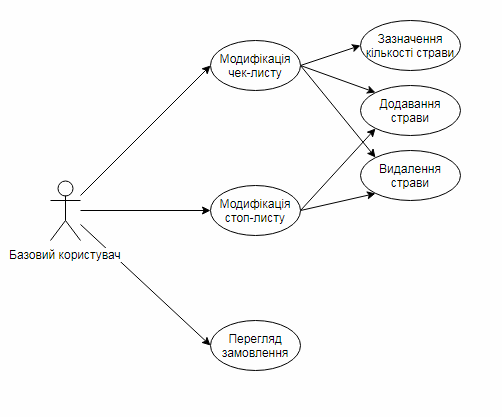
Use-Case діаграма для базового користувача зображена на рисунку 2.2.

Рисунок 2.2 – Use-Case діаграма для базового користувача

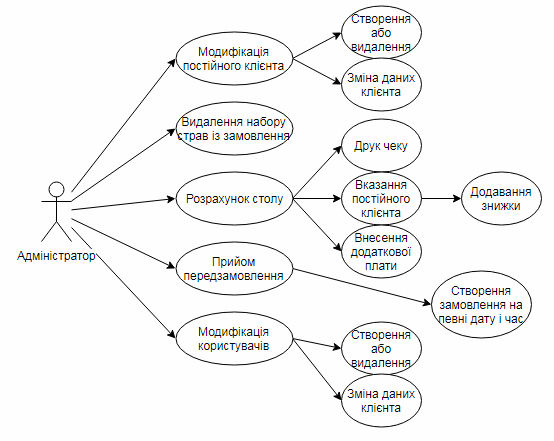
Use-Case діаграма для користувача «адміністратор» зображена на рисунку 2.3.

Рисунок 2.3 – Use-Case діаграма для користувача «адміністратор»

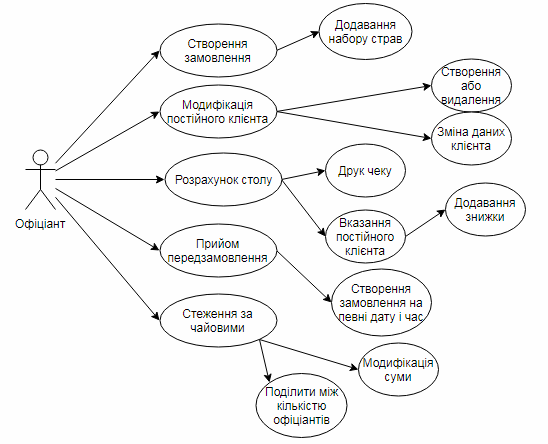
Use-Case діаграма для користувача «офіціант» зображена на рисунку 2.4.

Рисунок 2.4 – Use-Case діаграма для користувача «офіціант»

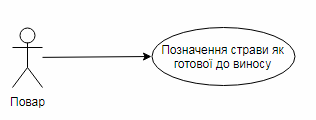
Use-Case діаграма для користувача «повар» зображена на рисунку 2.5.

Рисунок 2.5 – Use-Case діаграма для користувача «повар»

# 3 ЗАСОБИ РОЗРОБКИ

Система автоматизації діяльності ресторану розробляється у вигляді веб-додатку. Для цього необхідні такі компоненти як: база даних, інструменти для розробки серверної частини додатку та створення дизайну та навігації – Front-End частини додатку.

Необхідні обрані мови програмування, фреймворки, мова розмітки, мова бази даних та система управління базами даних описані у наступних трьох підрозділах.

## 3.1 Система управління базами даних

Існує два типи баз даних: реляційні та нереляційні або постреляційні.

Реляційні бази даних – бази даних, основані на реляційній моделі даних. Реляційна модель даних – це логічна модель даних, котра є додатком до задач обробки інформації (даних) з використанням таких розділів як математика, теорія множин, нормалізації та логіка першого порядку. Реляційна модель даних містить компоненти аспекту цілісності, аспекту маніпулювання даними та структурного аспекту. Теорія основана на математичному понятті відносин. Метою нормалізації реляційної бази даних є усунення недоліків структури бази даних, що призводять до надмірності, яка, в свою чергу, потенційно призводить до різних аномалій і порушень цілісності даних. SQL – декларативна мова програмування, застосовувана для створення, модифікації та управління даними в реляційній базі даних, керованої відповідною системою управління базами даних [6].

У нереляційних базах даних немає жорсткої структури і не використовується перетин таблиць, за великим рахунком вони взагалі не мають таблиць. Нереляційні бази даних поділяють на кілька типів, які визначаються залежністю від їх масштабованості, моделями даних і запитів, а також системою зберігання даних. Єдину класифікацію ще не розробили, але розрізняють наступні моделі даних:

1. графові. В даних моделях для уявлення інформації використовуються вершини та ребра графу;
2. колонково-орієнтовна. Ця модель зберігає дані в стовпцях замість звичного зберігання у рядках, що є вигідним для різного роду архівів інформації та каталогів, в яких велика частина обчислень відбувається над подібними вибірками даних;
3. документо-орієнотовна. У даній моделі кожен запис зберігається у вигляді окремого документа, що має власний набір полів, що відрізняється від документа до документа;
4. ключ – значення. Дану модель можна уявити у вигляді величезної таблиці, у якої в кожній клітинці зберігаються дані довільного типу, їх структура нічим не обмежується. Кожному значенню присвоєно певний код – «ключ», за яким його можна знайти. Усі дані, які відносяться до конкретного об’єкта зберігають в одному місці, тому при запитах немає необхідності звертатися до різних таблиць.

У даному веб-додатку вкрай необхідна цілісність даних та необхідна заснована на усталених стандартах технологія, використовуючи яку можна розраховувати на великий досвід розробників та технічку підтримку. Тому для розробки даного проекту обрано реляційну мову опису бази даних SQL.

SQL має великий набір систем управління базами даних. Серед них найоптимізованіші та найпопулярніші (популярність має значення тому, що люди їх обирають через великий досвід розробників та найкращі технічну підтримку та підтримку різних версій продуктів): MySQL, PostgreSQL, Oracle та DB2.

Для розробки веб-додатку для автоматизації діяльності ресторану обрано PostgreSQL тому, що вона безкоштовна, на відміну від Oracle, продуктивніша за DB2, захищена від втрат даних на відміну від MySQL. PostgreSQL також має такий недолік низька продуктивність при обробці величезних об’ємах даних, але у ресторану таких об’ємів даних бути не може, якщо, звісно, ресторан не займає таку площу, як середньостатичне село. Також PostgreSQL має зручний візуальний інтерфейс у вигляді програмного забезпечення – PgAdmin.

## 3.2 Фреймворк

За даними «Jetruby» на 20.03.2018 найкращими фреймворками для розробки веб-додатків є Front-End фреймворк AngularJS, Ruby on Rails, найпопулярніший Python-фреймворк для розробки веб-додатків Django та один з найкращих PHP-фреймворків Laravel [7].

