ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕНЬ РЕСТОРАННОГО БІЗНЕСУ МОВОЮ PYTHON»**

Виконав: студент 5 курсу, групи ППЗ-51
Спеціальності:
121 Інженерії програмного забезпечення
(шифр і назва спеціальності)

Шушков П.М.
(прізвище та ініціали)
Керівник Негоденко О.В.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль_____
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____О.В. Негоденко

“____”____20___року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Шушкову Павлу Миколайовичу
(прізвище, ім‘я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для автоматизації замовлень ресторанного бізнесу мовою Python»
Керівник роботи к.т.н., доцент Негоденко О.В.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “12” березня 2021 року №65.
2. Строк подання студентом роботи 1.06.2021.
3. Вихідні дані до роботи:
   3.1. Положення існуючих інформаційних технологій для автоматизації замовлень ресторанного бізнесу;
   3.2. Аналіз програмних засобів для автоматизації замовлень ресторанного бізнесу;
   3.3. Розробка моделі програмного забезпечення з автоматизації замовлень;
   3.4. Науково-технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1. Загальні положення побудови веб-додатків;
   4.2. Аналіз технологій і методів побудови веб-додатку;

4.3. Побудова веб-додатку для автоматизації замовлень ресторанного бізнесу;
4.4. Висновки

5. Перелік графічного матеріалу.
5.1. Основні характеристики роботи;
5.2. Актуальність задачі;
5.3. Архітектура веб-додатку;
5.4. Висновки.

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.21	Виконано
2	Дослідження положення побудови рекомендаційних систем	23.04.21	Виконано
3	Аналіз методів побудови рекомендаційних систем	27.04.21	Виконано
4	Розробка моделі рекомендаційної системи	02.05.21	Виконано
5	Висновки, оформлення роботи	19.05.21	Виконано
6	Розробка демонстраційних матеріалів	21.05.21	Виконано
7	Здача роботи	23.05.21	Виконано

Студент _____ Шушков П.М.
(підпис) (прізвище та ініціали)
Керівник роботи _____ Негоденко О.В.
(підпис) (прізвище та ініціали

РЕФЕРАТ

Текстова частина бакалаврської роботи: 51ст., 24 рис., 39 джерел.

АВТОМАТИЗАЦІЯ ЗАМОВЛЕНЬ, ВЕБ-ДОДАТОК, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ.

Об'єкт дослідження: замовлення в ресторанному бізнесі

Мета роботи: розробка та реалізація програмного забезпечення для автоматизації замовлень ресторанного бізнесу мовою Python.

Предмет дослідження: автоматизація замовлень ресторанного бізнесу

Методи дослідження: аналіз, порівняння, узагальнення, методи математичної обробки, методи графічного представлення інформації, комп'ютерне моделювання.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному суспільстві кожне підприємство для успішного ведення бізнесу намагається автоматизувати більшість з можливих бізнес-процесів. Автоматизація бізнес процесів – це переведення типових бізнес-задач та стандартних операцій під контроль програм, що на виході дозволяє вивільнити ресурси та збільшити продуктивність та ефективність праці.

Варто розуміти, що розробка інформаційних систем є одним з найбільш популярних інструментів для успішної організації діяльності будь-якої компанії, але лише гарно розвинені та великі підприємства можуть собі дозволити написання подібних програм під потреби свого бізнесу.

Разом з розповсюдженням та доступністю Інтернету за останні десять років ми спостерігаємо значне зростання електронної комерції, яке відповідно пов’язане з переходом споживачів в онлайн.

Особливої актуальності в останні роки набула автоматизація діяльності ресторанного комплексу, так спостерігається як зростання ресторанів з функцією онлайн замовлень, так і сторонніх сервісів, що пропонують замовити їжу з ресторану.

Такі зміни обумовлені як попитом зі сторони клієнта, який хоче економити свій час та мати можливість не готувати та оформлювати замовлення з доставкою не виходячи за межі свого дому, так і зі сторони бізнесу, який зацікавлений в подібних сторонніх платформах, які не потребують розробки та дають постійний потік клієнтів.

І авжеж, особливої популярності подібні сервіси набули за умов пандемії COVID-19, з обмеженими можливостями для бізнесу зі сторони ресторанів та бажанням максимально залишатись у безпеці зі сторони клієнтів

Виходячи з вищезазначеного ми можемо зробити висновок, що автоматизація ресторанного бізнесу як ніколи актуальна тема, оскільки вирішує як питання бізнесу, так і спрощує життя людей.

В даній роботі пропонується до розгляду питання розробки сучасного програмного забезпечення для автоматизації ресторанного бізнесу.

Мета роботи: засоби автоматизації замовлень ресторанного бізнесу за допомогою додатку на мові Python.

Завдання дипломної роботи:

- Дослідити використання інформаційних технологій в автоматизації замовлень ресторанного бізнесу
- Проаналізувати програмні засоби для автоматизації замовлень із ресторанів
- Проаналізувати вимоги до програмного забезпечення для автоматизації замовлень ресторанів
- Запропонувати архітектуру для програмного забезпечення
- Дослідити та обрати оптимальні технології для реалізації програмного забезпечення
- Спроектувати базу даних
- Розробити програмне забезпечення з автоматизації ресторанного бізнесу
- Запропонувати можливості для покращення програмного забезпечення

Об'єкт дослідження: підвищення ефективності роботи ресторанного бізнесу

Предмет дослідження: технології автоматизації замовлень ресторанного бізнесу

Методи дослідження: аналіз, порівняння, узагальнення, методи математичної обробки, методи графічного представлення інформації, комп’ютерне моделювання.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ.

1.1 Інформаційні технології та автоматизація замовлень ресторанного бізнесу.

Швидке зростання електронної комерції породило багато нових форм бізнесу, таких як B2B (бізнес для бізнесу), C2C (від клієнта до клієнта), B2C (від бізнесу до клієнта) та O2O (від онлайну до офлайну. тобто через Інтернет) [1].

Бізнес O2O - це маркетинговий метод, заснований на інформаційно-комунікаційних технологіях, за допомогою якого споживачі розміщують замовлення на товари чи послуги в Інтернеті та отримують товари чи послуги за адресою [2].

Однією із значних подій, що спричинила вибух комерції O2O, стало поширення смартфонів і планшетів та розвиток інфраструктури для підтримки оплати та доставки. У 2019 році відбулось 5,2 мільярда підключень для смартфонів, і це було передбаченням того, що до кінця 2020 року половина людей усвіті матиме доступ до послуг мобільного Інтернету [3].

Послуги O2O з’явились у різних сферах, включаючи придбання різноманітних продуктів та послуг таких як їжа, готельні номери, нерухомість або оренда автомобілів [4]. В даній роботі мова йтиме про процес замовлення та доставки їжі онлайн, коли їжу замовлену у ресторану через Інтернет, готують та доставляють споживачеві. Розвиток цієї сфери було підкріплено розвитком інтегрованих Інтернет-платформ доставки їжі, таких як Uber eats, Deliveroo, Swiggy та Meituan.

Інтернет-платформи з доставки їжі виконують різноманітні функції, включаючи (рисунок 1) [5]:

- надання споживачам широкого вибору продуктів харчування;
- прийняття замовлень;
- передача замовлення закладу, для виконання замовлення;
- моніторинг оплати;

- організація доставки їжі;
- надання засобів відстежування.

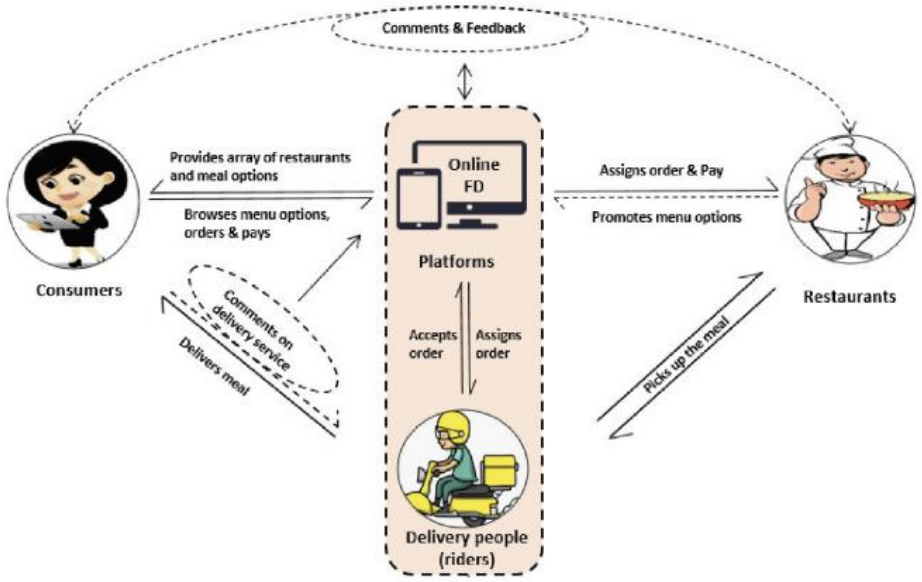


Рисунок 1.1.1 - Функції, пов'язані з платформами онлайн доставки їжі (FD)

Стрілки вказують на рух інформації або логістики; рядки вказують необхідні маршрути; пунктирними лініями вказуються додаткові маршрути.

Провайдери, які надають послуги з постачання страв можуть бути категоризовані як:

- від ресторану до споживача – ресторан сам виготовляє страви та постачає замовлення споживачу;
- від платформи з доставки їжі до споживача – замовлення здійснюється через сторонню платформу та доставляється споживачу. Такі платформи різняться від країни до країни: для США це Uber eats, Eleme для Китаю, Just Eat для UK, Swiggy для Індії, Glovo для України. [6].

Зростання онлайнових платформ з доставки їжі - це глобальна тенденція, у багатьох країнах світу є хоча б одна основна платформа для доставки їжі. Онлайн платформи з доставки їжі займають дуже активну позицію на нових ринках та сприяють розвитку харчових звичок споживачів. Наприклад, у 2018

році індійська онлайн платформа з доставки їжі Foodpanda надали споживачам великі знижки, що призвело до збільшення кількості користувачів Foodpanda у 10 разів [7]. Більше того, у 2018 році онлайн платформа Eleme у Китаї витратила три мільярди юанів (443 млн. Дол. США) протягом трьох місяців на маркетингову стратегію, та збільшила свою частку ринку до понад 50 % на китайському ринку [8].

Незважаючи на те, що в деяких регіонах онлайн платформи з доставки їжі дуже сильні, в цілому по всьому світу вони перебувають на ранніх стадіях розвитку ринку, часто такі платформи на етапах свого розвитку кооперуються з ресторанами та надають останнім субсидії. [9]. Наприклад, ресторан може проводити промо-кампанію на платформі з доставки їжі, в якій споживач отримує 50 грн знижку, за умови, що загальна сума замовлення сягає 300 грн. Насправді, ця знижка може коштувати ресторану лише 20 грн, оскільки він отримає субсидію в 30 грн від платформи (фактичні правила можуть варіюватися від однієї платформи до іншої, але основний момент це кооперація з метою росту популярності)[10]. Такий підхід вигідний для ресторану, оскільки він дозволяє залучити більше споживачів та збільшити кількість замовлень, а для платформи з доставки це можливість формувати нові харчові звички споживачів, знайомити їх з процесом вибору та придбанням їжі через Інтернет, формувати довіру до замовлень через платформу та можливість сформувати звичку до простих та зручних замовлень через їх платформу.

Аби досягти таких цілей платформи нерідко надають споживачу можливість їсти за нижчою ціною ніж в закладі, або отримувати замовлення з безкоштовною доставкою, таким чином платформи та постачальники заохочують споживачів збільшувати кількість онлайн замовлень.

<u>Економічна вигода</u>

- з виникненням вірусу COVID-19 – такі платформи для багатьох закладів стали єдиною можливістю аби продовжити свою діяльність та зберегти робочі місця;

- це можливість для традиційних закладів харчування та невеликих кафе нагадати про себе, або ж сформувати знання, та залучити нову аудиторію і залишатись у бізнесі при цьому не витрачаючи кошти на розробку власного сервісу з онлайн замовлень;
- розширення робочих місць для кухарів, людей, що займаються доставкою, програмістів та скорочення часових витрат на функції обслуговування;
- можливість для закладів зменшити зали та скоротити витрати пов’язані з забезпеченням простору;
- можливість легко змінювати та оновлювати меню, ціни;
- можливість від однієї компанії створювати декілька закладів, але при цьому готувати в межах однієї кухні для обох;
- можливість швидко інтегрувати промо-пропозиції та точно знати, що споживач про них дізнається;
- залучення нової аудиторії без додаткових інвестицій зі сторони ресторану.

Соціальна вигода

Онлайн-замовлення впливають на взаємозв'язок між споживачем та їжею, змінюючи спосіб яким споживачі отримують, готують та споживають їжу. У свою чергу, ці зміни впливають на стосунки між людьми, що призвело до суттєвих дискусій щодо того, посилюють чи зменшують онлайн платформи з доставки їжі якість сімейного часу та взаємодії людей.

Традиційно члени сім'ї спілкувалися між собою і насолоджувались сімейною компанією, займаючись повсякденними аспектами сімейного життя, пов’язаними з харчуванням - наприклад, покупка продуктів та приготування їжі вдома [11]. Справді, в деяких випадках це так, є дані, що заміжні корейські жінки рідше користуються подібними платформами, оскільки вони вважають, що мають моральний обов'язок готувати їжу для своїх сімей [12]. На відміну від цього, інші дослідження повідомляють що деякі споживачі у Китаї [13] та

Великобританії [14] розглядають можливість замовлення їжі додому, як спосіб швидко та легко отримати їжу та не витрачати час на її приготування, а провести цей час з родиною. Наприклад, якісне дослідження в Гуанчжоу (найбільше місто Південного Китаю) людей у віці від 18 до 35 років, які замовляють доставку їжі принаймні раз на тиждень, виявили, що вони використовують платформи з онлайн доставки їжі, оскільки це дозволило їм насолодитись сімейним затишком та зменшити стрес, пов'язаний з купівлею та приготуванням їжі [15].

Не виникає сумнівів, що замовлення їжі онлайн скорочує витрати часу на покупки та приготування, але особливо цікавим є аспект єднання, який формується за рахунок об'єднання людей в межах одного замовлення. Такі ситуації часті для родин, друзів та співробітників. Відтак, якщо замовлення сформовано спільно з колегами, такий підхід не лише економить час але й сприяє кращому спілкуванню, оскільки вони можуть насолоджуватись своїми стравами разом [16]. В Італії «Just Eat Observatory» зафіксувала зростання замовлень на 137% за доставлені обіди в 15 італійських містах у 2017 році, які вони приписували працівникам, які все частіше замовляли та їли страви, які доставляються безпосередньо до їх роботи [17].

З іншого боку замовлення онлайн надають можливість харчуватись окремо для тих, хто не бажає ділити цей процес та час ще з кимось.Таким чином, онлайн замовлення дають можливість людям, які бажають цього – їсти наодинці та одночасно забезпечуючи групи, які бажають поїсти разом можливістю розділити цей процес з групою інших людей та ще й розділити вартість доставки.

Нерідко у замовленнях, які оформляються в закладі чи за телефоном можуть виникати непорозуміння, що призводить до розчарування. Замовлення, які розміщені онлайн в повному контролі та відповідальності клієнта, що робить процес кристально чистим.

Особливо позитивним є той факт, що онлайн-замовлення та доставка забезпечили критично важливий порятунок людей протягом 2020 року, під час

пандемія COVID-19 для десятків мільйонів людей, які перебувають удома на карантині. А можливість швидко адаптуватись під нові реалії дала можливість онлайн-платформа швидко внести зміни в процес обслуговування та звести контакт практично до 0.

Отож, ми можемо зробити висновок, що галузі, які відстають від систем онлайн-замовлення, залишилися позаду на відміну від ресторанів, які активізували свою діяльність у даному напрямку. Впровадження нових технологій не тільки приносить користь життю клієнтів, але й допомагає бізнесу ефективно функціонувати. Справжні переваги системи онлайн-замовлення: залучення людей, які мають зайнятий спосіб життя; онлайн замовлення їжі дозволяє клієнтам розміщувати замовлення віртуально, в будь-який час і з будь-якого місця, що допомагає економити час споживачів, оптимізувати роботу персоналу та контролювати замовлення клієнтом.

1.2 Аналіз програмних засобів для автоматизації замовлень із ресторанів

Ресторани з метою автоматизації своїх замовлень можуть використати один з трьох варіантів: використовувати незалежну платформу з онлайн замовлень, підключити платний модуль до свого веб-сайту, замовити розробку власної системи оформлення замовлень онлайн.

Задачі, які вирішує незалежна платформа з доставки їжі:

- збирати користувачів в межах однієї платформи;
- надавати інформацію про заклади з яких можна зробити замовлення;
- надавати інформацію, щодо опису закладу, меню, цін на страви, вигідні пропозиції;
- надати можливість оформити та оплатити замовлення;
- обрати час та місце для доставки.