AngularJS – JavaScript-фреймворк, що можна узнати з назви, з відкритим вихідним кодом. AngularJS використовується для розробки односторінкових веб-додатків [8]. Фреймворк працює з HTML, що містить додаткові користувальницькі атрибути, що описуються директивами, і зв’язує введення та виведення області сторінки з моделлю, яка представляє собою звичайні змінні JavaScript. AngularJS – це набір інструментів для побудови фреймворка, найбільш відповідного для створення веб-додатку. Він розширюваний і відмінно взаємодіє з іншими бібліотеками. Будь-яка особливість може бути змінена або замінена відповідно за унікальним шляхом розвитку і потребами певного додатку. На відміну від інших фреймворків, немає необхідності успадковуватися від власних типів; обертати модель методами доступу. Тільки JavaScript. Це дозволяє легко тестувати код, обслуговувати, повторно використовувати і звільнити його від шаблонів. На рисунку 3.1 зображено приклад коду AngularJS.

Рисунок 3.1 – Приклад коду AngularJS та HTML.

AngularJS виділяється серед конкурентів через спрощену двосторонню прив’язку даних; він входить у пакет «NO MEAN», який також включає MongoDB, Express.js і Node.js, тому він дозволяє керувати Front-End і Back-End проекту за допомогою JavaScript; він не пристосований обробляти маніпуляції з DOM з великою кількістю даних, так як покладається на «брудні перевірки» для управління змінними DOM (будь-яка зміна змінних тягне оновлення DOM); AngularJS також не підтримує високонавантажені галереї фото, тому розробляти фото-хостинг на ньому не варто.

Ruby on Rails – фреймворк написаний на мові програмування Ruby, котрий, як і вищезгаданий AngularJS, реалізує архітектурний шаблон MVC для веб-додатків [9]. Розповсюджується за ліцензією MIT [10]. На рисунку 3.2 зображено приклад коду Ruby on Rails.

Рисунок 3.2 – Налаштування БД MySQL на Ruby on Rails

RoR має більше недоліків за переваги. Ruby on Rails складний у вивченні, в ньому багато підводних каменів, в основному пов’язаних з налаштуванням серверу і нюансами, які потрібно один раз подолати, і більше з ними проблем не буде. Він має досить мало документації російською мовою, про українську не треба навіть згадувати. Є багато хостингів для RoR, але вони потребують великої плати за розміщення (від 3$, у той час, коли за PHP потребують від 1$).

Краща альтернатива Ruby on Rails – це Laravel, написаний на PHP. Фреймворк Laravel також реалізує шаблон MVC, має ліцензію MIT.

Laravel – фреймворк вищого за RoR рівня, котрий містить у собі такі компоненти:

1. пакети для створювання та підключення модулів у форматі Composer;
2. вбудована ORM – концепція, що пов’язує бази даних з концепціями об’єктно-орієнтованого програмування;
3. REST-контролери. Додатковий шар для обробки POST- і GET-запросів;
4. міграції – система управління версіями бази даних.

Вище перераховано не усі компоненти Laravel, а лише самі необхідні для спрощення розробки веб-додатків. Далі, на рисунку 3.3, можна побачити приклад коду фреймворка Laravel.

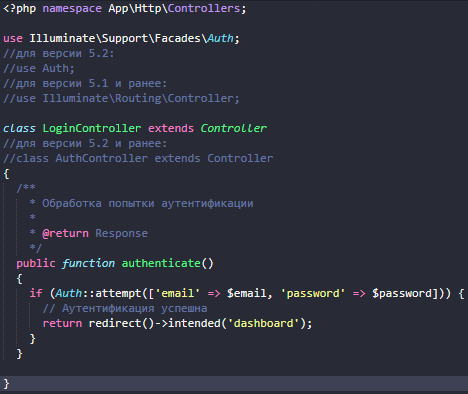


Рисунок 3.3 – Приклад ручної автентифікації у Laravel.

Django – це безкоштовний глобальний веб-фреймворк високого рівня з відкритим вихідним кодом, написаний на мові програмування Python [11]. Він сприяє високій швидкості розробки веб-додатку і чистому та прагматичному дизайну. Django розроблений досвідченими розробниками, він дозволяє сконцентруватися на розробці додатку без необхідності «створювати велосипед». Він розроблюється з 2005 року та підтримується донині, це означає, що він старший за інші фреймворки та більш розвинений. Django також ідеально підходить для розробки, якщо у проекті використовуються бази даних PostgreSQL. Для роботи з базами даних Django має вбудовану ORM-систему, що дозволяє оголошувати таблиці бази даних як класи, її стовпці як атрибути класу, а кортежі як об’єкти класу, що значно полегшує розробку додатку та підвищує швидкість розробки. За необхідністю можна не використовувати ORM-систему, а виконувати «сирі» запити [12]. У ньому є вбудований веб-сервер для запуску на різних пристроях: від локального пристрою до хостингу. Вбудований веб-сервер автоматично виявляє зміни у файлах проекту, та перезапускається для їх відображення. На стадії розробки сервер виводить ймовірні помилки як у терміналі, так і на самій сторінці веб-додатку. На стадії продакшену виведення помилок треба відключати, тому що зловмисник може скористатися вразливістю додатку та використати інформацію у базі даних у гірших цілях.

Django – це «all inclusive» у світі веб-розробки. Він включає в себе різні додаткові опції для допомоги з картами сайту, автентифікацією користувачів, адмініструванням контенту, RSS-канали та багато іншого. Кожна з них надає істотну допомогу в процес веб-розробки.

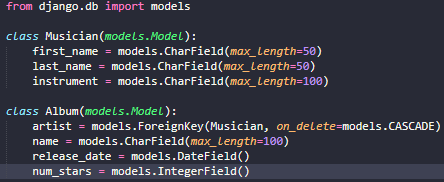
Приклад створення таблиць у базі даних за допомогою ORM Django зображено на рисунку 3.4.

Рисунок 3.4 – Створення таблиць бази даних у Django

Розібравши усі переваги і недоліки вищеперерахованих веб-фреймворків, вирішено обрати найкращий для розробки веб-додатку автоматизації діяльності ресторану. Цим фреймворком став Django останньої на сьогоднішній день версії 2.2.

## 3.3 Засоби розробки Front-End частини

Існує безліч фреймворків для роботи з Front-End частиною веб-додатків. Більшість з них написані мовою програмування JavaScript для спрощення розробки великих проектів, що працюють з величезними базами даних. Нижче приведено кілька прикладів:

1. AngularJS, котрий був описаний раніше. Він використовується одночасно для розробки і Back-End, і Front-End частини веб-додатків. Детальніше він описаний у підрозділі 3.2;
2. React.js. Оснований на описі усіх частин коду як компоненти. React-компоненти реалізують метод render(), який приймає вхідні дані і повертає якісь дані чи інший компонент для виведення.   
   Цей фреймворк підтримується корпораціями Facebook та Instagram, це означає, що він, як і попередні, регулярно оновлюється та написаний з використанням досвіду кращих у світі розробників. React – це ідеальний вибір для невеликих односторінкових проектів [13];
3. Vue.js. Новий, але підкорив серця тисяч розробників по усьому світі через свою швидкість при обробці великих даних. Vue також використовується для розробки односторінкових веб-додатків, але, на відміну від React, він здатен на обробку великої кількості даних без оновлення DOM, та використовує «чистий» JavaScript у той час як React використовує JSX (JSX – це «синтаксичний цукор» для опису JavaScript-коду [14]) з додавання сторонніх бібліотек.

Для розробки Front-End частини веб-додатку, з якою безпосередньо працює користувач, не було обраного жодного з вищеперелічених фреймворків, тому що, по-перше, даний веб-додаток не має бути односторінковим, та, по-друге, усі ці фреймворки сповільнять роботу додатку, який, у свою чергу, має бути максимально швидким. Тому для Front-End частини обрано лише HTML останньої на сьогоднішній день п’ятої версії та CSS-фреймворк Bootstrap.

Bootstrap – це CSS-фреймворк, що складається з шаблонів. Він майже не включає в себе додаткові JavaScript-бібліотеки, а лише дозволяє використовувати відкритий CSS-код шаблонів. Простіше кажучи, він не несе додаткових функцій, а лише прискорює час розробки та надає можливість створювати приємний для очей користувача інтерфейс.

# 4 РОЗРОБКА БАЗИ ДАНИХ

## 4.1 Сутності предметної області

У ході розробки бази даних веб-додатку для автоматизації діяльності ресторану виявлено такі сутності:

1. Користувач. Користувачі системи поділяються на такі типи: адміністратори, офіціанти та повара. Вони характеризуються за ім’ям та типом користувача;
2. Інгредієнт. Інгредієнт – це складова частина страви, що характеризується лише за назвою;
3. Страва. Страва складається з інгредієнтів. Характеризується за назвою, типом, масою та ціною;
4. Стіл. Кожен ресторан має певний набір столів. Стіл може мати необмежену кількість страв. Він може характеризуватися за номером, за клієнтом та мати офіціанта, що обслуговує цей стіл;
5. Постійний клієнт (гість). Постійний клієнт – зареєстрована у базі сутність, що не є користувачем. Клієнт характеризується за ім’ям та знижкою.

Таблиця 4.1 – Опис сутності «Користувач» та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Обмеження |
| user | | |
| id | Ідентифікатор користувача | PK |
| password | Захеширований пароль користувача |  |
| last\_login | Дата и час останньої автентифікації користувача |  |
| is\_superuser | Чи є користувач "суперкористувачем" |  |
| is\_staff | Чи є користувач доступ до admin-панелі |  |
| is\_active | Чи є користувач активним (не ботом) |  |
| username | Скорочене ім'я користувача | U |
| first\_name | Ім'я користувача |  |
| last\_name | Прізвище користувача |  |
| email | Адреса електронної пошти користувача | U |
| date\_joined | Дата реєстрації користувача |  |
| type | Тип користувача: адміністратор, офіціант чи повар |  |

Таблиця 4.2 – Опис сутності «Інгредієнт» та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Обмеження |
| ingridient | | |
| id | Ідентифікатор інгредієнта | PK |
| name | Повна назва інгредієнта | U |

Таблиця 4.3 – Опис сутності «Страва» та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Обмеження |
| dish | | |
| id | Ідентифікатор страви | PK |
| name | Повна назва страви | U |
| type | Тип страви |  |
| calorie | Калорійність страви |  |
| ingridients | Інгредієнти, з яких складається | N |
| price | Ціна страви |  |
| mass | Маса страви |  |

Таблиця 4.4 – Опис сутності «Стіл» та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Обмеження |
| table | | |
| id | Ідентифікатор столу | PK |
| number | Номер столу | U |
| waiter | Офіціант, що обслуговує | FK, N |
| available | Чи вільний стіл |  |
| bill | Сума чеку |  |
| dishes | Страви, що має стіл | N |

Таблиця 4.5 – Опис сутності «Постійний клієнт» та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Обмеження |
| guest | | |
| id | Ідентифікатор клієнта | PK |
| name | Повне ім'я клієнта | U |
| discount | Знижка клієнта |  |

## 4.2 Розробка діаграми сутність-зв'язок

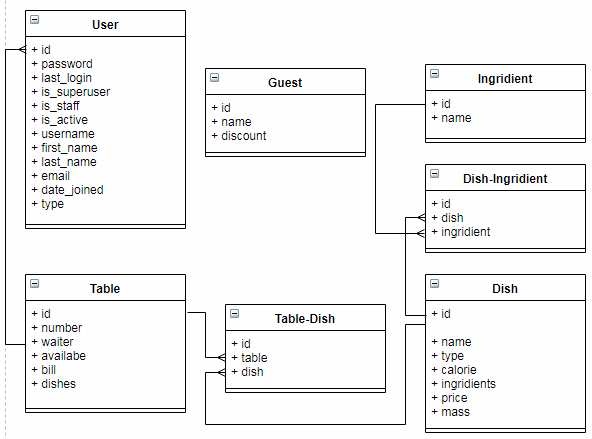
Після виявлення сутностей проекту у базі даних веб-додатку автоматизації діяльності ресторану було розроблено діаграму сутність-зв'язок, котру зображено на рисунку 4.1.

Рисунок 4.1 – Діаграма сутність-зв'язок

Між сутністю «Користувач» та сутністю «Стіл» існує умовний зв'язок один до багатьох, що реалізується додаванням первинного ключа як зовнішній ключ у таблицю N-пов’язаної сутності.

Між сутностями «Страва» та «Інгредієнт» і «Стіл» та «Страва» існує умовний зв'язок багато до багатьох, що реалізується додаванням транзитивних сутностей «Страва-Інгредієнт» та «Стіл-Страва» з реалізацією зв’язку один до багатьох.

Сутність «Постійній клієнт» існує без зв’язків у вигляді довідника.

## 4.3 Фізична реалізація бази даних засобами PostgreSQL

Усі об’єкти бази даних (таблиці, хранимі процедури, тригери, представлення) створені за допомогою використання веб-фреймворка Django і відповідних запросів.

Об’єкти бази даних:

1. rest\_dish – таблиця, що містить дані про страву. Сутністю цієї таблиці є приведена у пункті 4.1 сутність «Страва»;

Таблиця 4.6 – Опис таблиці rest\_dish та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Тип даних |
| id | Ідентифікатор страви | int |
| name | Повна назва страви | string |
| type | Тип страви | string |
| calorie | Калорійність страви | float |
| ingridients | Інгредієнти, з яких складається | int |
| price | Ціна страви | float |
| mass | Маса страви | float |

1. rest\_dishtype – таблиця-довідник, що містить типи страв;
2. rest\_guest – таблиця-довідник, що містить дані про постійних клієнтів. Сутністю цієї таблиці є приведена у пункті 4.1 сутність «Постійний клієнт»;

Таблиця 4.7 – Опис таблиці rest\_guest та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Тип даних |
| id | Ідентифікатор клієнта | int |
| name | Повне ім'я клієнта | string |
| discount | Знижка клієнта | int |

1. rest\_ingridient – таблиця-довідник, що містить дані про існуючі інгредієнти. Сутністю цієї таблиці є приведена у пункті 4.1 сутність «Інгредієнт»;

Таблиця 4.8 – Опис таблиці rest\_ingridient та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Тип даних |
| id | Ідентифікатор інгредієнта | int |
| name | Повна назва інгредієнта | string |

1. rest\_stoplistdish – таблиця, що містить перелік страв, що знаходяться у стоп-листу;
2. rest\_table – таблиця, що містить дані про столи. Сутністю цієї таблиці є приведена у пункті 4.1 сутність «Стіл»;

Таблиця 4.9 – Опис таблиці rest\_table та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Тип даних |
| id | Ідентифікатор столу | int |
| number | Номер столу | int |
| waiter | Офіціант, що обслуговує | int |
| available | Чи вільний стіл | boolean |
| bill | Сума чеку | float |
| dishes | Страви, що має стіл | int |

1. rest\_tabledishes – таблиця, що містить дані про те, у яких столів є які страви. Усі поля обов’язкові до заповнення;
2. rest\_dish\_ingridients – таблиця, що містить дані про те, які інгредієнти входять до складу конкретних страв. Усі поля обов’язкові до заповнення;
3. auth\_user – таблиця, що має дані про користувачів системи. Сутністю цієї таблиці є приведена у пункті 4.1 сутність «Користувач».