Платформи для онлайн бронювання самостійно розробляють додатки, в яких клієнту зручно оформлювати замовлення. Сервіс може просити у заклада харчування абонентську плату за користування, або знижки для клієнтів. Ресторан в свою чергу має надати інформацію, щодо своїх послуг, а надалі лише оновлює інформацію та отримує повідомлення щодо нових замовлень, які має обробити.

В Україні функціонує дві платформи з онлайн-замовлень, пропоную розглянути їх більш детально.

Сервіс Glovo - це відома міжнародна платформа оформлення онлайн-замовлень з доставки їжі, яка працює у понад 20-ти країнах світу серед яких і Україна. У даній системі клієнти можуть замовити будь-який товар невеликого розміру (40х40х30см) вагою до 9 кг[18]. У Європі понад 85 % [19] замовлень складає доставка їжі. Після того як клієнт оформить замовлення, він має можливість відстежувати переміщення кур'єра в режимі реального часу. У сервісу велика кількість закладів, інформативність та наглядність у подачі інформації. Сервіс вказує не час доставки, а відстань яку долає кур’єр, але сервіс обіцяє доставити будь-яке замовлення за годину після того, як його візьме в роботу кур'єр. До акаунту можна прив'язати карту, або розрахуватись готівкою. Перевагами даного сервісу є те, що можна замовляти не лише їжу, але й інші товари, а також те, що сервіс доступний для використання через десктоп, IOS та Android.

Сервіс Roket – це українська компанія, яка надає послуги з доставки їжі та продуктів за допомогою мобільного додатку. Вартість доставки замовлення фіксована та складає 40 грн. Сервіс доступний лише у вигляді мобільного додатку.

Другий варіант – підключити готовий модуль онлайн-бронювання на власний сайт. Підприємство купує модуль у розробника, самостійно залучає аудиторію до форми бронювання, через пошукові системи чи соціальні мережі.

Підприємство самостійно вирішує, на яких умовах здійснювати бронювання та самостійно управляє броню.

Третій варіант – самостійно розробити модуль для оформлення замовлень для сайту чи мобільного додатку. Основна перевага в тому, що ресторан може реалізувати будь-які потреби свого бізнесу. Недоліки такого підходу – час для реалізації та ціна.

Системи автоматизації замовлень ресторанного бізнесу стають все більш функціональними, а кількість закладів та клієнтів демонструють постійне зростання.

Оскільки саме незалежні платформи демонструють стрімке зростання, як клієнтів так і закладів, які долучаються до даних систем, саме на них буде зосереджено основний фокус уваги в межах даної роботи, як оптимальних та найбільш ефективних для вирішення основної мети нашої роботи – автоматизації замовлень.

1.3 Аналіз вимог до програмного забезпечення

Виходячи аналізу зробленого вище, можемо зробити висновок, що веб-додатки на сьогоднішній день це найпопулярніший сервіс для онлайн замовлень, оскільки має такі переваги як: кросплатформеність, простота в розробці та адаптації, зручність для клієнта, тому ми приймаємо рішення, що розробка веб-додатку є оптимальним рішенням для досягнення мети даної роботи.

Веб-додаток – це комп’ютерна програма, яка використовує веб-браузер для виконання певної функції та присутні на багатьох сайтах. Простий приклад - контактна форма на веб-сайті та являє собою клієнт-серверна програму. Веб-додатки розвивалися з моменту їх винаходу. Одні з перших додатків, на мові Perl, на стороні сервера, була розроблена в 1987 році. Це було до того, як Інтернет справді став популярним поза академічними та технологічними

колами. Перші веб-програми були відносно простими і стали більш досконалими наприкінці 90-х. Сьогодні вони є частиною повсякденного життя мільйонів американців.

Призначення веб-додатка “Restaurant Automation” (назва автора) - автоматизація роботи ресторанного бізнесу за рахунок створення незалежної платформи в межах якої ресторани можуть надавати свої послуги з приготування та доставки їжі, а клієнт має можливість обрати заклад та оформити своє замовлення в режимі онлайн.

Вимоги до функцій веб-додатку “Restaurant Automation” (онлайн замовлення з ресторанів):

1. Створення закладу працівником ресторану;
2. Формування меню закладу працівником ресторану;
3. Наповнення меню закладу працівником ресторану позиціями та цінами;
4. Можливість отримати замовлення рестораном від клієнта (можливість одночасно отримувати замовлення від декількох клієнтів);
5. Перегляд доступних до замовлення закладів;
6. Перегляд та вибір клієнтом закладу та позиції для замовлення;
7. Можливість клієнта змінити заклад та позиції в замовленні;
8. Можливість клієнта обрати час та місце доставки замовлення;
9. Можливість клієнта оплатити замовлення через картку онлайн.

Загальні сценарії роботи користувача з програмним забезпеченням (рисунок 1.3.1., 1.3.2):

1. Користувач (клієнт) заходить на веб-сторінку;
2. Клієнт бачить перелік закладів з яких можна зробити замовлення;
3. Клієнт обирає заклад та вибирає позиції, які він додає до кошику замовлення;
4. Клієнт проходить реєстрацію;

5. Переходить на сторінку оплати, де бачить перелік позицій свого замовлення та суму до оплати;

6. Клієнт має можливість на сторінці оплати обрати адресу та час доставки та оплачує своє замовлення;

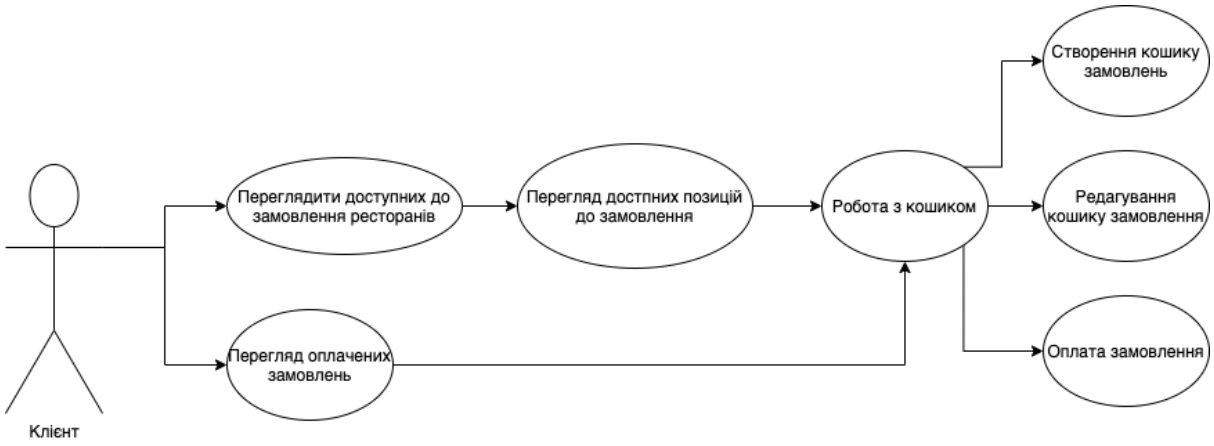


Рисунок 1.3.1 - UML-діаграма можливостей клієнта в системі

7. Працівник закладу бачить замовлення: замовленні позиції, годину до котрої потрібно доставити замовлення та адресу;

8. Працівник закладу має змогу редагувати замовлення.

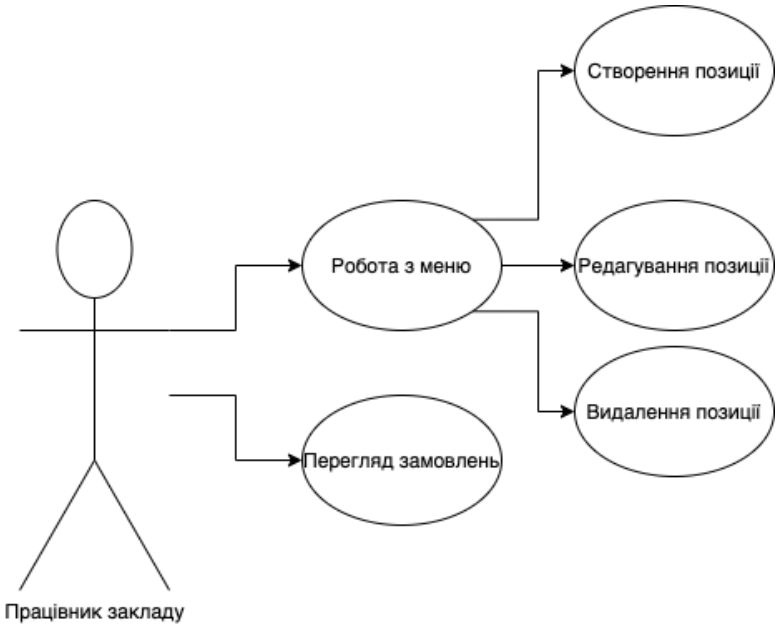


Рисунок 1.3.2 - UML-діаграма можливостей працівника закладу в системі

Необхідно передбачити можливість інтеграції програмного забезпечення “Restaurant Automation” з веб-сайтом закладу та внесення змін в оформлення: зовнішній вигляд титульного банеру закладу та зображення позицій закладу за рахунок файлу стилів CSS та розділення коду на модулі.

Серед можливих варіантів розширення функціоналу даного веб-додатку можемо виділити наступне: підключення модуля оплати за замовлення (на кшталт Privat24, Liqpay тощо), захист від спаму при оформленні замовлення онлайн, антифрот система, внесення додаткових даних користувача (поля типу email, побажання, відгуки та т.п.), можливість реєстрації через oAuth (FB або Google)б можливість розсилки e-mail пропозицій клієнтам, надання індивідуальних промо-пропозицій, таких як знижка до Дня народження клієнта тощо.

Виходячи з вимог до функціоналу веб-додатку “Restaurant Automation” можемо зробити висновок, що він підвищить зручність послуг з замовлень їжі з закладів харчування для клієнтів, оскільки надасть можливість оформлювати їх онлайн, оптимізує роботу персоналу закладу, підвищить привабливість закладу для клієнтів та дозволить створити та вести базу клієнтів, що розширить можливості закладу з точки зору промо-активностей та цікавих пропозицій.

2. ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕНЬ

2.1. Архітектура веб додатку

Клієнт-серверна архітектура - це обчислювальна модель, в якій сервер розміщує, постачає та управляє більшістю ресурсів та послуг, які споживає клієнт (рисунок 2.1.1). Цей тип архітектура має один або кілька клієнтських комп'ютерів, підключених до центрального сервера через мережу або підключення до інтернету. Клієнт-серверна архітектура також відома як модель обчислювальних мереж або клієнт-серверна мережа, оскільки всі запити та послуги доставляються через мережі.

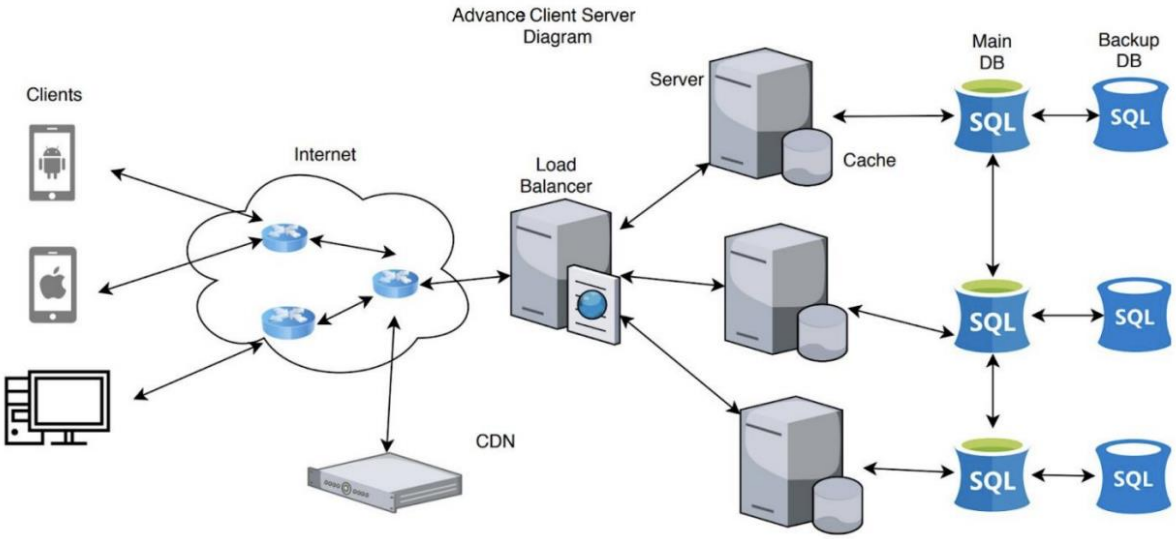


Рисунок 2.1.1 - клієнт-серверна архітектура

Потік даних є односпрямованим і утворює цикл. Зазвичай він ініціюється клієнтом, який запитує якісь дані і сервер обробляє запит враховуючи бізнес логіку і відправляє якісь дані назад до клієнта. Клієнти не можуть безпосередньо спілкуватися між собою.

Типовий топологічний потік даних відбувається наступним чином:

1. Клієнт запитує дані від сервера
2. Балансир навантаження направляє запит на відповідний сервер
3. Сервер обробляє запит клієнта враховуючи бізнес логіку
4. Сервер запитує відповідну базу даних для деяких даних
5. База даних повертає запитувані дані назад на сервер
6. Сервер обробляє дані і відправляє дані назад клієнту
7. Цей процес повторюється

Логіка веб-додатку розподілена між сервером і клієнтом, існує канал для обміну інформацією, а дані та логіка додатку зберігаються на сервері. Перевагою такого підходу є незалежність клієнта від операційної системи користувача, саме тому веб-додатки є міжплатформовими сервісами. Особливої популярності веб-додатки набули в кінці 1990-х - початку 2000-х років, якою завдячують простоті своєї розробки та універсальності.

Отож, серед переваг веб-додатків варто зазначити незалежність виконання функцій від операційної системи, що обумовлює відсутність необхідності адаптовувати чи писати веб-додаток під різні операційні системи, а можливість створити додаток один раз для довільної платформи на якій його і буде розгорнуто. Але такі фактори як різна реалізація HTML [20], CSS [21], DOM [22] й інших специфікацій в браузерах може викликати проблеми при розробці веб-додатків і подальшої їх підтримки. Також перешкоджати роботі застосунку можуть індивідуальні налаштування браузера користувачем.

Наступним кроком є вибір паттерну.

Паттерн - це повторювана архітектурна конструкція, яка представляє собою рішення проблеми проектування в рамках певного контексту, який часто виникає.

Основними критеріями для вибору патерну можна розділити на наступні групи:

Критерії розробників

1. Швидкість розробки – впровадження нових функцій, рефакторинг, розпаралелювання процесу розробки програмного забезпечення.

2. Продуктивність – максимальна швидкість відповіді сервера з мінімальним споживанням обчислювальної потужності.

3. Масштабованість – можливість збільшити обчислювальну потужність або простір на диску при збільшенні кількості інформації та / або кількості користувачів. Якщо використовується виділена масштабована система, потрібно забезпечити узгодженість даних, доступність та допуск на розділи (теорема CAP). Варто також зазначити, що випадок, коли кількість функцій / екранів клієнтської програми збільшується на прохання власника програмного забезпечення, залежить не від типу архітектури веб-додатків, а від структури та реалізації.

4. Тестування – можливість і простота автоматизованого модульного тестування.

Критерії власника програмного продукту

1. Можливість розширювати функції – можливість розробки нового функціоналу за мінімальний час та бюджет.

2. SEO – користувачі повинні мати можливість знайти програму через будь-яку пошукову систему.

3. Підтримка – окрім розробки власного програмного забезпечення, існують додаткові витрати: обладнання, мережева інфраструктура, обслуговування.

4. Безпека – власник програмного забезпечення повинен бути впевнений, що як ділові дані, так і інформація про користувачів захищаються. В якості основного критерію безпеки варто розглянути можливість зміни функціональних можливостей поведінки додатків на стороні клієнта та всі пов'язані з цим ризики. Стандартні небезпеки однакові для всіх архітектур. Ми не розглядаємо безпеку на каналі сервер-клієнт, оскільки всі ці архітектури однаково схильні до злому. Цей канал може бути однаковим.

5. Адаптивність: перетворення на мобільний або настільний додаток з мінімальними додатковими витратами.

Нижче розглянемо основні варіанти патернів:

Паттерн Model-View-Controller (MVC) - це фундаментальний паттерн, який знайшов своє застосування в багатьох технологіях та дав поштовх для розвитку новим технологіям і кожен день полегшує життя розробникам.

Вперше паттерн MVC з'явився в мові SmallTalk, як архітектурне рішення, яке дозволило відокремити графічний інтерфейс від бізнес логіки, а бізнес логіку від даних. В класичному варіанті, MVC складається з трьох частин, які і дали йому назву. Розглянемо їх:

Модель - частина, що містить в собі бізнес-логіку програми. Вона має бути незалежною від інших частин продукту, а модельний шар не має знати нічого про елементи дизайну і яким чином дизайн буде відображатись. Таким чином отримуємо результат, яки дозволяє змінювати представлення даних не чіпаючи саму модель. Модель налідена наступними ознаками:

- Це бізнес-логіка програми;
- Вона володіє знаннями про саму себе та не знає про контролери та уявлення;
- Для певних проектів модель виступає як шар даних, а для інших вона менеджер бази даних, набір об’єктів чи просто логіка додатку.

View - відображає дані отримані від моделі, але ця частина не має безпосереднього впливу на модель, тобто view має доступ “лише на читання” даних. View наділене наступними ознаками:

- Реалізує відображення даних, що виходять від моделі;
- Інколи може мати код, який реалізує певну бізнес-логіку.

Приклади: HTML-сторінка, WPF форма, Windows Form.

Відмінності MVC & MVVM & MVP

Найбільш поширені види MVC-патерну, це:

Model-View-Controller - точкою входу в програму є контролер. Контролер має посилання на модель і на представлення. Контролер підписується на події представлення, а представлення підписується на події моделі. Коли користувач робить якісь дії, управління переходить до контролера і контролер впливає на модель. При зміні моделі йде вплив контролер отримує відповідну подію і впливає на view для оновлення даних. У реалізації MVC незалежною частиною є модель, вона нічого не знає про подання і контролері.

Використовується в ситуації, коли зв'язок між view та іншими частинами програми неможлива (і ви не можете використовувати MVVM або MVP). Частим прикладом використання може служити ASP.NET MVC.

Model-View-Presenter - основною метою даного шаблону є відділення моделі від view. У шаблоні MVP інформація про зміну моделі надходить на презентер (Presenter), який впливає на представлення для оновлення стану.

Використовується в ситуації, коли зв'язування даних неможливо (можна використовувати Binding). Частим прикладом може бути використання Windows Forms.

Model-View-View-Model - даний підхід дозволяє пов'язати елементи view з властивостями і подіями View-моделі. Можна стверджувати, що кожен шар цього патерну не знає про існування іншого шару.

Використовується в ситуації, коли можливо зв'язування даних без необхідності введення спеціальних інтерфейсів view (тобто відсутня необхідність реалізовувати IView). Частим прикладом є технологія WPF.

Ми зупинили свій вибір на виді MVC паттерну Model-View-Controller, як на найбільш простому для використання та подальшого розширення, а також як на найбільг популярному та вживаному на ринку.

2.2 Вибір технологій реалізації веб сервісу

З метою вибрати мову програмування для написання нашого додатку було проаналізовано чотири сучасних бізнес джерела, на основі аналізу яких ми будемо приймати рішення. Це наступні джерела:

- **Pypl**
- **Рейтинг мов програмування 2021- dou.ua**
- **Tiobe**
- **Programming languages used on the job - geekwire.com**

Pypl - це індекс популярності мов програмування, що створюється шляхом аналізу того, як часто здійснюється пошук підручників з певної мови програмування у Google: чим більше запитів на пошук, тим популярнішою вважається мова. Це один з провідних показників популярності мов програмування. Вихідні дані надходять від Google Trends.

Побутує думка, рейтинг Pypl можна вважати “колективною мудрістю”, оскільки пошук відбувається з різних точок світу та різними людьми, а спираючись на його індекси можна навіть приймати рішення, яку мову програмування варто вивчати, або використовувати в новому проекті з розробки програмного забезпечення.

На графіку нижче ми бачимо(рисунок 2.2.1), що серед мов Python, Java та PHP позитивну динаміку демонструє мова Python, яка в 2015 році випередила PHP, а в 2018 році обійшла і Java та стала лідером. Мова Python вже третій рік поспіль тримає лідерство в рейтингу Pypl і продовжує демонструвати зростання.

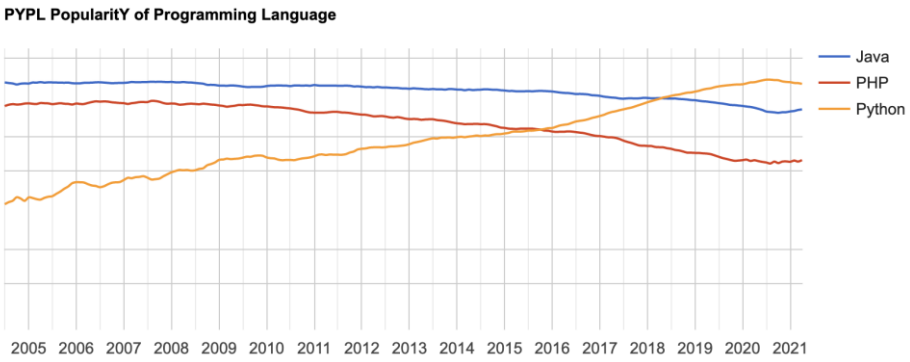


Рисунок 2.2.1 - PYPL Popularity of Programming Language Chart [36]

Рейтинг мов програмування Dou.ua - щорічне опитування щодо мов програмування від сервісу Dou.ua, спираємось на данні 2020 року (вибірка 9747 анкет).

Частка ринку - показує скільки з опитаних респондентів (рисунок 2.2.2) пишуть тією чи іншою мовою програмування. Таким чином топ 5 найпопулярніших мов програмування це: JavaScript, Java, C#, Python та PHP. У свою чергу Python демонструє найбільшу динаміку зростання на ринку.

«Індекс вподобання» — це відношення розробників, що пишуть на мові X, які для наступного проекту у своїй області також оберуть мову X. Ми бачимо, що серед TOP мов на ринку мова Python на другому місці після C#.

№	Мова	Частка ринку	Зміни	Основна	Додаткова	Свої проекти	Індекс вподобання
1	JavaScript	18.4	0.7	1622	4525	2892	0.59
2	Java	15.45	-2.42	1360	1193	1577	0.72
3	C#	13.76		1211	779	1411	0.83
4	Python	13.21	2.3	1163	1802	1802	0.76
5	PHP	10.88	-1	958	805	1112	0.63

Рисунок 2.2.2 - рейтинг мов програмуваня за опитом 2021 dou.ua

Також серед мов, які респонденти планують вивчати наступного року: 20,6% зазначили Python (мова, яка стала лідером опитування), 7,9% - Java, а тих кого цікавить PHP виявилось 2,3%.

На питання, яку мову програмування розробник обрав би якби зараз він починав комерційний проект і у нього була свобода вибору 16,3% респондентів зазначили Python (який став фаворитом), тих хто вказав Java - 13,9%, а PHP вказали - 7,3%.

Отож, ми можемо зробити висновок, що Python це мова яка найдинамічніше розвивається (зростання частки ринку) та мова якою активно цікавляться розробники.

Рейтинг TIOBE - побудований на оцінці результатів пошукових запитів, які містять назву мови програмування. Логіка індексу (рисунок 2.2.3) дуже проста “Якщо мову шукають в пошукових система, значить вона популярна”. Авжеж, таку заяву можна піддати сумніву, оскільки досвідчені програмісти рідко будуть використовувати запити з мовою програмування, але плюсом даного рейтинга можна вважати те, що він досить об’єктивно демонструє інтерес до тієї чи іншої мови програмування.

З 2018 року мова Python демонструє активне зростання пошукових запитів та на сьогодні знаходиться на одному рівні з точки зору інтересу з мовою Java.

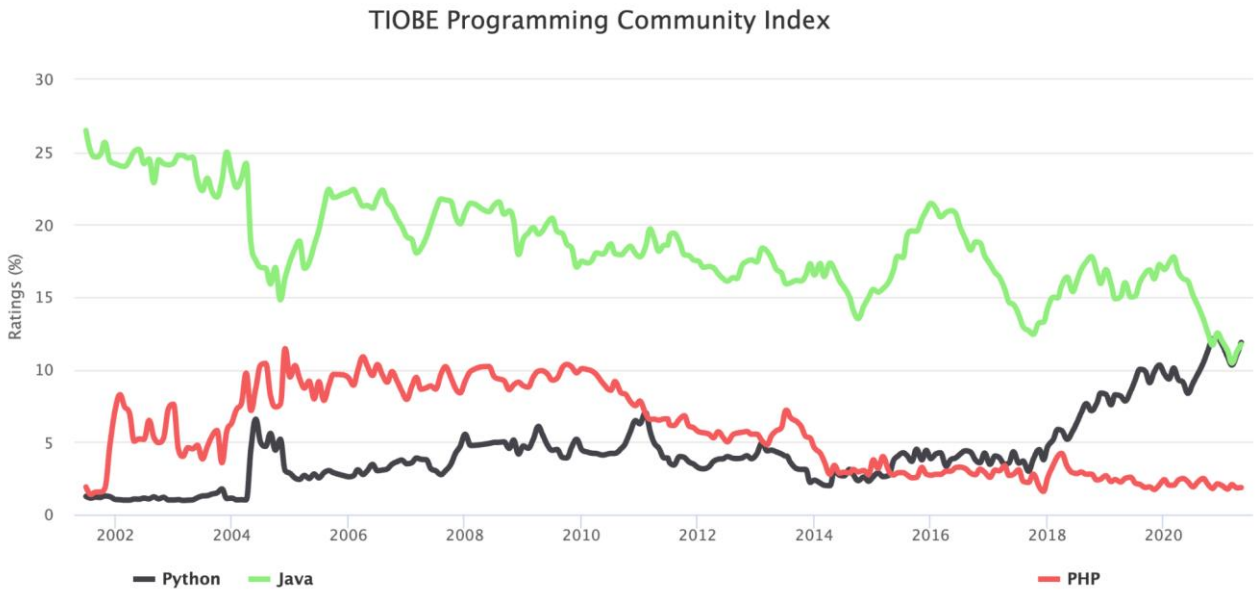


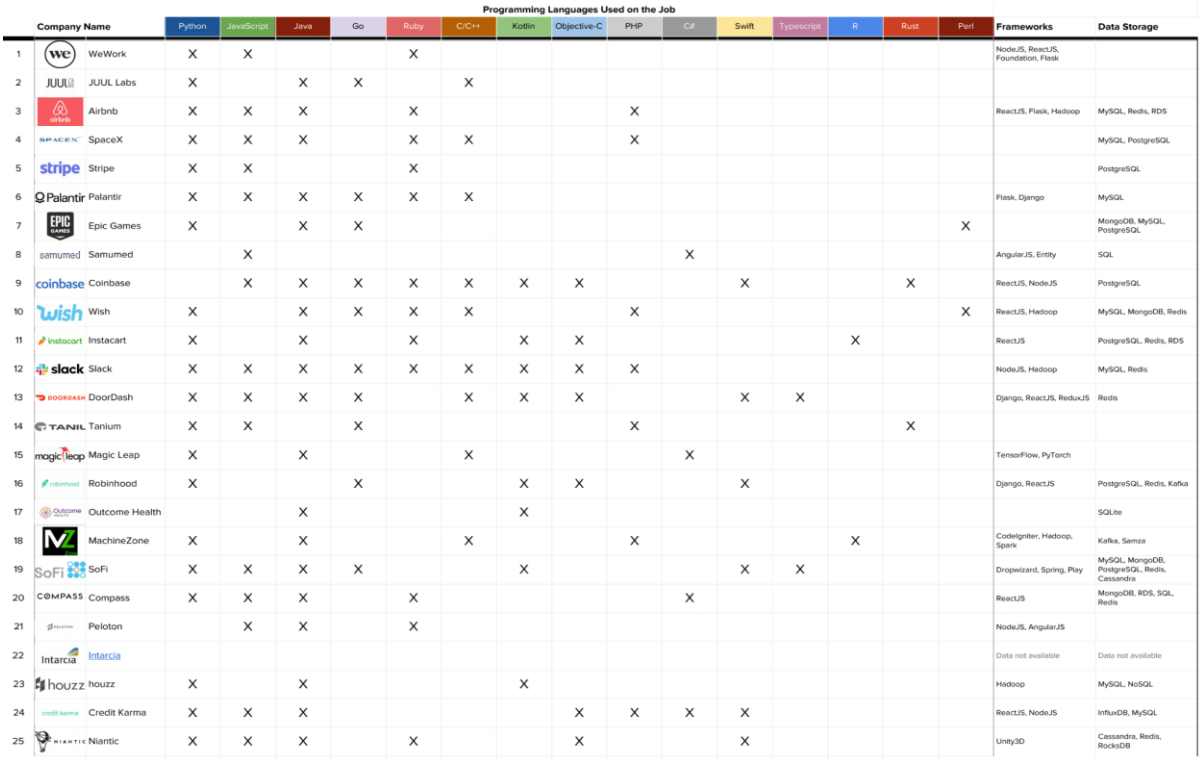
Рисунок 2.2.3 - рейтинг TIOBE [37]

Programming languages used on the job - geekwire.com

Coding Dojo, навчальний курс з програмування в Белвью, Вашингтон, проаналізував дані з сайту Indeed.com, CB Insights, PitchBook і Stackshare, щоб виявити найбільш популярні інструменти, що використовують такі компанії як Uber, Snap, Airbnb, SpaceX тощо.

Більшість компаній використовують від трьох до шести мов. Деякі використовували дві, тоді як Coinbase і DoorDash використовують 10 - "що є свідчення широкомасштабних цілей та інновацій, які переслідують ці компанії", - зазначив Coding Dojo.

Python було визнано найкращою мовою програмування: "Python є загальною за своїм призначенням мовою програмування, а це означає, що її можна використовувати для веб-розробки, скриптів, програмного забезпечення та багато іншого", - зазначив у своєму звіті Coding Dojo. «У наші дні ми бачимо, як Python багато використовують для Data Science, що зробило революцію в технологічній індустрії. Тому є сенс в тому, що Python буде використовуватись дедалі більше».



Programming Languages Used on the Job

	Company Name	Python	JavaScript	Java	Go	Ruby	C/C++	Kotlin	Objective-C	PHP	C#	Swift	Typescript	R	Rust	Perl	Frameworks	Data Storage
1	WeWork	X	X			X											NodeJS, ReactJS, Foundation, Flask	
2	JUUL Labs	X		X	X		X											
3	Airbnb	X	X	X		X				X							ReactJS, Flask, Hadoop	MySQL, Redis, RDS
4	SpaceX	X	X	X		X	X			X								MySQL, PostgreSQL
5	Stripe	X	X			X												PostgreSQL
6	Palantir	X	X	X	X	X	X										Flask, Django	MySQL
7	Epic Games	X		X	X											X		MongoDB, MySQL, PostgreSQL
8	Samumed		X								X						AngularJS, Entity	SQL
9	Coinbase		X	X	X	X	X	X	X			X			X		ReactJS, NodeJS	PostgreSQL
10	Wish	X		X	X	X	X			X						X	ReactJS, Hadoop	MySQL, MongoDB, Redis
11	Instacart	X		X		X		X	X					X			ReactJS	PostgreSQL, Redis, RDS
12	Slack	X	X	X	X	X	X	X	X	X							NodeJS, Hadoop	MySQL, Redis
13	DoorDash	X	X	X	X		X	X	X			X	X				Django, ReactJS, ReduxJS	Redis
14	Tanium	X	X		X					X					X			
15	Magic Leap	X		X			X				X						TensorFlow, PyTorch	
16	Robinhood	X			X			X	X			X					Django, ReactJS	PostgreSQL, Redis, Kafka
17	Outcome Health			X				X										SQLite
18	MachineZone	X		X			X			X				X			CodeIgniter, Hadoop, Spark	Kafka, Samza
19	SoFi	X	X	X	X			X				X	X				Dropwizard, Spring, Play	MySQL, MongoDB, PostgreSQL, Redis, Cassandra
20	Compass	X	X	X		X					X						ReactJS	MongoDB, RDS, SQL, Redis
21	Peloton		X	X		X											NodeJS, AngularJS	
22	Intarcia																Data not available	Data not available
23	houzz	X		X				X									Hadoop	MySQL, NoSQL
24	Credit Karma	X	X	X					X	X	X	X					ReactJS, NodeJS	InfluxDB, MySQL
25	Niantic	X	X	X		X			X			X					Unity3D	Cassandra, Redis, RocksDB

Рисунок 2.2.4 - Programming Language Used on the Job [38]

Виходячи з вище проаналізованих даних та лідируючих позицій мови Python у всіх вищезазначених рейтингах, було прийнято рішення, що використання мови програмування Python є оптимальним для розробки нашого додатку.

2.3 Вибір фреймворку

У сучасному світі, онлайн-додатки грають важливу роль у бізнесі. Тепер підприємства можуть рости та ставати простішими, використовуючи веб-програми, і набагато швидше досягати своїх цілей.