Таблиця 4.10 – Таблиця auth\_user та її атрибути

|  |  |  |
| --- | --- | --- |
| Поле | Опис | Тип даних |
| id | Ідентифікатор користувача | int |
| password | Захеширований пароль користувача | string |
| last\_login | Дата и час останньої автентифікації користувача | date |
| is\_superuser | Чи є користувач "суперкористувачем" | boolean |
| is\_staff | Чи є користувач доступ до admin-панелі | boolean |
| is\_active | Чи є користувач активним (не ботом) | boolean |
| username | Скорочене ім'я користувача | string |
| first\_name | Ім'я користувача | string |
| last\_name | Прізвище користувача | string |
| email | Адреса електронної пошти користувача | string |
| date\_joined | Дата реєстрації користувача | date |
| type | Тип користувача: адміністратор, офіціант чи повар | string |

Усі вищеперелічені об’єкти бази даних та зберігаються в схемі public. Власником цієї схеми, а також усієї бази даних, є користувач kukol.

# 5 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 5.1 Розробка Front-End частини

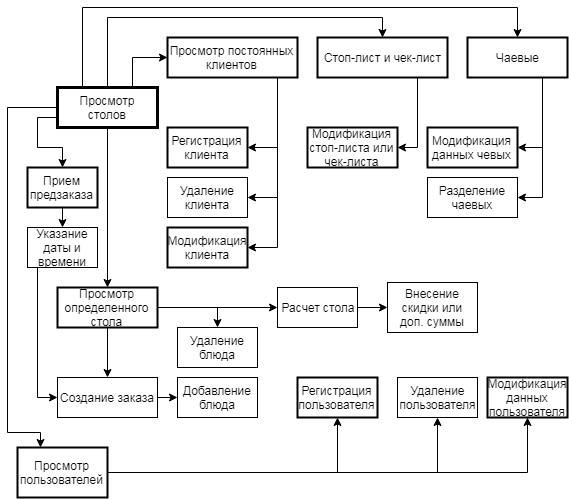
Front-End частина додатку має таку ієрархію сторінок (рис. 5.1):

Рисунок 5.1 – Ієрархія сторінок веб-додатку

На схемі ієрархії окремі сторінки зображені у вигляді прямокутників з товстими межами, а кнопки чи форми, які виконують певний функціонал зображені у формі прямокутників із тонкими межами. Стрілки вказують на те, що сторінка містить посилання на відкриття іншої сторінки, а кнопки містять посилання на відображення інших кнопок чи форм.

Файлова архітектура Front-End частини веб-додатку знаходиться на стороні Back-End частини у модулі Template, її зображено на рисунку 5.2.

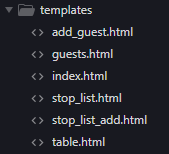


Рисунок 5.2 – Файлова архітектура Front-End частини веб-додатку

Усі елементи є статичними, але відображаються динамічно за допомогою Back-End, а саме модуля View, котрий описаний у наступному підрозділі.

## 5.2 MVC або MVT

Додаток організовується по шаблону MVC. В Django цей шаблон називають MVT, але суті це не міняє. Що з себе являє MVC: Модель (Model), Уявлення (View), Контролер (Controller). Модель відповідає за роботу з даними. Вона взаємодіє з базою даних, перевіряє дані на коректність, створює розрахункові дані в разі потреби. Уявлення відповідає за те, як ці дані будуть виглядати, надає їм легку для читання форму. Контролер є прошарком між моделлю, уявленням і відвідувачем сайту. Коли відвідувач сайту запитує сторінку, цей запит передається контролеру. Він в свою чергу збирає всі моделі, які можу стати в нагоді, і просить їх надати дані. Потім контролер вибирає уявлення, яке найкраще відповідає запиту користувача і передає йому дані. Сформована сторінка (уявлення) передається як відповідь на запит [15]. На рисунку 5.3 зображено схему роботи MVT у Django.

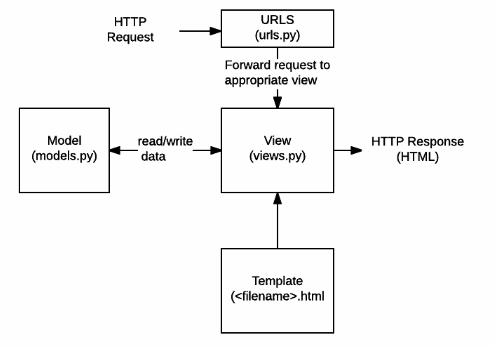


Рисунок 5.3 – Схема роботи моделі MVT y Django

## 5.3 Розробка діаграми розгортки системи

Під час розробки веб-додатку розроблено діаграму розгортки системи. Користувач звертається до контролеру сервера за допомогою візуального інтерфейсу. Контролер звертається до бази даних через модель, стягує з неї дані. Дані переміщуються у зворотному порядкові до користувача. Детальну діаграму розгортки системи зображено на рисунку 5.4.

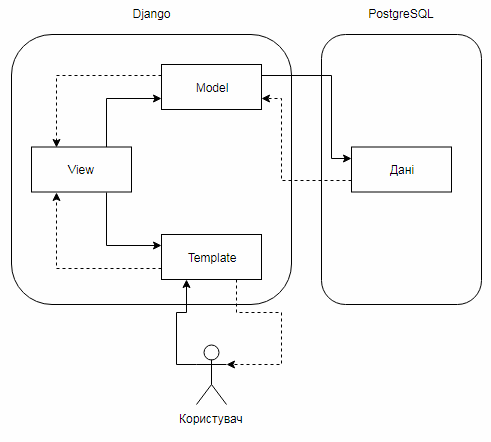
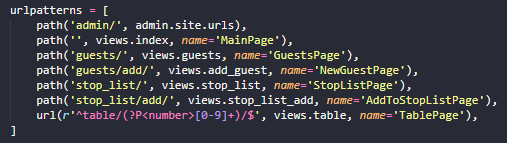


Рисунок 5.4 – Діаграма розгортки системи веб-додатку

## 5.4 Програмна реалізація модулів системи

У діаграмі розгортки системи Template – це Front-End частина додатку, що описана у підрозділі 5.1.

View у діаграмі розгортки (підрозділ 5.2) складається з декількох модулів: urls.py, admin.py, forms.py, views.py. Кожен з них відповідає у відповідному порядкові за перенаправлення веб-браузера користувача по карті сайту (веб-додатку), реєстрацію таблиць бази даних у admin-панелі, створення та використання форм заповнення даними, виконування функцій уявлень, що приймають веб-запроси та відправляють веб-відповіді для заповнення Front-End частини даними з бази даних. Приклади їх програмної реалізації зображено на рисунках 5.5 – 5.8.



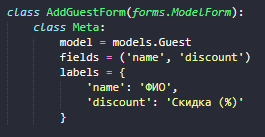
Рисунок 5.6 – Функції реєстрації таблиць у admin-панелі додатку

Рисунок 5.7 – Форма реєстрації постійного клієнта у вигляді класу

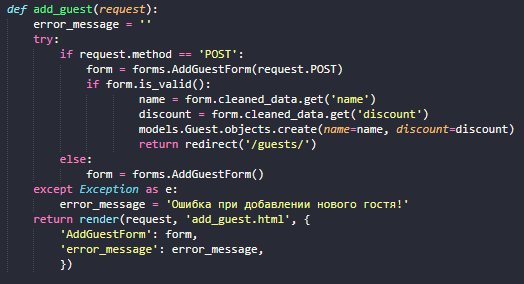


Рисунок 5.8 – Функція уявлення реєстрації постійного клієнта з використанням форми його реєстрації

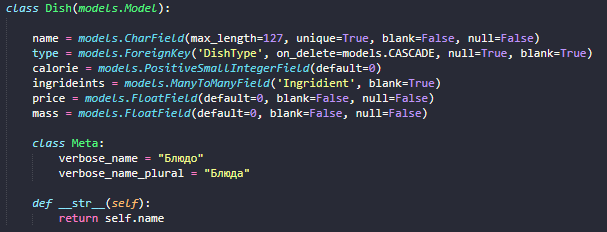
Model, описана у підрозділі 5.2, складається лише з одного модулю models.py, що звертається безпосередньо до бази даних. Приклад програмної реалізації опису таблиці для сутності «Страва» зображено на рисунку 5.9.

Рисунок 5.9 – Опис таблиці сутності «Страва» описаної як клас

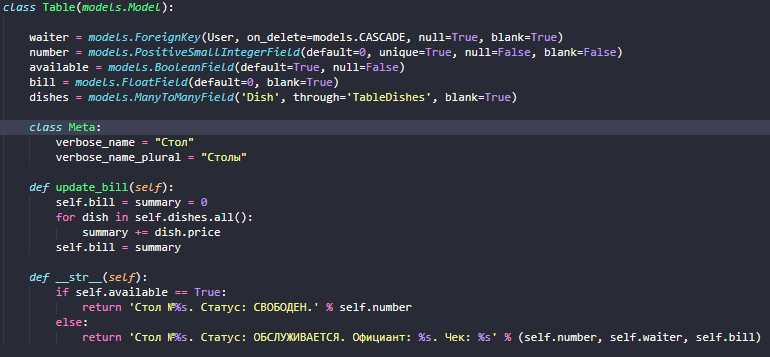
Приклад програмної реалізації опису таблиці для сутності «Стіл» зображено на рисунку 5.10. У даної моделі є п’ять полів (офіціант, номер, зайнятість, сума чеку та набір страв), два методи, що оновлюють суму чеку при додаванні чи видаленні страви зі столу та повертають повну назву об’єкта у вигляді Unicode-строки.

Рисунок 5.10 – Приклад програмної реалізації опису таблиці для сутності «Стіл»

# 6 ОПИС ФУНКЦІЮВАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ

## 6.1 Опис функціювання користувача «Повар»

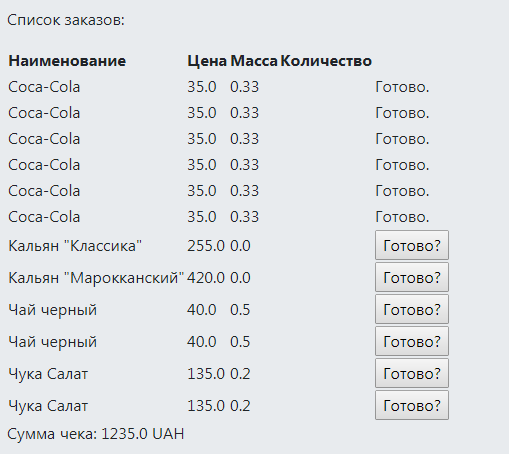
Повар відрізняється від інших користувачів системи лише за однією функцією – при перегляді замовлення він може позначити страву (або декілька) як готову до виносу. На рисунку 6.1 зображено цю функцію візуально для користувача.

Рисунок 6.1 – Сторінка перегляду замовлення з елементами готовності страв

## 6.2 Опис функціювання користувача «Офіціант»

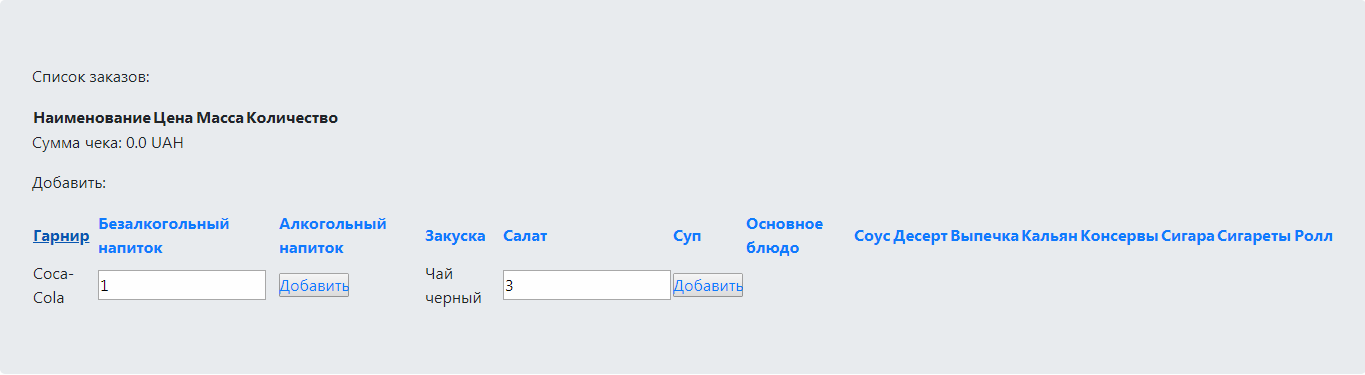
Основними функціями офіціанта є створення замовлення, розрахунок столу та модифікація постійних клієнтів, тому на наступних рисунках 6.2 – 6.4 зображено інтерфейс користувача за заданими функціями.

Рисунок 6.2 – Створення замовлення

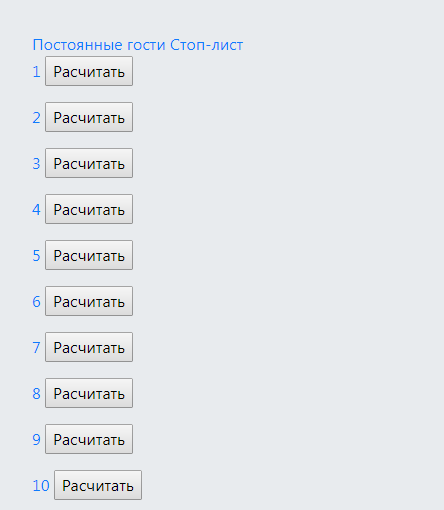


Рисунок 6.3 – Сторінка розрахунку столів



Рисунок 6.4 – Сторінка створення та видалення постійних клієнтів

## 6.3 Опис функціювання користувача «Адміністратор»

Адміністратор відрізняється від інших користувачів системи за такими головними функціями як модифікація користувачів та видалення страв із замовлення створеного офіціантом. Далі на рисунках 6.5 та 6.6 зображено сторінки з таким функціоналом.