Серверні веб-фреймворки (інакше, "фреймворки веб-додатків") - це програмні фреймворки, які полегшують написання, обслуговування та масштабування веб-програм. Вони надають інструменти та бібліотеки, які спрощують загальні завдання веб-розробки, включаючи маршрутизацію URL-адрес до відповідних обробників, взаємодію з базами даних, підтримку сеансів

та авторизацію користувачів, форматування виводу (наприклад, HTML, JSON, XML) та покращення захисту від веб-атак.

Проаналізуємо три фреймворки - найпопулярніші для кожної з мов програмування, які ми розглянули вище та приймемо фінальне рішення, щодо технологій для розробки нашого додатку:

- PHP - Laravel
- Python - Django
- Java - Spring (Spring Boot Framework)

Laravel - це фреймворк PHP, який широко використовується для створення великих веб-додатків. Він пропонує широкий спектр компонентів веб-побудови, оскільки це повнофункціональний внутрішній фреймворк.

Django - це фреймворк з відкритим кодом, створений за допомогою однієї з найбільш швидкозростаючих мов програмування: Python. Django широко використовується як основа для гнучкої розробки API та високоякісних фонових програм. Фреймворк названий на честь відомого гітариста Джанго Рейнхардта, і був створений в 2003 році. Django підтримує швидку розробку веб-додатків з малою фактичною роботою з кодуванням. Також django має велику кількість різноманітних бібліотек. Популярні сайти, розроблені за допомогою Django, включають Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest and Open stack.[23]

Spring Boot - це заснований на Spring фреймворк, для розробки внутрішнього веб-класу виробничого рівня програми мовою програмування Java. Цей фреймворк готовий до мікропослуг, що дозволяє правильно обробляти запити на різних пристроях і платформах. Спочатку він був випущений у 2002 році. [24] Його сила полягає у створенні більш масштабних додатків за допомогою “хмарного” підходу, але він може також вирішувати і менші проблеми. Це дозволяє паралельним програмам взаємодіяти між собою. Деякі з

них можуть надавати користувальницький інтерфейс, а інші - доступ до бази даних або виконувати іншу роботу. [25]

Spring Boot запускає автоматично налаштовані бібліотеки у Spring, а також сторонні внутрішні фреймворки. Цей фреймворк поставляється з CLI Spring Boot (скорочення від інтерфейсу командного рядка), що дозволяє писати код за допомогою спрощеної мови програмування під назвою Groovy. Spring Boot не вимагає генерації коду та конфігурації XML.

Criterion	Laravel	Ruby On Rails	Django	Spring
1. Customized for large applications	0	0	1	1
2. High scalability	0	0	1	1
3. Adapted to beginners	1	1	1	0
4. Good for businesses applications development	0	0	1	1
5. Rapid prototype development	1	0	1	0
6. A growing Google Trend	1	0	0	0
7. Points by Haker News users	0.5	0.5	1	0.5
8. Points by Reddit users	0.5	0.5	1	0.5
9. Star Rating on GitHub	1	0.5	0.5	0.5
10. The number of Stack Overflow tags	0.5	1	0.5	0.5
TOTAL	5.5	3.5	8	5

Рисунок 2.3.1 - comparison according to the additional criteria

Виходячи з проаналізованої літератури та на основі порівняльного аналізу фреймворків, представленного в таблиці вище (рисунок 2.3.1), було прийнято рішення: що фреймворк Django є оптимальний для використання в розробці нашого додатку оскільки має високу можливість масштабування, є швидким для розробки та зручним для розробки саме бізнес-додатків.

Отож, для написання додатку ми фінально зупинились на мові програмування Python та фреймворку Django.

2.4 Вибір технологій для розробки клієнтської частини

Разом із зростанням використання смартфонів зростають і вимоги клієнтів, щодо додатків якими вони користуються, оскільки останні мають бути зручними на різних девайсах, тому слідуючи сучасним тенденціям нами було прийнято рішення зробити наш веб-додаток адаптивним, для цього ми будемо використовувати Bootstrap 4.

Bootstrap - це інтуїтивно зрозумілий та потужний мобільний інтерфейс, який використовують для швидкого використання та спрощення веб-розробки. Він використовує HTML, CSS та Javascript.

Bootstrap розробили Марк Отто та Джейкоб Торнтон у Twitter. Він був випущений як продукт з відкритим кодом у серпні 2011 року на GitHub.

- Підтримується усіма популярними браузерами
- Лише знання HTML та CSS може отримати кожен
- розпочато з Bootstrap. Також офіційний сайт Bootstrap має хорошу документацію.
- Адаптивний CSS Bootstrap підлаштовується під Настільні ПК, Планшети та мобільні телефони. Детальніше про адаптивний дизайн - у розділі Bootstrap Responsive Design.
- Забезпечує чітке та єдине рішення для побудови інтерфейсу для розробників.
- Він містить красиві та функціональні вбудовані компоненти, які легко налаштувати.

Основуючись на тому, що вище ми обрали мову програмування Python та фреймворк Django, в якості методу для відображення фронтенду ми зупинили свій вибір на Django Template як оптимальному.

Мова шаблонів Django - це власна система шаблонів Django, яка є простим доступним вбудований варіантом для використання.

Додатково використовуємо Bower в якості пекедж-менеджера для завантаження фронтенд бібліотек.

Веб-сайти складаються з багатьох речей - фреймворків, бібліотек, ресурсів та службових програм. Bower керує всіма цими речами за нас. Bower може керувати компонентами, які містять HTML, CSS, JavaScript, шрифти або навіть файли зображень. Bower не об'єднує, не зменшує код і не робить нічого іншого - він просто встановлює потрібні версії необхідних пакетів та їх залежності. Bower оптимізований для інтерфейсу. Якщо кілька пакетів залежать від пакету - наприклад, jQuery - Bower завантажить jQuery лише один раз. Це відоме як плоский графік залежностей, і це допомагає зменшити завантаження сторінки.

2.5 Вибір бази даних

База даних - це організована колекція структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп’ютерній системі. База даних контролюється системою управління базами даних (СУБД). Дані та СУБД разом із пов'язаними з ними програмами називають системою баз даних, або скорочують просто до бази даних.

Дані в найпоширеніших типах баз даних зазвичай моделюють рядки та стовпці в серії таблиць, щоб зробити обробку та запит даних ефективними: до них можна легко отримати доступ, керувати ними, модифікувати, оновлювати, контролювати та організовувати. Більшість баз даних використовують структуровану мову запитів (SQL) для запису та запиту даних.

Існує багато різних типів баз даних. Найкраща база даних для конкретної організації залежить від того, яка задача стоїть перед нею та які дані будуть зберігатись, розглянемо можливі варіанти нижче:

Реляційні бази даних - стали домінуючими у 1980-х. Елементи реляційної бази даних організовані як набір таблиць зі стовпцями та рядками. Технологія

реляційних баз даних забезпечує найбільш ефективний та гнучкий спосіб доступу до структурованої інформації.

Об’єктно-орієнтовані бази даних - інформація представлена у вигляді об’єктів, як і в ООП.

Розподілені бази даних - складається з двох або більше файлів, розташованих на різних сайтах. База даних може зберігатися на декількох комп'ютерах, розташованих в одному фізичному місці або розкиданих по різних мережах.

Сховища даних - це тип бази даних, спеціально розроблений для швидких запитів та аналізу.

Бази даних NoSQL, або нереляційна база даних, дозволяє зберігати та маніпулювати неструктурованими та напівструктурованими даними (на відміну від реляційної бази даних, яка визначає, яким чином повинні бути складені всі дані, що введені в базу даних). Бази даних NoSQL стали популярнішими, оскільки веб-програми стали більш складними.

Графічні бази даних - зберігають дані в термінах сутностей та взаємозв’язків між сутностями.

Бази даних OLTP - це швидка, аналітична база даних, призначена для великої кількості транзакцій, що виконуються декількома користувачами.

Спираючись на те що для розробки нашого додатку ми обрали фреймворк Django, розглянемо які саме бази даних він підтримує[27]. Отож, на момент написання роботи в документації Django вказано, що фреймворк підтримує наступні бази даних:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

PostgreSQL - це об'єктно-реляційна система управління базами даних з відкритим кодом. В даний час PostgreSQL пропонує такі функції, як складні запити, зовнішні ключі, тригери, перегляди, цілісність транзакцій, повнотекстовий пошук та обмежена реплікація даних. Користувач може розширити PostgreSQL за допомогою нових типів даних, функцій, операторів або методів індексування.

PostgreSQL підтримує різноманітні мови програмування (включаючи C, C ++, Java, Perl, Tcl та Python), а також інтерфейси бази даних JDBC та ODBC.

MySQL - це система управління реляційними базами даних із відкритим кодом (СУБД).

MariaDB - це система управління реляційними базами даних створена як відгалуження MySQL (RDBMS).

Oracle (зазвичай її називають СУБД Oracle або просто Oracle) - це багатомодельна [28] система управління базами даних, що розроблена та продається корпорацією Oracle.

SQLite - це вбудована реляційна база даних з відкритим кодом. Перший реліз вийшов у 2000 році, щоб забезпечити зручний спосіб для програм керувати даними без додаткових витрат, що часто постачається із спеціальними реляційними системами управління базами даних. SQLite має гарну репутацію, проста у використанні, ефективна та надійна.

Оскільки фреймворк Django краще за все підтримує базу даних Postgresql, тому саме її ми обрали для розробки додатку.

3. РОЗРОБКА І РЕАЛІЗАЦІЯ ВЕБ ДОДАТКУ ДЛЯ ОНЛАЙН ЗАМОВЛЕНЬ

3.1. Проектування бази даних

В результаті проектування було виявлено наступні сутності:

Клієнт (Client):

- Ім’я
- Прізвище
- Адрес - для автозаповнення при оплаті замовлення
- Електронна пошта
- Номер телефону - використовується для входу в систему;
- Доступ до адмін панелі - використовується для працівників ресторанів
- Активний чи неактивний статус клієнта - використовується для включення та відключення від системи
- Дата реєстрації
- Дата останньої зміни

Ресторан (Business):

- Назва
- Зображення логотипу
- Зображеня для хедера
- Опис ресторану
- Активний чи неактивний статус користувача зі сторону ресторану - використовується для включення та відключення від системи
- Дата реєстрації
- Дата останньої зміни

Позиція (SKU):

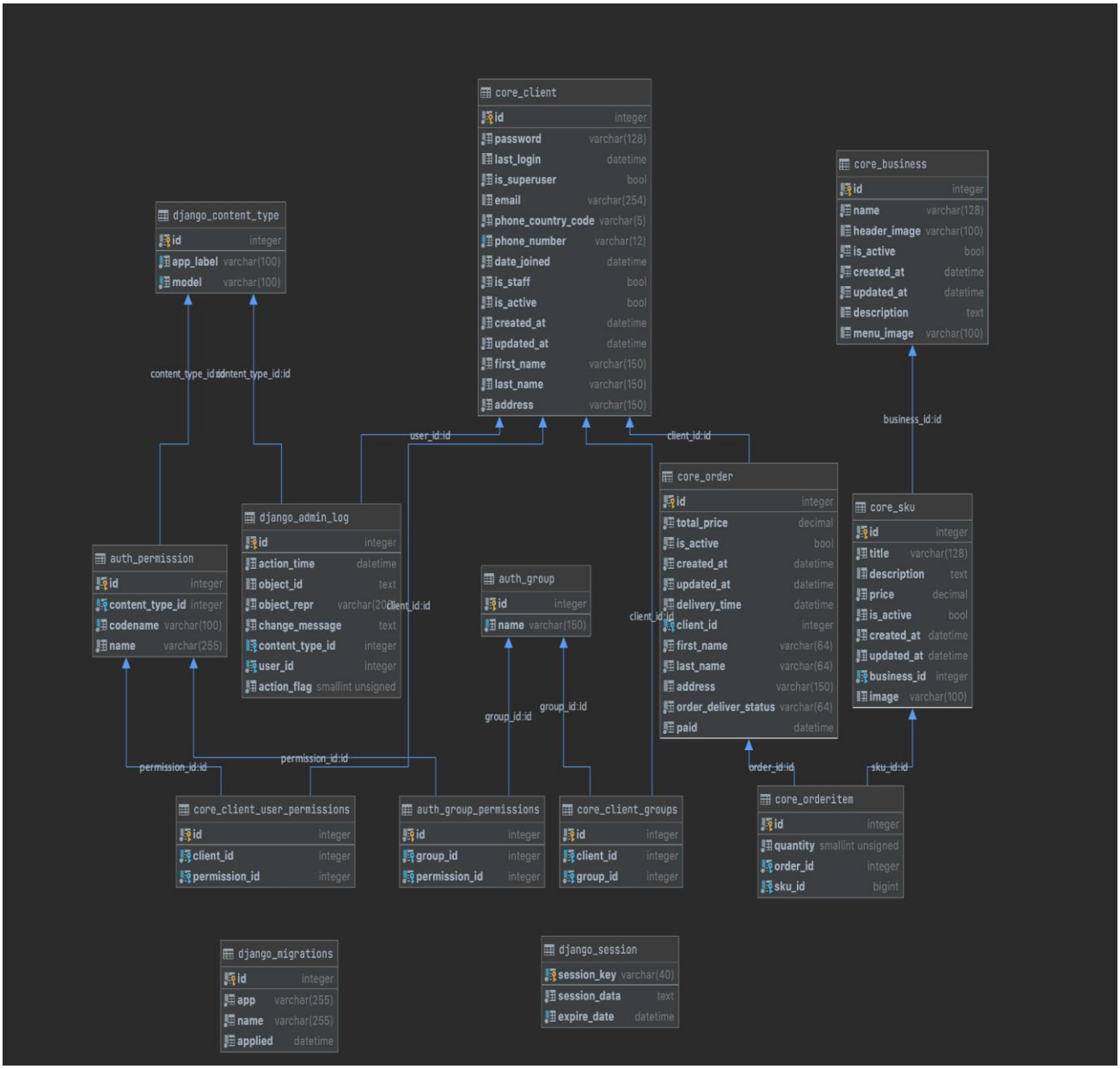
- Ресторан (Business)
- Зображення позиції
- Заголовок
- Опис
- Ціна
- Активний чи неактивний статус позиції - використовується для включення та відключення від системи
- Дата створення
- Дата останньої зміни

Позиція замовлення (OrderItem)

- Позиція (SKU)
- Замовлення (Order)
- Кількість

Замовлення (Order):

- Клієнт (Client)
- Ім'я отримувача
- Прізвище отримувача
- Адрес доставки
- Час доставки - на котру годину та день доставити замовлення
- Сума замовлення
- Статус доставки - використовується для відслідковуванням на якому етапі замовлення
- Оплата
- Активний чи неактивний статус замовлення - використовується для включення та відключення від системи
- Дата створення
- Дата останньої зміни


core_client
id integer
password varchar(128)
last_login datetime
is_superuser bool
email varchar(254)
phone_country_code varchar(5)
phone_number varchar(12)
date_joined datetime
is_staff bool
is_active bool
created_at datetime
updated_at datetime
first_name varchar(150)
last_name varchar(150)
address varchar(150)
core_business
id integer
name varchar(128)
header_image varchar(100)
is_active bool
created_at datetime
updated_at datetime
description text
menu_image varchar(100)
django_content_type
id integer
app_label varchar(100)
model varchar(100)
content_type_id:id
content_type_id:id
user_id:id
client_id:id
business_id:id
core_order
id integer
total_price decimal
is_active bool
created_at datetime
updated_at datetime
delivery_time datetime
client_id integer
first_name varchar(64)
last_name varchar(64)
address varchar(150)
order_deliver_status varchar(64)
paid datetime
core_sku
id integer
title varchar(128)
description text
price decimal
is_active bool
created_at datetime
updated_at datetime
business_id integer
image varchar(100)
django_admin_log
id integer
action_time datetime
object_id text
object_repr varchar(200)
change_message text
content_type_id integer
user_id integer
action_flag smallint unsigned
auth_permission
id integer
content_type_id integer
codename varchar(100)
name varchar(255)
auth_group
id integer
name varchar(150)
client_id:id
client_id:id
group_id:id
group_id:id
permission_id:id
permission_id:id
order_id:id
sku_id:id
core_client_user_permissions
id integer
client_id integer
permission_id integer
auth_group_permissions
id integer
group_id integer
permission_id integer
core_client_groups
id integer
client_id integer
group_id integer
core_orderitem
id integer
quantity smallint unsigned
order_id integer
sku_id bigint
django_migrations
id integer
app varchar(255)
name varchar(255)
applied datetime
django_session
session_key varchar(40)
session_data text
expire_date datetime

3.2 Початкові налаштування роботи системи

Django працює як з версіями Python 2.7, так і з версією 3.x+. В даній роботі будемо використовувати Python версії 3.9.0. - це одна з останніх версій Python на момент написання роботи. Для встановлення Python було створено директорію, а через термінали ми створюємо віртуальне оточення, яке буде використовуватись лише для нашого проекта. Для встановлення всіх додаткових пакетів в Python використовується PIP (Pip Installs Packages). PIP уже включено в версії Python від 2.x - 3.9.0. За допомогою PIP в Python встановлюються пакети, які можливо знайти через термінал ./pip search [package-name], або через веб-сайт https://pypi.org.

Знаходячись в активному віртуальному оточенні наступним кроком буде встановлення Django та psycopg2-binary(для використання PostgreSQL). Після завершення установки пакету можемо створити проект через команду:

```
mkdir restaurant_automation
python -m venv .env
source .env/bin/activate
pip install django psycopg2-binary
django-admin startproject restaurant_automation
```

Після виконання буде створена директорія проекту в середині якої структура мінімального проекту для запуску Django.

```
├── manage.py
└── restaurant_automation
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

Нижче представлено більш детальний опис файлів:

- **manage.py** - утиліта для терміналу (командний рядок), яка дозволяє взаємодіяти з нашим проектом багатьма способами;
- **restaurant_automation/__init__.py** - пустий файл, який вказує Python на те, що дана директорія являється Python пакетом;

- **restaurant_automation/settings.py** - конфігураційний файл для даного проекту, з його допомогою Django визначає базу даних, що використовується, параметри підключення до бази даних, підключення до проекту тощо;
- **restaurant_automation/urls.py** - елегантна схема маршрутизації URL-адреси, яка визначає яка саме функція буде виконувати обробку запиту, в деякому сенсі це “зміст” майбутнього додатку.
- **restaurant_automation/wsgi.py** - вхідна точка для WSGI-поєднуваних веб серверів, що переадресовує запити до веб-додатків

Тепер створимо необхідний додаток:

```
django-admin startapp core
```

Після чого приєднуємо додаток до Django за допомогою restaurant_automation/settings.py:

```
INSTALLED_APPS = [
      ...
    "core",
]
```

Також було доповнено restaurant_automation/core/urls.py з метою увімкнення адмін панелі Django та доступу до шаблонів додатку “core”.

```
urlpatterns = [
   path("admin/", admin.site.urls),
   path("", include("core.urls", namespace="core")),
]
```

Слідом приєднуємо базу даних до нашого додатку в конфігураційному файлі restaurant_automation/settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'restaurant_automation',
        'USER': '',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Також додано налаштування для статичних файлів та бібліотек завантажених за допомогою bower.

```
STATICFILES_DIRS = (
   os.path.join(BASE_DIR, 'static'),
   os.path.join('bower_components'),
)
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static_files/')
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')
```

3.3 Реалізація авторизації та реєстрації

Система автентифікації Django обробляє як автентифікацію, так і авторизацію. Коротко кажучи, автентифікація підтверджує, що користувач є “реальний, справжній” та є тим, ким він претендує бути, а авторизація визначає, що дозволено виконувати автентифікованому користувачеві[29].

Система автентифікації в Django дуже проста та загальна (django.contrib.auth.urls), модель складається з:

- логін
- пароль
- електронна пошта

Оскільки для цілей нашого додатку базового рішення автентифікації, який пропонує Django виявилось недостатньо, було прийнято рішення розробити власну модель користувача на основі уже існуючої у Django. Таке вдосконалення нам вдалося за рахунок того, що Django використовує ООП, що дозволило нам використати поліморфізм та успадкувати абстрактну модель користувача (AbstractBaseUser)[30] базової моделі Django та доповнити її наступними полями:

- адреса
- номер телефону

Для зручності користувача було прийнято рішення також оптимізувати та зробити більш сучасним і процес аутентифікації, а саме: змінити модель “юзернейм -> пароль” на модель “номер телефону -> пароль”, оголосивши змінну USERNAME_FIELD = “phone_number”[31], тим самим перевизначивши її за замовчуванням.

```
class Client(AbstractBaseUser, PermissionsMixin):
    first_name = models.CharField('first name', max_length=150, blank=True)
    last_name = models.CharField('last name', max_length=150, blank=True)
    address = models.CharField("Address", max_length=150, blank=True)
    email = models.EmailField(blank=True, null=True)
    phone_country_code = models.CharField(max_length=5, default="+38")
    phone_number = models.CharField(max_length=12, blank=True, unique=True)

    USERNAME_FIELD = "phone_number"
    REQUIRED_FIELDS = []

    date_joined = models.DateTimeField(default=timezone.now)
    is_staff = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    objects = CustomUserManager()

    def __str__(self):
        return self.get_full_name()

    def get_full_name(self):
        return f"{self.first_name} {self.last_name}".strip()
```

Для коректного відображення об’єкту користувача було використано магічні методи в Python [32] - це спеціальні методи, які починаються і закінчуються подвійними підкресленнями. Їх ще називають методами дундера. Магічні методи не призначені для виклику безпосередньо, а виклик відбувається внутрішньо від класу за певної дії.

В описаній моделі користувача було використано магічний метод __str__(). Що відповідає за повернення об’єкту моделі у рядковому представленні.

Для кастомізації відображення і поведінки аутентифікації ми створюємо html шаблон (рисунок 3.3.1) core/templates/registration/login.html.

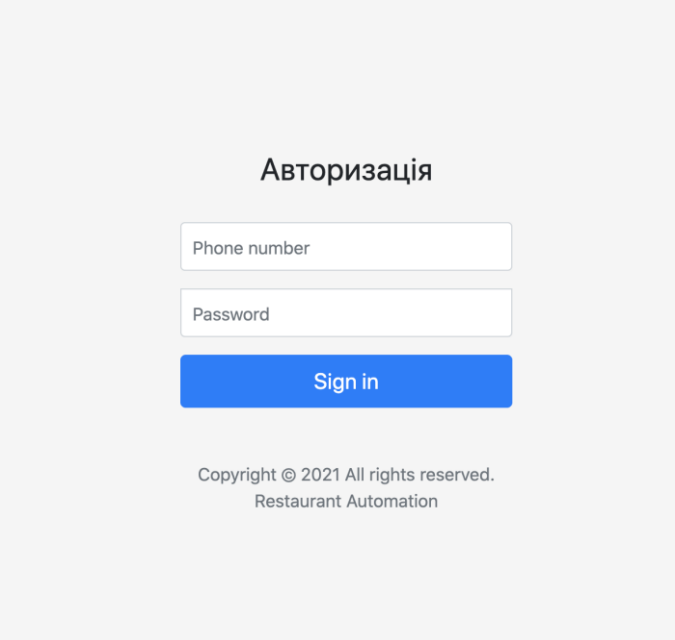


Рисунок 3.3.1 - відображення авторизації

Підключаємо авторизацію через додаток core/urls.py та переходимо до процесу реєстрації.

```
urlpatterns = [
    # Authorization
    path("", include("django.contrib.auth.urls")),
]
```

Реєстрація користувача - це створення облікового запису, який буде зберігатися на сервері і за допомогою якого користувач може заходити в систему і робити замовлення (рисунок 3.3.2).

Описуємо компонент реєстрації та підключаємо до неї форму з наступними полями:

- phone_number
- first_name
- last_name
- address

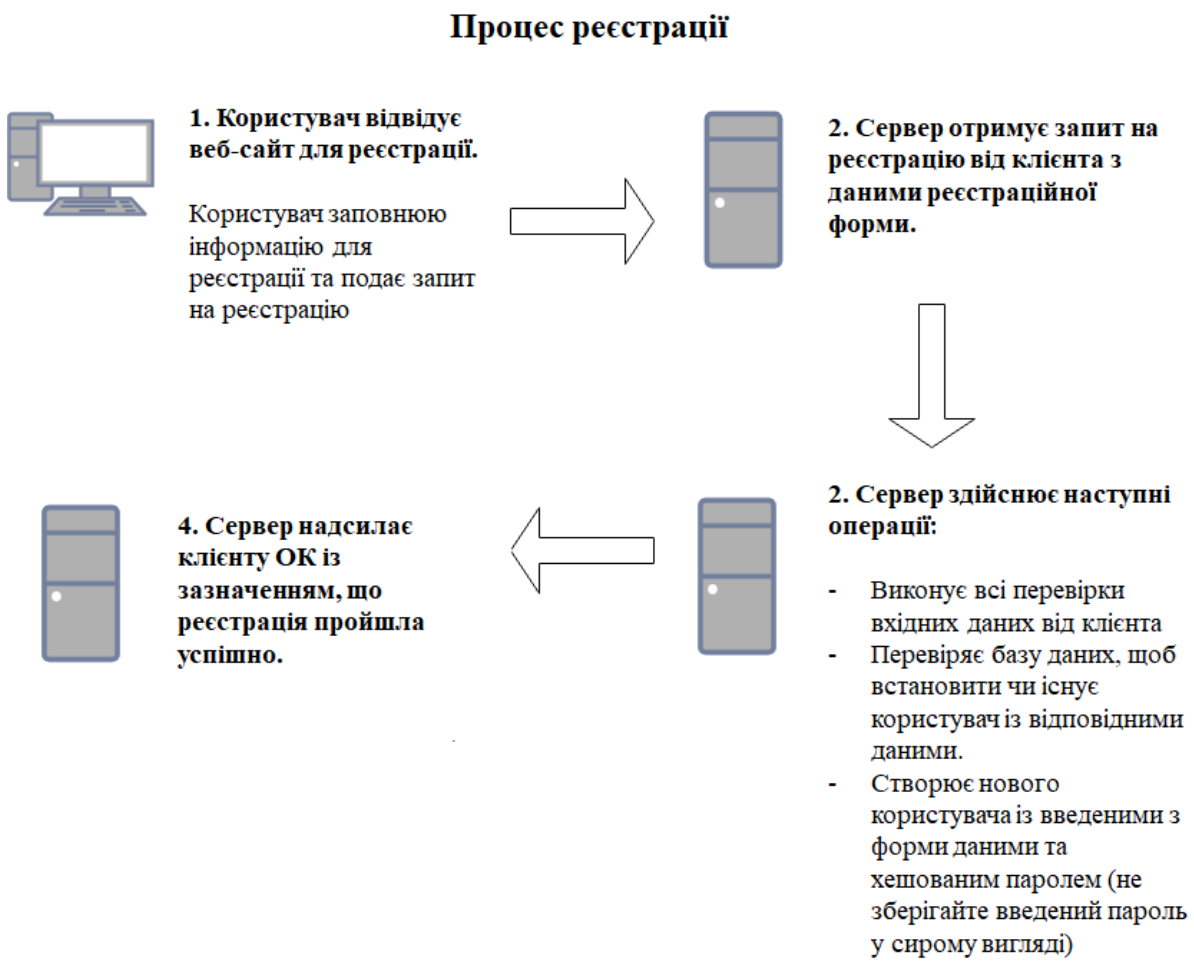


Рисунок 3.3.2 - процес реєстрації

Реалізуємо View реєстрації та зареєструємо її в core/urls.py, та додаємо шаблон у core/templates/registration/signup.html.

```
def signup(request):
    """Registration form."""
    if request.user.is_authenticated:
        return redirect("core:home")

    if request.method == "POST":
        form = ClientCreationForm(request.POST)
        if form.is_valid():
            form.save()
            phone_number = form.cleaned_data.get("phone_number")
            raw_password = form.cleaned_data.get("password1")
            user = authenticate(phone_number=phone_number, password=raw_password
            login(request, user)
            return redirect("core:home")
    else:
        form = ClientCreationForm()
    return render(request, "registration/signup.html", {"form": form})
```

Зовнішній вигляд поля реєстрації (рисунок 3.3.3):

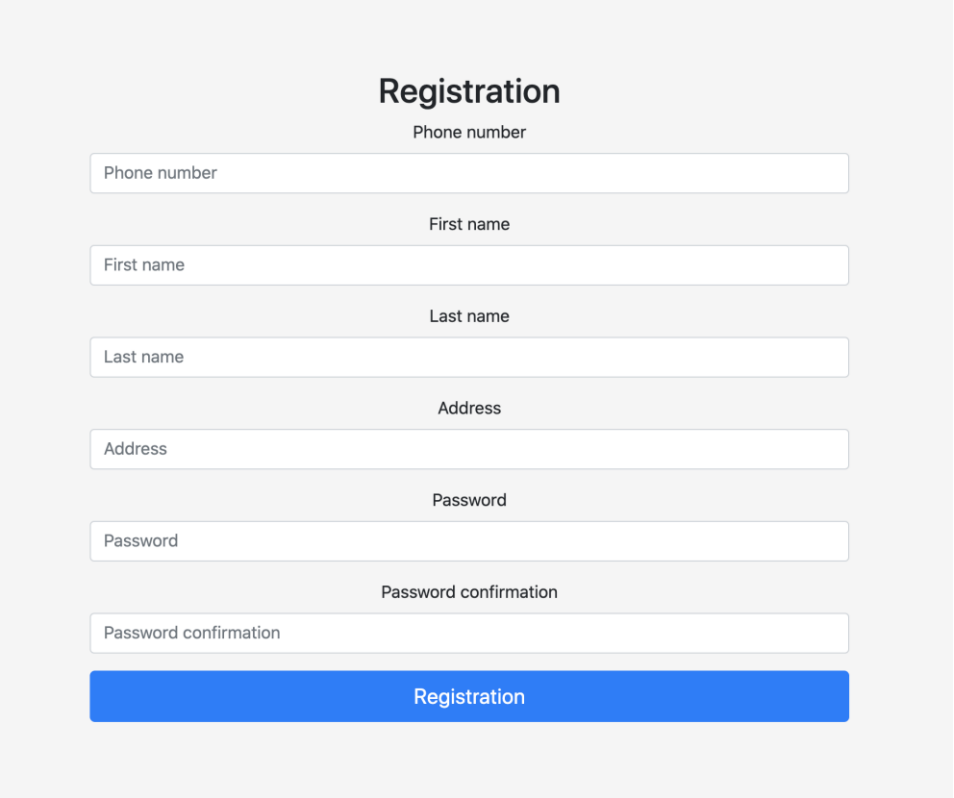


Рисунок 3.3.3 - Реєстрація

Для зручності додамо модель клієнта в панель адміністратора, таким чином адміністратор має змогу переглядати активних клієнтів(рисунок 3.3.4):

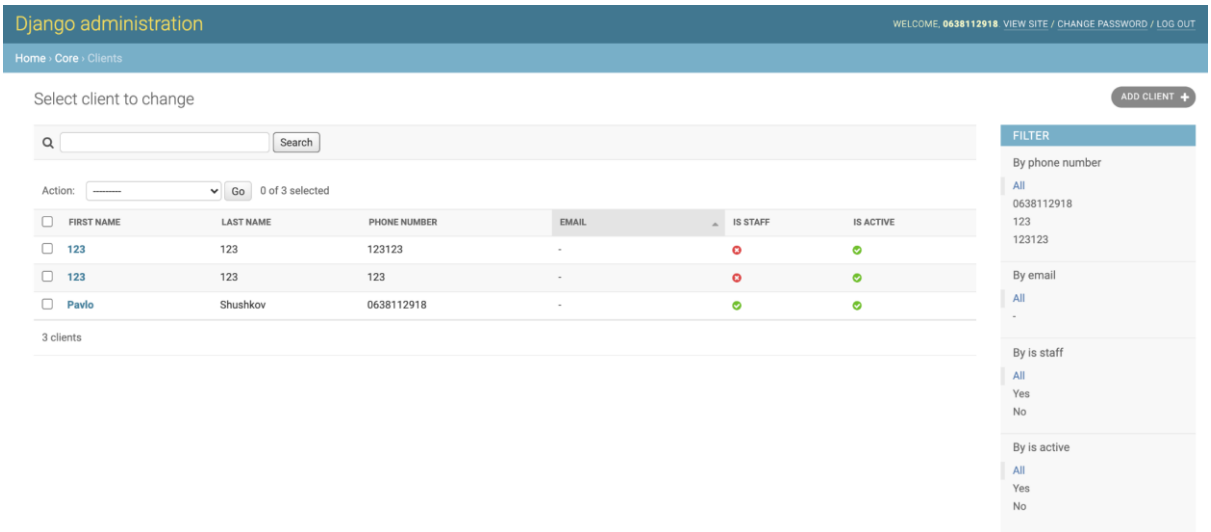


Рисунок 3.3.4 - Адміністративна панель

3.4 Реалізація відображення меню ресторану

Компонент меню ресторану відповідає за наступні функції:

- Створення/Редагування/Видалення ресторану у системі.
- Створення/Редагування/Видалення позицій для кожного ресторану у системі.
- Відображення списку ресторанів
- Відображення меню ресторану

Для виконання вищезазначених функцій буде використано django admin та відображення на фронтальній частині

Опишемо модель:

```
class Business(models.Model):
    name = models.CharField(max_length=128)
    header_image = models.FileField(blank=True, null=True)
    menu_image = models.FileField(blank=True, null=True)
    description = models.TextField(blank=True, null=True)

    # Service information
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name


class SKU(models.Model):
    business = models.ForeignKey(Business, on_delete=models.CASCADE)

    image = models.FileField(blank=True, null=True)
    title = models.CharField(max_length=128)
    description = models.TextField(blank=True, null=True)
    price = models.DecimalField(decimal_places=2, max_digits=15)

    # Service information
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.title} | {self.business}"
```

Django має вже реалізовану зручну та гнучку адміністративну панель, що обумовило її використання у проекті без змін. Адміністративну панель було оголошено “core/admin.py”.