Рисунок 6.5 – Частина сторінки з елементами видалення страв із замовлення

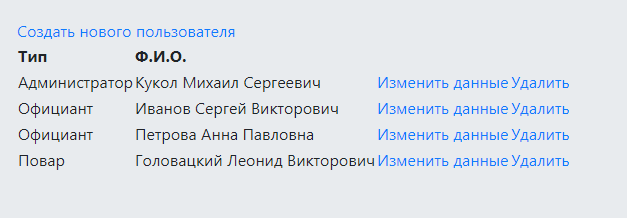


Рисунок 6.6 – Сторінка модифікації користувачів системи

# ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

В результаті виконання даної роботи проаналізовано предметну область веб-орієнтованої системи діяльності ресторану та інших закладів спільного харчування. Для даної веб-системи були поставлені основні завдання, такі як створення, зміна, видалення та перегляд замовлення, зробленного клієнтом, розрахунок окремого клієнта, зміна чек-листу та стоп-листу, позначення страви як готової до виносу, реєстрація та видалення постійних клієнтів, розрахунок чайових для персонала, введення додаткової плати за обслуговування та введеня знижки, та використані такі технології як Python, Django, HTML, CSS, Bootstrap, SQL та PostgreSQL. Тема є актуальною, тому що існуючі системи такого типу не є зручними та досконалими. Система не є досконалою, до неї можна додавати ще безліч функціоналу, але є необхідні для зручності функції, що описані нижче у рекомендаціях.

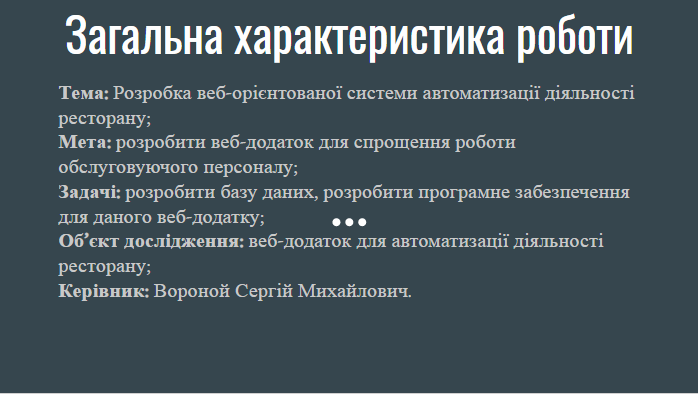
Рекомендації. До системи необхідно додати функції бронювання столів через сторонній веб-сервіс, що є візитною карткою ресторану. Система повинна реалізувати більшість функціоналу, перерахованого в даній роботі і легко масштабуватися. Також рекомендується після створення додатка викласти його на хостінг.

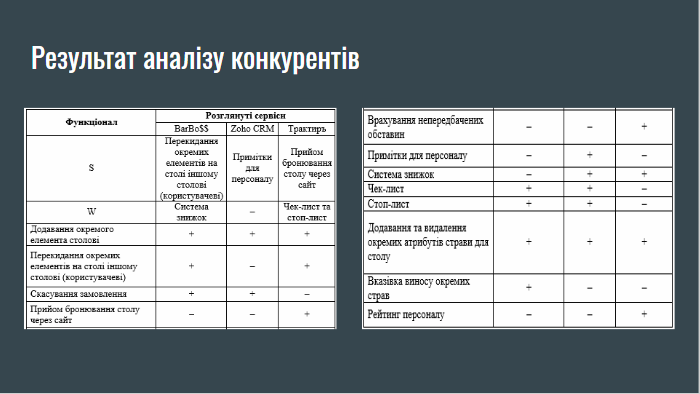
# ПЕРЕЛІК ПОСИЛАНЬ

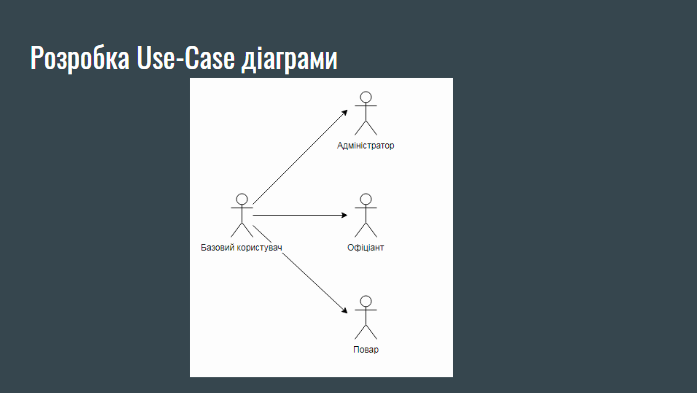
1. Исследование: больше половины украинских компаний все еще ведут учет клиентов в Excel, 3% – на бумаге | AIN.UA [Електронний ресурс] – Режим доступу: <https://ain.ua/2018/04/03/issledovanie-po-crm/>
2. О компании: ТРАКТИРЪ - разработка программного продукта для ресторанов, кафе, столовой [Електронний ресурс] – Режим доступу: <https://traktir.ru/about/>
3. BarBo$$ – искусство управлять. ПО BarBo$$ [Електронний ресурс] – Режим доступу: <https://www.barboss.info/brbousdusd>
4. Free restaurant management software | Manage restaurants, the easy way - Zoho Creator [Електронний ресурс] – Режим доступу: <https://www.zoho.com/creator/apps/restaurant-management.html>
5. Основы UML — диаграммы использования (use-case) — Блог программиста [Електронний ресурс] – Режим доступу: <https://pro-prof.com/archives/2594>
6. SQL – Энциклопедия язык программирования [Електронний ресурс] – Режим доступу: <http://progopedia.ru/language/sql/>
7. Топ фреймворки для веб-разработки в 2018 году - JetRuby Agency [Електронний ресурс] – Режим доступу: <https://jetruby.com/ru/blog/top-freimworki-2018/>
8. AngularJS – Superheroic JavaScript MVW Framework [Електронний ресурс] – Режим доступу: <https://angularjs.org/>
9. Чем хорош Ruby on Rails и как он ускоряет разработку / Блог компании Skillbox / Хабр [Електронний ресурс] – Режим доступу: <https://habr.com/ru/company/skillbox/blog/428487/>
10. Ліцензія MIT – Вікіпедія [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_MIT>
11. Django Overview | Django [Електронний ресурс] – Режим доступу: <https://www.djangoproject.com/start/overview/>
12. Performing raw SQL queries [Електронний ресурс] – Режим доступу: <https://docs.djangoproject.com/en/2.2/topics/db/sql/#django.db.models.Manager.raw>
13. Введение: знакомство с React – React [Електронний ресурс] – Режим доступу: <https://ru.reactjs.org/tutorial/tutorial.html>
14. Introducing JSX – React [Електронний ресурс] – Режим доступу: <https://reactjs.org/docs/introducing-jsx.html>
15. Django, часть 2 MTV или MVC. Модели | VR-online - бесплатный электронный журнал для всех [Електронний ресурс] – Режим доступу: <http://www.vr-online.ru/blog/django-chast-2-mvt-ili-mvc-modeli-8967>