```
@admin.register(Business)
class BusinessAdmin(admin.ModelAdmin):
    pass


@admin.register(SKU)
class SKUAdmin(admin.ModelAdmin):
    pass
```

На виході ми отримали наступний варіант візуалізації адміністративної панелі, що дозволяє виконувати CRUD операцій [34]:

1. Список ресторанів (рисунок 3.4.1):

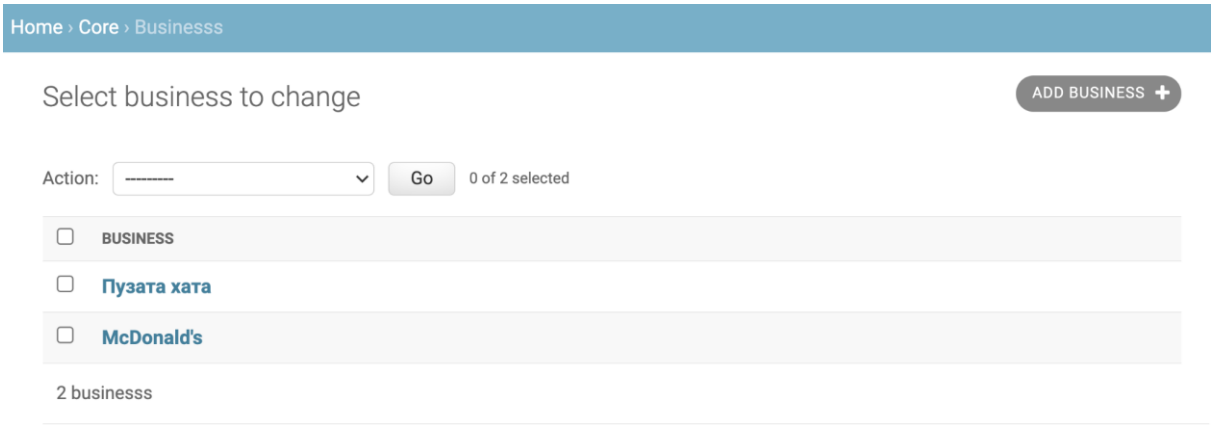


Рисунок 3.4.1 - Адміністративна панель зі списком ресторанів

2. Операції CRUD з моделью Business (рисунок 3.4.2):

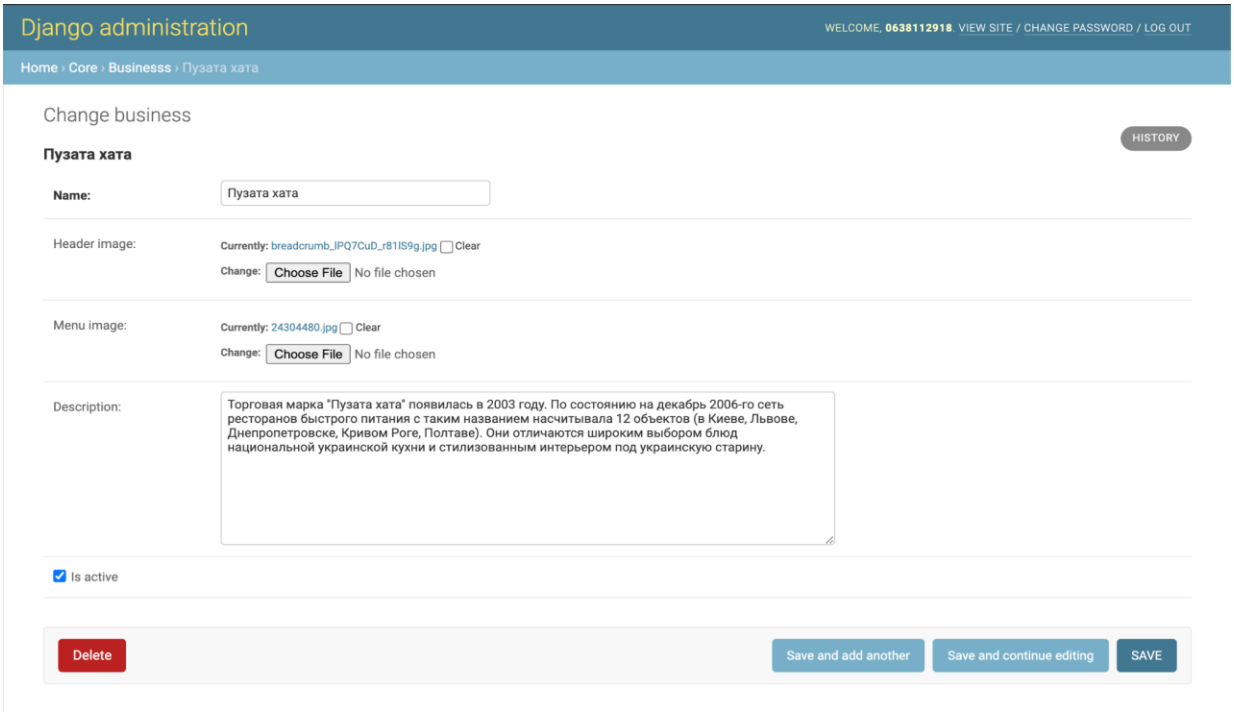


Рисунок 3.4.2 - Адміністративна панель з редагуванням ресторану

3. Список позицій ресторану:

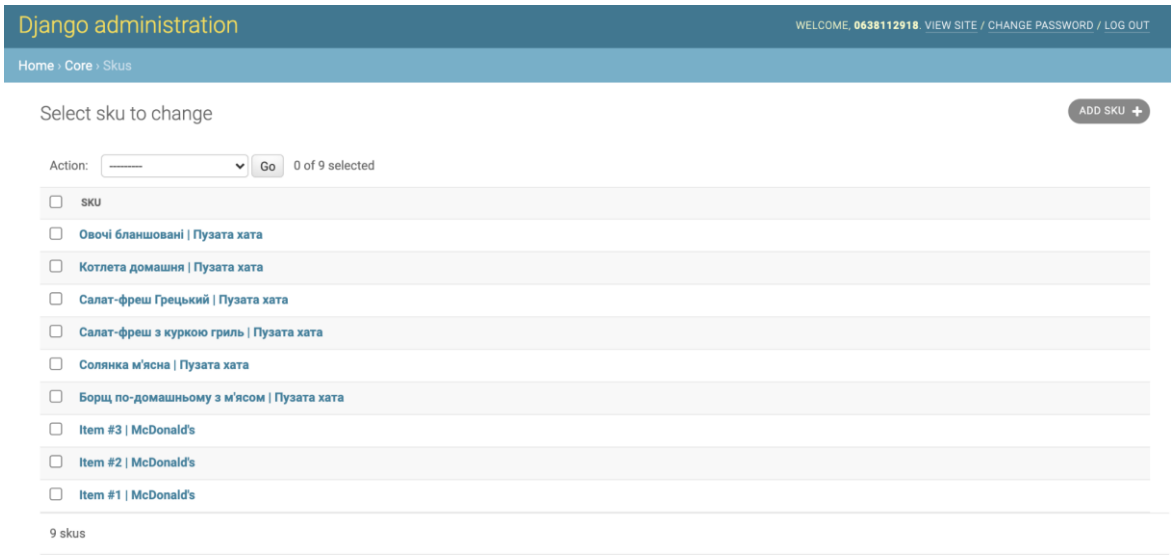


Рисунок 3.4.3 - Адміністративна панель з списком позицій

4. Операції CRUD з моделью SKU(рисунок 3.4.4):

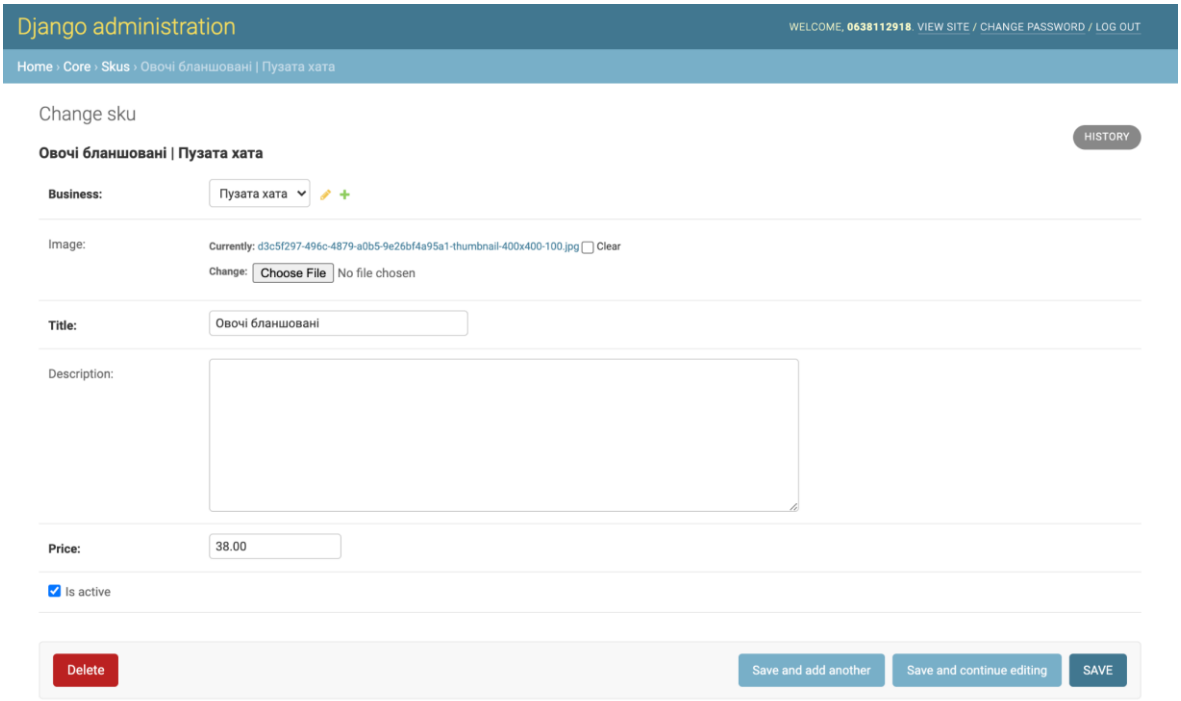


Рисунок 3.4.4 - Адміністративна панель з редагуванням позиції

Для відображення ресторанів використовуємо вбудовані в Django View - ListView[35], DetailView[36], які створені для того щоб спростити роботу програмістів.

```
class MainView(ListView):
    model = Business
    template_name = "main_screen.html"
    paginate_by = 10

class SKUListView(DetailView):
    model = Business
    template_name = "business/sku_list.html"
```

Нижче наведено приклад відображення для клієнта спику ресторанів (рисунок 3.4.5) та списку позицій (рисунок 3.4.6)

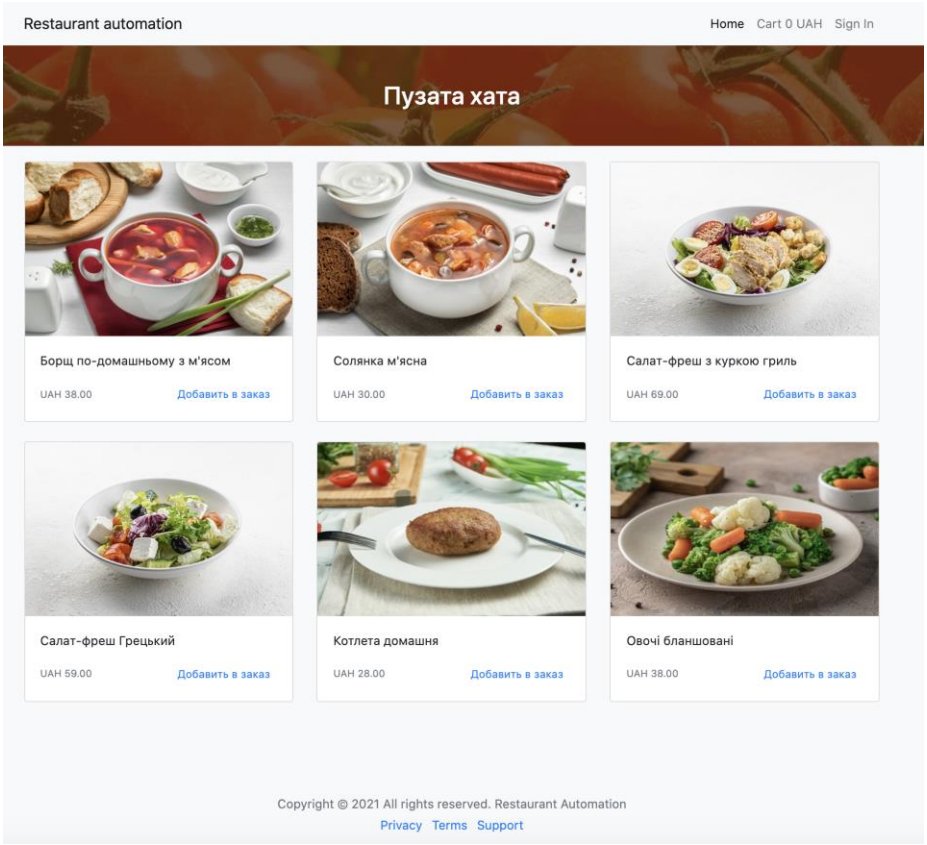


Рисунок 3.4.5 - Адміністративна панель з редагуванням позиції

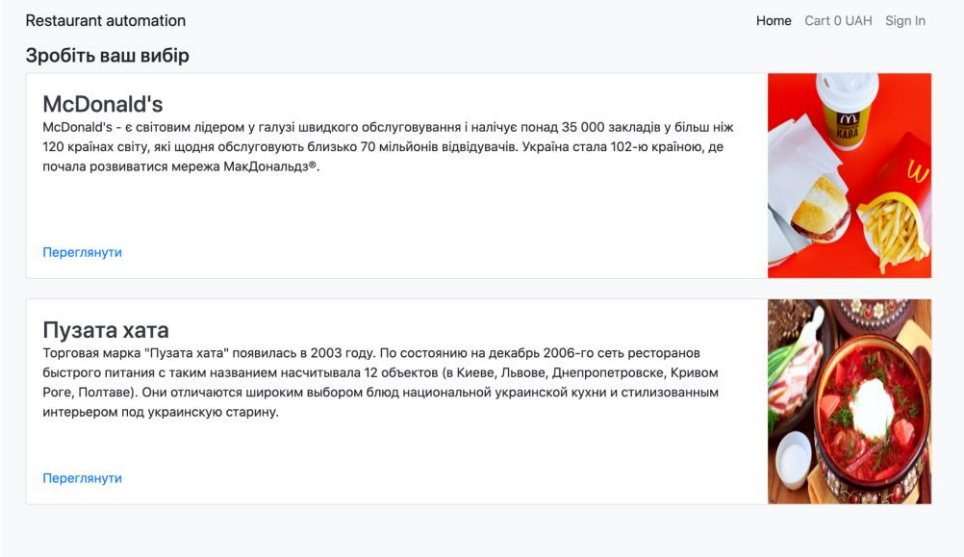


Рисунок 3.4.6 - Адміністративна панель з редагуванням позиції

3.5 Реалізація компонента замовлення

Даний компонент відповідає за наступні функції:

- Кошик створення/редагування/видалення
- Екран оплати замовлення

Кошик - це фіча, яка дозволяє користувачу збирати замовлення на кшталт візка у супермаркеті. Вона дозволяє включити усі позиції які хоче клієнт та інформативно відображати кількість та вартість зібраних позицій.

Для її реалізації було використано додаткові технології(рисунок 3.4.1), а саме AJAX та JavaScript method fetch, які допомагають відправляти запити до веб-додатку у фоновому режимі, швидко отримувати відповіді (без додаткового оновлення сторінки) та модифікувати сторінку, що в свою чергу економить час клієнта та робить додаток динамічним та зручним.

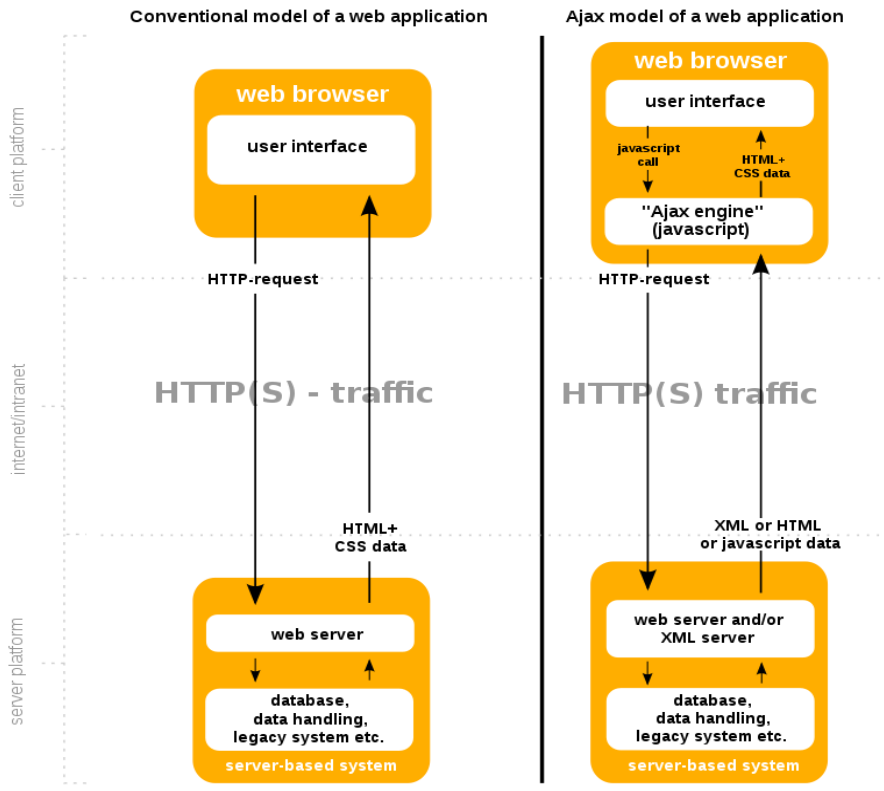


Рисунок 3.4.1 - Порівняння простою моделі та AJAX

Отож, створюємо окремий endpoint в стилі REST на який від клієнта надходять запити AJAX, що згідно з HTTP мають один з методів. Запити з відповідним методом посилаючись на REST отримують наступну реакцію системи:

- GET - отримання оновленої частини html коду;
- POST - додавання позиції до кошику;
- DELETE - видалення позиції з кошику.

Зі сторони бекенду ми зберігаємо дані за допомогою django session, що в свою чергу зберігає дані у базі даних.

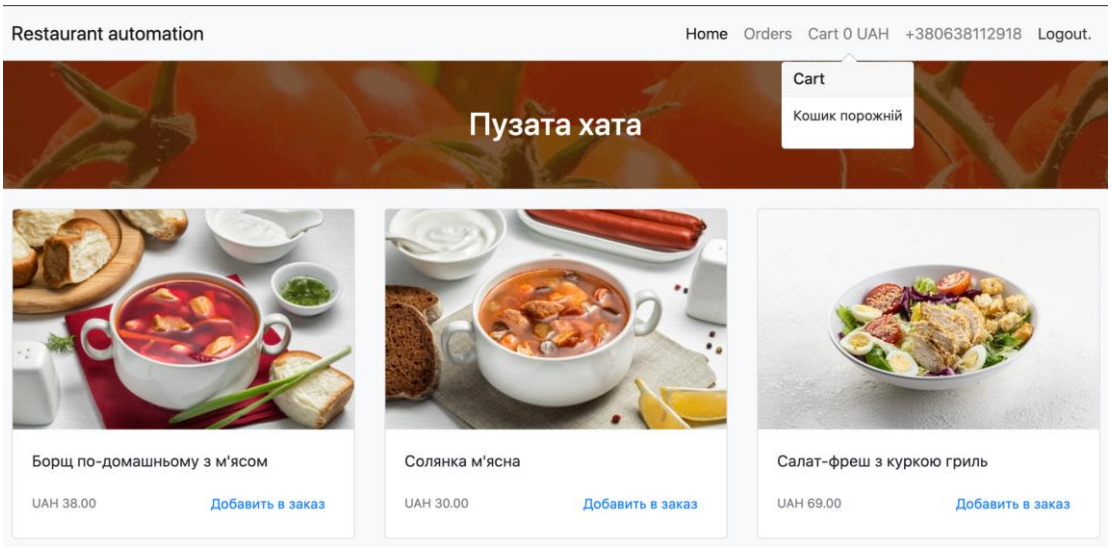


Рисунок 3.5.1 - Відображення меню

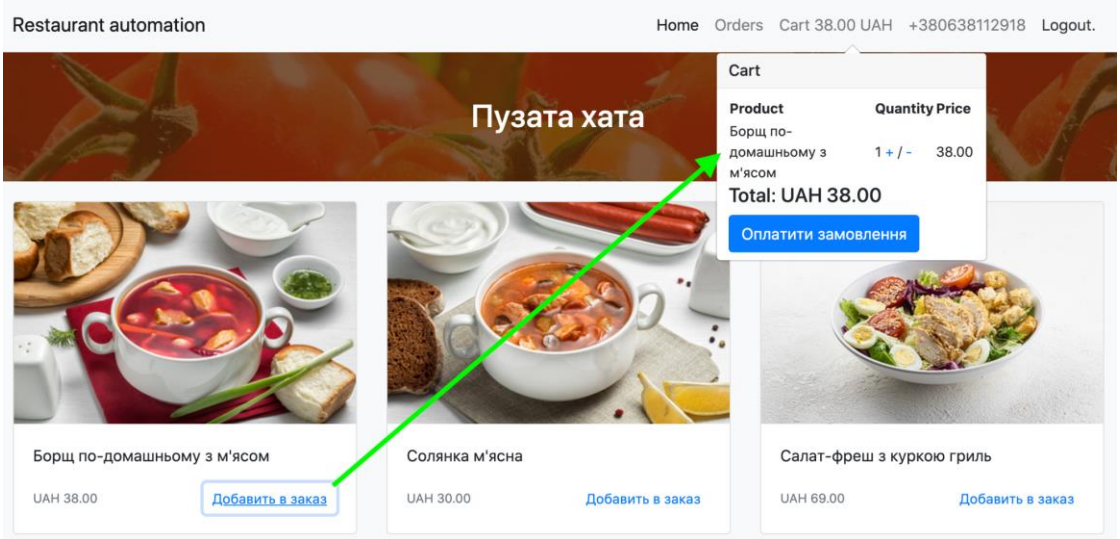


Рисунок 3.5.2 - Приклад реалізації замовлення в корзині

Екран оплати замовлення

На сторінці замовлення розташовані наступні блоки (рисунок 3.5.3)

1. **Кошик:** містить інформацію щодо переліку продукції до замовлення, її вартості у розрізі позицій та загальної вартості замовлення. Формується автоматично.

2. **Адреса для доставки:** складається з імені та прізвища замовника, адреси для доставки та поля з вибором часу доставки. Заповнюється замовником.

3. **Оплата:** блок для оплати карткою складається з номеру картки та деталей по картці необхідних для оплати. Заповнюється замовником.

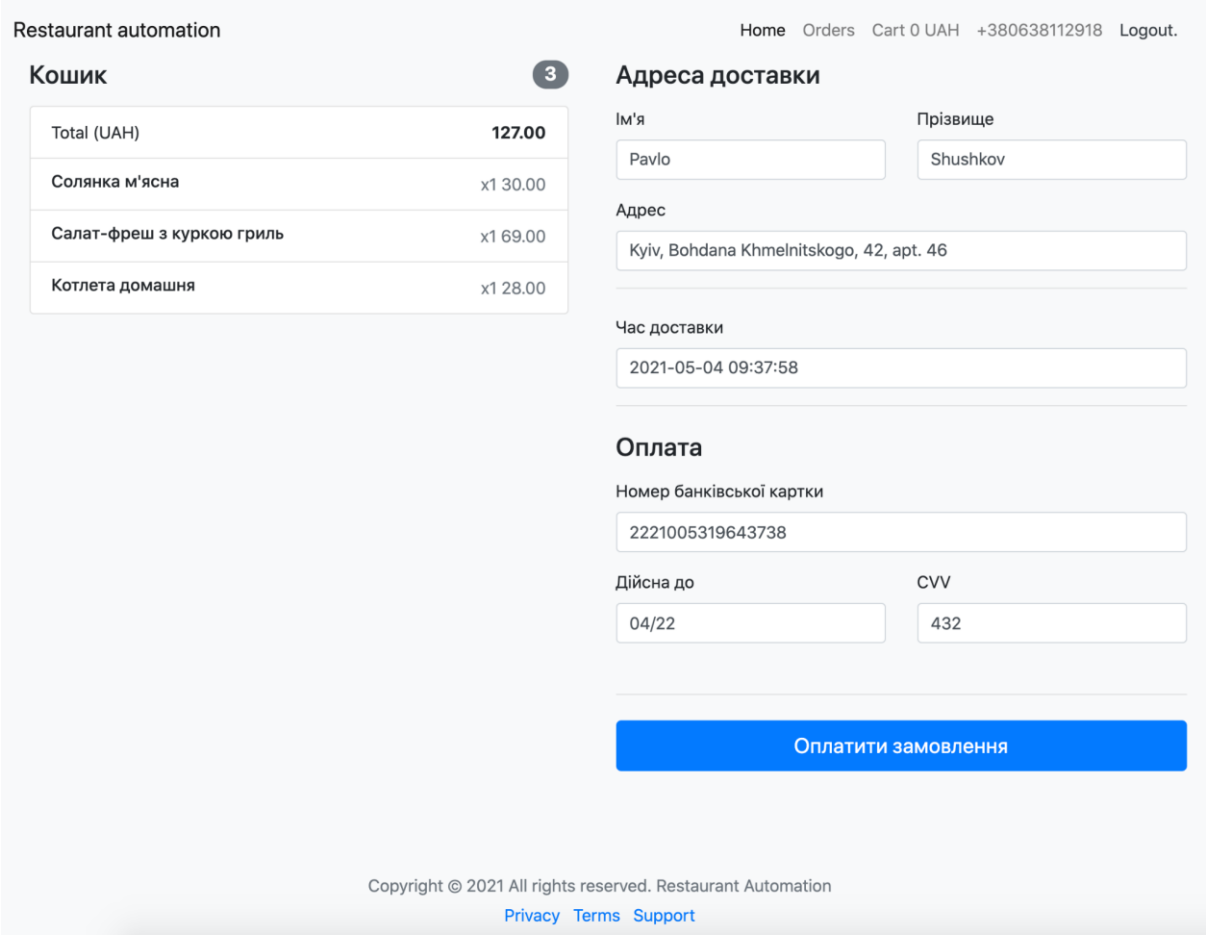


Рисунок 3.5.3 - Приклад оформлення замовлення

Оплати замовлення має свої маршрути та свій модуль контролерів.

```
urlpatterns = [
    # Main
    path("", views.MainView.as_view(), name="home"),
    path("business/<int:pk>/", views.SKUListView.as_view(),
name="business_sku_list"),
    path("checkout/", views.Checkout.as_view(), name="checkout"),

    # Cart API
    path("cart/", views.cart_api, name="cart_add"),

    # Client
    path("orders/", views.OrdersListView.as_view(), name="orders_list"),
    path("orders/<int:pk>/", views.OrdersDetailView.as_view(), name="orders_deta
]
```

Приклад View оплати замовлення.

```
class Checkout(LoginRequiredMixin, FormView):
    model = Order
    queryset = Order.objects.all()
    form_class = forms.ClientOrderForm
    template_name = 'order.html'
    success_url = "/main/"

    def __init__(self, *args, **kwargs):
        super(Checkout, self).__init__(*args, **kwargs)
        self.cart = None

    def get_form_kwargs(self):
        kwargs = super(Checkout, self).get_form_kwargs()
        kwargs["cart"] = self.cart
        kwargs["request"] = self.request
        return kwargs

    def dispatch(self, request, *args, **kwargs):
        self.cart = Cart(self.request.session)
        return super(Checkout, self).dispatch(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        context = super(Checkout, self).get_context_data(**kwargs)
        context.update({"cart": self.cart, "request": self.request})
        return context

    def get_initial(self):
        initial = super(Checkout, self).get_initial()
        if self.request.user.is_authenticated and self.request.user:
            initial.update(
                {
                    "first_name": self.request.user.first_name,
                    "last_name": self.request.user.last_name,
                    "address": self.request.user.address,
                }
            )
        return initial
```

Базовий шаблон сторінки

```
{% extends "base.html" %}

{% block main %}
   {% if request.user.is_authenticated and request.user.order_set.all %}
       <div class="container">
           <h4>Замовлення</h4>
           {% if request.user.is_authenticated and
request.user.order_set.order_by_delivery_time %}
               <ul>
                   {% for active_order in
request.user.order_set.order_by_delivery_time %}
                       <li>
                           <a href="{% url "core:orders_detail" pk=active_order
%}">{{ active_order }} - {{ active_order.order_deliver_status }}</a>
                       </li>
                   {% endfor %}
               </ul>
           {% endif %}
       </div>
   {% endif %}