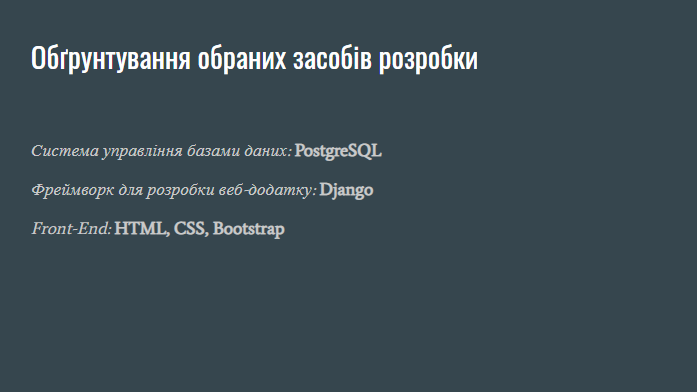
# Додаток А

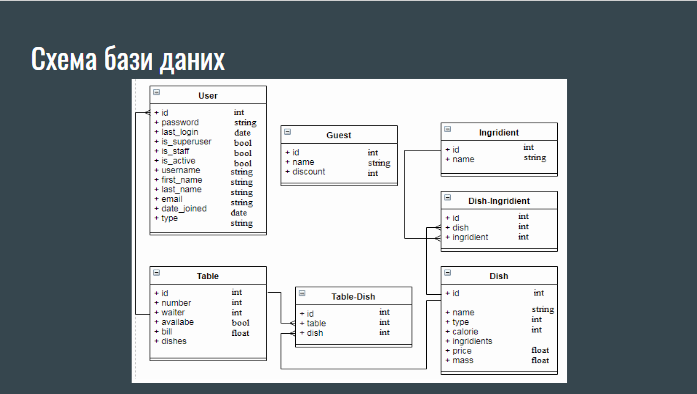
# ПЕРЕЛІК КОПІЙ ДЕМОНСТРАЦІЙНОГО МАТЕРІАЛУ

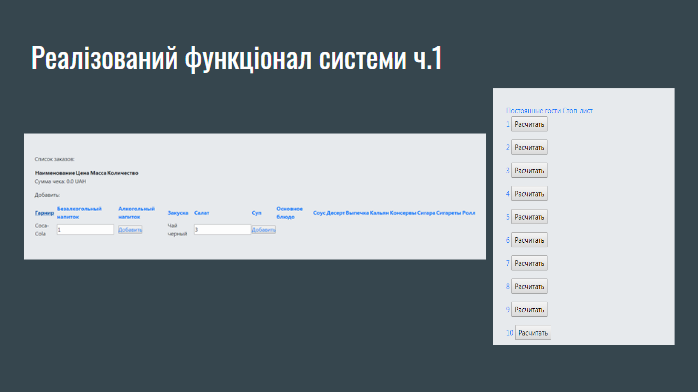


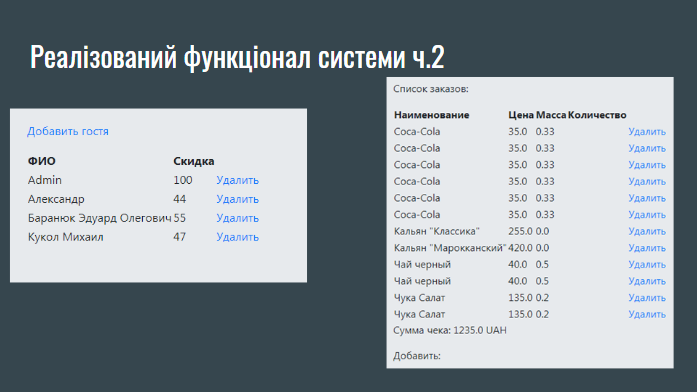












# Додаток Б

# СПИСОК РОЗВ’ЯЗУВАНИХ ЗАВДАНЬ

| **Задача** | **Підзадачі** | **Вхідні параметри** | **Вихідні параметри** |
| --- | --- | --- | --- |
| Модифікация стоп-листу | * Додавання або видалення страви | Страва | Набір страв |
| Модифікация чек-листу | * Додавання або видалення страви * Зазначення кількості страви | Страва, кількість | Набір страв та їх кількість |
| Перегляд замовлення | * Вибір столу та перегляд | Стіл | Набір страв |
| Модифікація постійного клієнта | * Створення та видалення або зміна даних персони | Дані клієнта | Дані клієнта |
| Видалення набору страв із замовлення | * Вибір столу * Вибір набору страв для видалення та їх видаення | Набір страв | – |
| Розрахунок столу | * Вибір столу * Вказання постійного клієнта та додавання знижки | Стіл, клієнт | Роздрукований чек |
| Прийом передзамовлення | * Створення замовлення на певні дату та час | Набір страв, дата та час | Набір страв |
| Модифікація користувачів | * Вибір користувача та зміна його даних | Користувач, дані | Дані користувача |
| Створення замовлення | * Вибір столу * Додавання набору страв по черзі | Стіл, набір страв | Набір страв |
| Стеження за чайовими | * Модифікація суми * Розподіл між вказаною кількістю персон | Сума, кількість персон | Сума на кожну персону |
| Позначення страви як готової до виносу | * Вибір столу * Вибір страви та позначення як готової | Стіл, страва | Страва |

# Додаток В

# ПРОГРАМНИЙ КОД МОДУЛІВ

Програмний код модулів views.py, forms.py, models.py, urls.py, settings.py, admin.py описано нижче відповідно.

# Импорт необходимых библиотек Django

from django.shortcuts import render, redirect

from rest import models, forms

# Объявление представления для главной страницы

def index(request):

try:

models.Table.objects.get(number=100)

except Exception as e:

hundred = [i for i in range(1, 101)]

for number in hundred:

models.Table.objects.create(number=number)

tables = models.Table.objects.all().order\_by('number')

for table in tables:

if table.waiter is not None:

table.available = False

return render(request, 'index.html', {

'tables': tables,

})

# Объявление представления для модификации пользователей

def users(request):

return render(request, 'users.html', {

})

# Объявление представления для модификации постоянных клиентов

def guests(request):

guests\_list = models.Guest.objects.all().order\_by('name')

if request.method == 'GET':

try:

models.Guest.objects.filter(id=request.GET.get('del')).delete()

except Exception as e:

print(e)

return render(request, 'guests.html', {

'guests': guests\_list,

})

# Объявление представленя для создания постоянного клиента

def add\_guest(request):

error\_message = ''

try:

if request.method == 'POST':

form = forms.AddGuestForm(request.POST)

if form.is\_valid():

name = form.cleaned\_data.get('name')

discount = form.cleaned\_data.get('discount')

models.Guest.objects.create(name=name, discount=discount)

return redirect('/guests/')

else:

form = forms.AddGuestForm()

except Exception as e:

error\_message = 'Ошибка при добавлении нового гостя!'

return render(request, 'add\_guest.html', {

'AddGuestForm': form,

'error\_message': error\_message,

})

# Объявление представление для модификации стоп-листа

def stop\_list(request):

dishes = models.StopListDish.objects.all().order\_by('dish\_\_name')

if request.method == 'GET':

models.StopListDish.objects.filter(pk=request.GET.get('del')).delete()

return render(request, 'stop\_list.html', {

'dishes': dishes,

})

# Объявление представленя для добавления блюд в стоп-лист

def stop\_list\_add(request):

error\_message = ''

try:

if request.method == 'POST':

form = forms.AddDishToStopListForm(request.POST)

if form.is\_valid():

dish\_id = form.cleaned\_data.get('dish')

if not models.StopListDish.objects.filter(dish=dish\_id):

models.StopListDish.objects.create(dish=dish\_id)

return redirect('/stop\_list/')

else:

error\_message = 'Данный элемент уже в стоп-листе'

else:

form = forms.AddDishToStopListForm()

except Exception as e:

print(e)

return render(request, 'stop\_list\_add.html', {

'AddToStopListForm': form,

'error\_message': error\_message,

})

# Объявление представления для каждого стола отедльно

def table(request, number):

table = models.Table.objects.get(number=number)

dishes = table.dishes.all()

types = models.DishType.objects.all()

alldishes = models.Dish.objects.filter(type=request.GET.get('f')).order\_by('name')

return render(request, 'table.html', {

'table': table,

'dishes': dishes,

'alldishes': alldishes,

'DishTypes': types,

})

from django import forms

from rest import models

# Объявление формы добавления блюда к столу

class AddDishForm(forms.Form):

quantity = forms.NumberInput()

# Объявление формы создания постоянного клиента

class AddGuestForm(forms.ModelForm):

class Meta:

model = models.Guest

fields = ('name', 'discount')

labels = {

'name': 'ФИО',

'discount': 'Скидка (%)'

}

# Объявление формы добавления блюда в стоп-лист

class AddDishToStopListForm(forms.ModelForm):

class Meta:

model = models.StopListDish

fields = ('dish',)

labels = {

'dish': 'Блюдо',

}

from django.db import models

from django.contrib.auth.models import User

# Объявление модели таблицы "Стол"

class Table(models.Model):

waiter = models.ForeignKey(User, on\_delete=models.CASCADE, null=True, blank=True)

number = models.PositiveSmallIntegerField(default=0, unique=True, null=False, blank=False)

available = models.BooleanField(default=True, null=False)

bill = models.FloatField(default=0, blank=True)

dishes = models.ManyToManyField('Dish', through='TableDishes', blank=True)

class Meta:

verbose\_name = "Стол"

verbose\_name\_plural = "Столы"

def update\_bill(self):

self.bill = summary = 0

for dish in self.dishes.all():

summary += dish.price

self.bill = summary

def \_\_str\_\_(self):

if self.available == True:

return 'Стол №%s. Статус: СВОБОДЕН.' % self.number

else:

return 'Стол №%s. Статус: ОБСЛУЖИВАЕТСЯ. Официант: %s. Чек: %s' % (self.number, self.waiter, self.bill)

# Объявление модели реализации связи многие-ко-многим между таблицей "Стол" и таблицей "Блюдо"

class TableDishes(models.Model):

Table = models.ForeignKey('Table', on\_delete=models.CASCADE)

Dish = models.ForeignKey('Dish', on\_delete=models.CASCADE)

Quantity = models.PositiveIntegerField(default=0, blank=False, null=False)

# Объявление модели таблицы "Блюдо"

class Dish(models.Model):

name = models.CharField(max\_length=127, unique=True, blank=False, null=False)

type = models.ForeignKey('DishType', on\_delete=models.CASCADE, null=True, blank=True)

calorie = models.PositiveSmallIntegerField(default=0)

ingrideints = models.ManyToManyField('Ingridient', blank=True)

price = models.FloatField(default=0, blank=False, null=False)

mass = models.FloatField(default=0, blank=False, null=False)

class Meta:

verbose\_name = "Блюдо"

verbose\_name\_plural = "Блюда"

def \_\_str\_\_(self):

return self.name

# Объявление модели таблицы-словаря "Ингридиент"

class Ingridient(models.Model):

name = models.CharField(max\_length=127, unique=True, blank=False, null=False)

class Meta:

verbose\_name = "Ингридиент"

verbose\_name\_plural = "Ингридиенты"

def \_\_str\_\_(self):

return self.name

# Объявление модели таблицы-словаря "Тип блюда"

class DishType(models.Model):

name = models.CharField(max\_length=255, unique=True, blank=False, null=False)

class Meta:

verbose\_name = "Тип блюда"

verbose\_name\_plural = "Типы блюд"

def \_\_str\_\_(self):

return self.name

# Объявление модели таблицы-словаря "Постоянный клиент"

class Guest(models.Model):

name = models.CharField(max\_length=255, unique=True, blank=False, null=False)

discount = models.PositiveSmallIntegerField(default=0, blank=True, null=False)

class Meta:

verbose\_name = "Гость"

verbose\_name\_plural = "Гости"

def \_\_str\_\_(self):

return '%s (-%s процентов)' % (self.name, self.discount)

# Объявление модели таблицы "Стоп-лист"

class StopListDish(models.Model):

dish = models.ForeignKey('Dish', on\_delete=models.CASCADE)

class Meta:

verbose\_name = "StopListDish"

verbose\_name\_plural = "StopListDishes"

def \_\_str\_\_(self):

return self.dish.name

from django.contrib import admin

from django.urls import path

from django.conf.urls import url

from rest import views

# Объявление реализованных веб-адресов, связанных с веб-приложением

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.index, name='MainPage'),

path('guests/', views.guests, name='GuestsPage'),

path('guests/add/', views.add\_guest, name='NewGuestPage'),

path('stop\_list/', views.stop\_list, name='StopListPage'),

path('stop\_list/add/', views.stop\_list\_add, name='AddToStopListPage'),

path('users/', views.users, name='UsersList'),

url(r'^table/(?P<number>[0-9]+)/$', views.table, name='TablePage'),

]

"""

Django settings for project project.

Generated by 'django-admin startproject' using Django 2.2.1.

For more information on this file, see

https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/2.2/ref/settings/

"""

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'hhue\*n\_zndlb$\*x4-w@2-$%0c6q1v%qi)elcp6#q!(7$\*!6j9c'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = ['\*']

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'rest',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'project.urls'

TEMPLATES\_DIR = os.path.join(BASE\_DIR, 'templates')

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [TEMPLATES\_DIR,],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'project.wsgi.application'

# Database

# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': 'Database',

'PASSWORD': '521467misha',

'USER': 'mikhail',

'PORT': 5432,

'HOST': 'localhost',

}

}

# Password validation

# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Europe/Kiev'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATICFILES\_DIR = os.path.join(BASE\_DIR, 'static')

STATICFILES\_DIRS = [

STATICFILES\_DIR,

]

STATIC\_URL = '/static/'

from django.contrib import admin

from .models import \*

# Регистрацией моделей в admin-панели

admin.site.register(Table)

admin.site.register(Dish)

admin.site.register(Ingridient)

admin.site.register(Guest)

admin.site.register(DishType)

admin.site.register(TableDishes)

admin.site.register(StopListDish)