   <div class="container">
       {% include "blocks/shops-list.html" %}
   </div>
{% endblock %}
```

Важливим моментом при розробці додатку було зробити інтерфейс користувача додатка адаптивним, тобто, інтерфейс займає весь екран у будь-якому пристрої, для цього було використано Bootstrap 4 (рисунок 3.5.4).

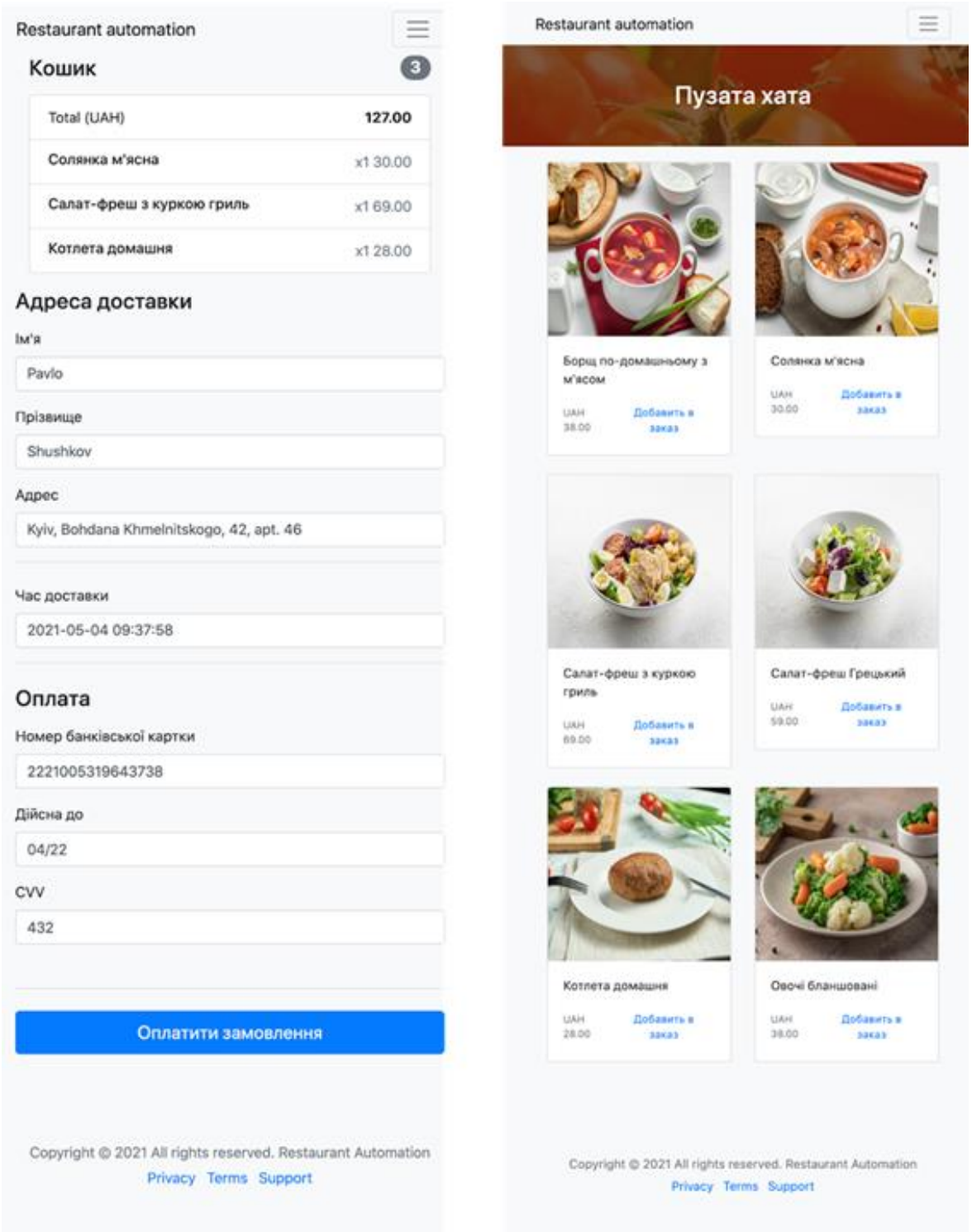


Рисунок 3.5.4 - приклад адаптивності

В результаті нами було повністю розроблено веб-додаток з можливістю оформлювати замовлення їжі онлайн, що надає можливість бізнесу збільшити канал для збуту послуг та оптимізувати витрати на технологічні розробки, і отримати більш прозору та прогнозовану модель для ведення бізнесу. Зі

сторони споживача такий додаток є зручним рішенням, оскільки економить час, надає можливість обирати місце та час для доставки та сплачувати замовлення онлайн. Також спектр послуг даного веб-додатку можна розширювати та вдосконалювати у відповідності до потреб як бізнесу та і споживачів, наприклад: надати можливість створювати та оформлювати спільні замовлення з друзями, колегами з можливістю отримувати знижки на подібні замовлення, що збільшить середній чек закладу, зменшить витрати на логістику та підніме інтерес зі сторони клієнтів.

ВИСНОВКИ

Метою даного дипломного проекту було створення програмного забезпечення для автоматизації замовлень у ресторанному бізнесі.

В першому розділі нами були проаналізовані сучасні рішення з автоматизації ресторанного бізнесу та виявлено, що особливою популярністю користуються незалежні платформи з доставки їжі. Виходячи з чого ми прийняли рішення, що розробка веб-додатку з онлайн замовлень, який спростить життя клієнта та запропонує сучасну форму ведення бізнесу для ресторанів, яка не вимагатиме коштів на індивідуальну розробку, є оптимальною для даної роботи.

Другий розділ присвячено аналізу технологій та інструментів. Як результат для розробки нашого веб-додатку ми обрали клієнт-серверну архітектуру, яку вирішили розробляти мовою Python. Рішення щодо мови програмування було прийнято на основі аналізу бізнес джерел, в яких дана мова демонструє динаміку у поширенні на ринку та популярність у своєму використанні. Також мова Python є високорівневою та інтерпретованою, з високою швидкістю розробки та легкою у читанні. В якості фреймворку ми використовуємо Django як найпопулярніший Python фреймворк для веб-розробки з найбільшою кількістю бібліотек. Для створення візуальної складової додатку нами було використано DjangoTemplate та JavaScript. При виборі бази даних ми зупинились на використанні PostgreSQL, оскільки дана база даних має найкращу підтримку зі сторони Django (Django ORM).

Третій розділ роботи присвячено безпосередньо розробці та реалізації веб-додатку. У даному розділі ми виявили та описали сутності, провели нормалізацію бази даних та описали її в Django моделях. Створили Django міграції та застосували їх. Також ми розробили основні складові додатку: нами були перевикористані авторизація Django та написана реєстрація клієнта, розроблено шаблони відображення ресторанів та позицій та розроблений компонент замовлення (кошик та екран оплати) з допомогою AJAX та

JavaScript, Django Session. Також ми подбали про те, щоб додаток буд адаптивним за допомогою Bootstrap 4.

У результаті ми повністю розробили веб-додаток та надали рекомендації щодо його майбутнього удосконалення, як для зручності клієнтів так і для покращення бізнес показників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ram, J.; Sun, S.Business benefits of online-to-offline ecommerce: A theory driven perspective. J. Innov. Econ.Manag. 2020, I77-XXVIII. Rani, N.S. E-commerce research, practices and applications. Stud. Indian Place Names 2020, 40, 773–780.
2. Why Online2O_ine Commerce is a Trillion Dollar Opportunity. [Електронний ресурс]. Ji, S.W.; Sun, X.Y.; Liu, D. Research on core competitiveness of Chinese Retail Industry based on O2O. Adv. Mater. Res. 2014, 834–836, 2017–2020. – Режим доступу: https://archive.is/zEodV.
3. The Mobile Economy 2020 [Електронний ресурс] – Режим доступу: https://archive.is/2Xhj1.
4. "Roh, M.; Park, K. Adoption of O2O food delivery services in South Korea: The moderating role of moral obligation in meal preparation. Int. J. Inf. Manag. 2019, 47, 262–273."
5. How Swiggy Works: Business model of India’s Largest Food Delivery Company. [Електронний ресурс] – Режим доступу: https://archive.is/JpNdK.
6. Online Food Delivery. [Електронний ресурс] – Режим доступу: https://archive.is/e7OK5.
7. Watch: Foodpanda’s Crave Party is Set to Be Its Biggest Food Experience Campaign. [Електронний ресурс] – Режим доступу: https://archive.is/F2uxR.
8. Alibaba’s ele.me Goes on 3 Billion Yuan Summer Spending Spree to Fight Competition. [Електронний ресурс] – Режим доступу: https://archive.is/woZLB.
9. Pigatto, G.; Machado, J.G.C.F.; Negreti, A.D.S.; Machado, L.M. Have you chosen your request? Analysis of online food delivery companies in Brazil. Br. Food J. 2017, 119, 639–657.

10. Investigation of Commission of Meituan: How Can Restaurants Become Tools for Platform Competition? [Електронний ресурс] – Режим доступу: https://archive.is/Q8AKn.

11. Chinese Ghost Kitchen Startup Secures $50 Million in Funding. [Електронний ресурс] – Режим доступу: https://archive.is/3GPYG Schnettler, B.; Rojas, J.; Grunert, K.G.; Lobos, G.; Miranda-Zapata, E.; Lapo, M.; Hueche, C. Family and food variables that influence life satisfaction of mother-father-adolescent triads in a South American country. Curr. Psychol. 2019.

12. Roh, M.; Park, K. Adoption of O2O food delivery services in South Korea: The moderating role of moral obligation in meal preparation. Int. J. Inf. Manag. 2019, 47, 262–273.

13. Liu, C.; Chen, J. Consuming takeaway food: Convenience, waste and Chinese young people's urban lifestyle. J. Consum. Cult. 2019.

14. Meah, A.; Jackson, P. Convenience as care: Culinary antinomies in practice. Environ. Plan. A 2017, 49, 2065–2081.

15. Liu, C.; Chen, J. Consuming takeaway food: Convenience, waste and Chinese young people's urban lifestyle.J. Consum. Cult. 2019.

16. Look, the Wonderful "Night Economy" of Online Food Delivery in Shanghai. [Електронний ресурс] – Режим доступу: https://archive.is/A3QhN.

17. Social Eating: When Eating Together Makes the Team More Productive. [Електронний ресурс] – Режим доступу: https://archive.is/GDnBZ.

18. Glovo [Електронний ресурс] – Режим доступу: https://glovoapp.com/en/faq/

19. Glovo the food delivery app. [Електронний ресурс] – Режим доступу: https://sifted.eu/articles/glovo-the-food-delivery-app-taking-on-deliveroo-and-winning/

20. https://uk.wikipedia.org/wiki/HTML

21. https://uk.wikipedia.org/wiki/CSS

22. https://uk.wikipedia.org/wiki/DOM

23. MDN web docs, Server-side web frameworks. Citied 21.11.2019, [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

24. Kellton Tech. Top 7 Backend Web Development Frameworks 2019. Cited 14.9.2019, [Електронний ресурс] – Режим доступу: https://www.kelltontech.com/kellton-tech-blog/top-7-backend-web-developmentframeworks-2019

25. MDN web docs, Server-side web frameworks. Citied 21.11.2019, [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

26. [Електронний ресурс] – Режим доступу: https://www.semanticscholar.org/paper/A-COMPARISON-OF-BACK-END-FRAMEWORKS-FOR-WEB-Kaluza-Kalanj/0ccd899defa3ff2a09bb0bc315828f33a7a20a8a

27. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/3.2/ref/databases/

28. "Multimodel Database with Oracle Database 12c Release 2" (PDF). Oracle. Archived (PDF) from the original on 14 April 2017. Retrieved 1 March 2017.

29. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/3.2/topics/auth/

30. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/3.2/topics/auth/customizing/#django.contrib.auth.models.AbstractBaseUser

31. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/3.2/topics/auth/customizing/#django.contrib.auth.models.CustomUser.USERNAME_FIELD

32. [Електронний ресурс] – Режим доступу: https://www.tutorialsteacher.com/python/magic-methods-in-python

33. [Електронний ресурс] – Режим доступу: https://www.codecademy.com/articles/what-is-crud

34. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/3.2/ref/class-based-views/generic-display/#listview

35. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/3.2/ref/class-based-views/generic-display/#detailview

36. [Електронний ресурс] – Режим доступу: https://pypl.github.io/PYPL.html

37. [Електронний ресурс] – Режим доступу: www.tiobe.com/tiobe-index

38. [Електронний ресурс] – Режим доступу: https://www.geekwire.com/2019/popular-programming-languages-used-worlds-largest-unicorn-startups/

39. [Електронний ресурс] – Режим доступу: https://www.semanticscholar.org/paper/A-COMPARISON-OF-BACK-END-FRAMEWORKS-FOR-WEB-Kaluza-Kalanj/0ccd899defa3ff2a09bb0bc315828f33a7a20a8a

Додаток

Презентація дипломної роботи на тему:

Розробка програмного забезпечення для автоматизації замовлень ресторанного бізнесу мовою Python

Мета роботи:

Розробка та реалізація програмного забезпечення для автоматизації замовлень ресторанного бізнесу мовою Pytho.

Предмет дослідження:

автоматизація замовлень ресторанного бізнесу

Об'єкт дослідження:

замовлення в ресторанному бізнесі

Інформаційні технології та автоматизація замовлень ресторанного бізнесу

Економічні вигоди

- за умов COVID-19 – можливість для бізнесу продовжувати свою діяльність та зберегти робочі місця;
- оптимізація витрат на простір (зали);
- гнучкість в оновленні меню, цін, промо-пропозицій;
- можливість від однієї компанії створювати декілька закладів, але при цьому готувати в межах однієї кухні для обох;
- залучення нової аудиторії без додаткових інвестицій зі сторони ресторану.

Соціальні вигоди

- безпечний формат споживання за за умов COVID-19;
- економія особистого часу який можна провести з родиною (не має потреби здійснювати покупки, готувати - що є стресом для багатьох домохазяйок або ж йти в заклад)
- можливість об'єднуватись з колегами, друзями, або ж їсти насамоті
- Дотступна інформація щодо промо-пропозицій
- Мінімізація випадків з непорозумінням в замовленні

Аналіз програмних засобів для автоматизації замовлень із ресторанів

Розробку власної системи оформлення замовлень онлайн

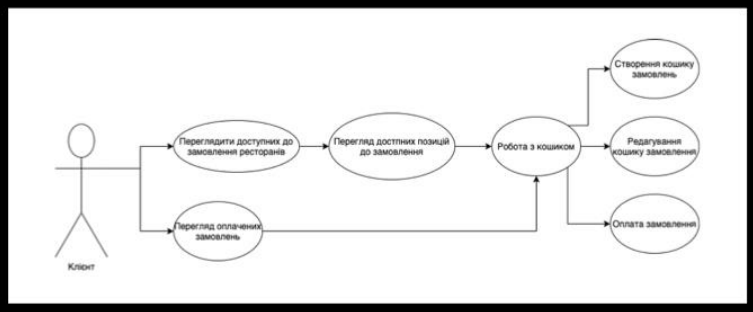
Підключити платний модуль до свого веб-сайту

Використовувати незалежну платформу з онлайн замовлень

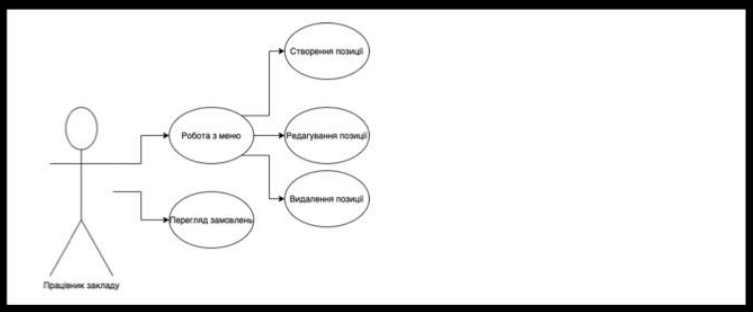
Аналіз вимог до програмного забезпечення

Веб-додаток **Restaurant Automation**

UML-діаграма можливостей клієнта в системі

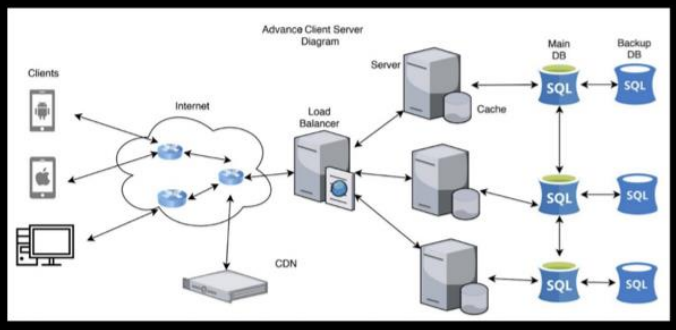


UML-діаграма можливостей працівника заклада в системі

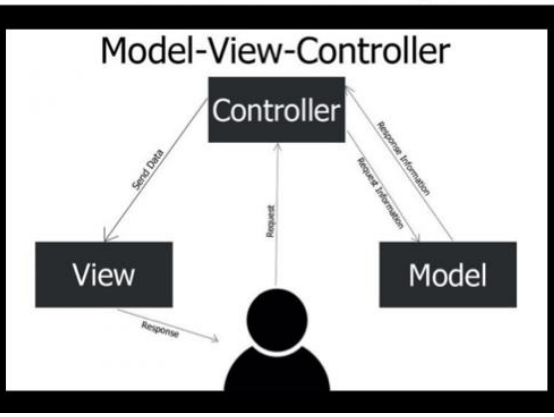


Архітектура веб додатку

Клієнт-серверна архітектура

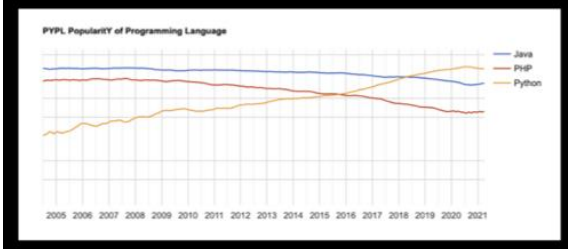


MVC



Вибір технологій реалізації веб сервісу

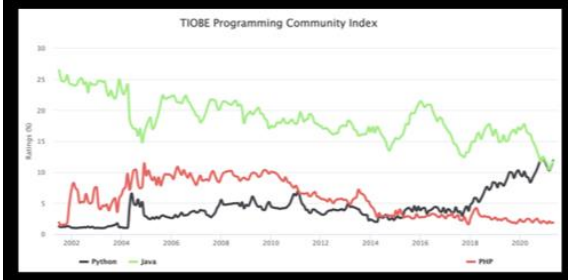
PYPL



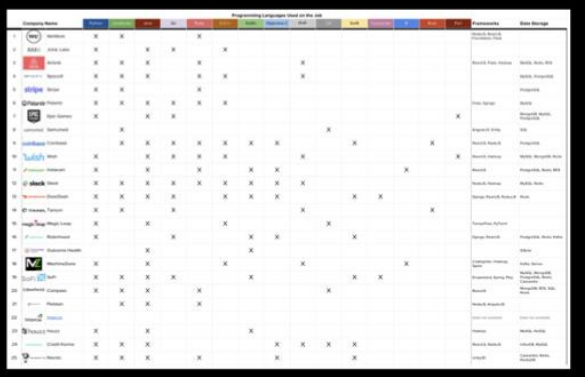
Рейтинг мов програмування dou.ua 2020

№	Мова	Частка ринку	Зміни	Основна	Додаткова	Свої проекти	Індекс вподобання
1	JavaScript	18.4	0.7	1622	4525	2892	0.59
2	Java	15.45	-2.42	1360	1193	1577	0.72
3	C#	13.76		1211	779	1411	0.83
4	Python	13.21	2.3	1163	1802	1802	0.76
5	PHP	10.88	-1	958	805	1112	0.63

Рейтинг TIOBE



Programming languages used on the job geekwire.com



Вибір фреймворку & Вибір технологій для розробки клієнтської частини & Вибір бази даних

Criterion	Laravel	Ruby On Rails	Django	Spring
1. Customized for large applications	0	0	1	1
2. High scalability	0	0	1	1
3. Adapted to beginners	1	1	1	0
4. Good for businesses applications development	0	0	1	1
5. Rapid prototype development	1	0	1	0
6. A growing Google Trend	1	0	0	0
7. Points by Haker News users	0.5	0.5	1	0.5
8. Points by Reddit users	0.5	0.5	1	0.5
9. Star Rating on GitHub	1	0.5	0.5	0.5
10. The number of Stack Overflow tags	0.5	1	0.5	0.5
TOTAL	5.5	3.5	8	5

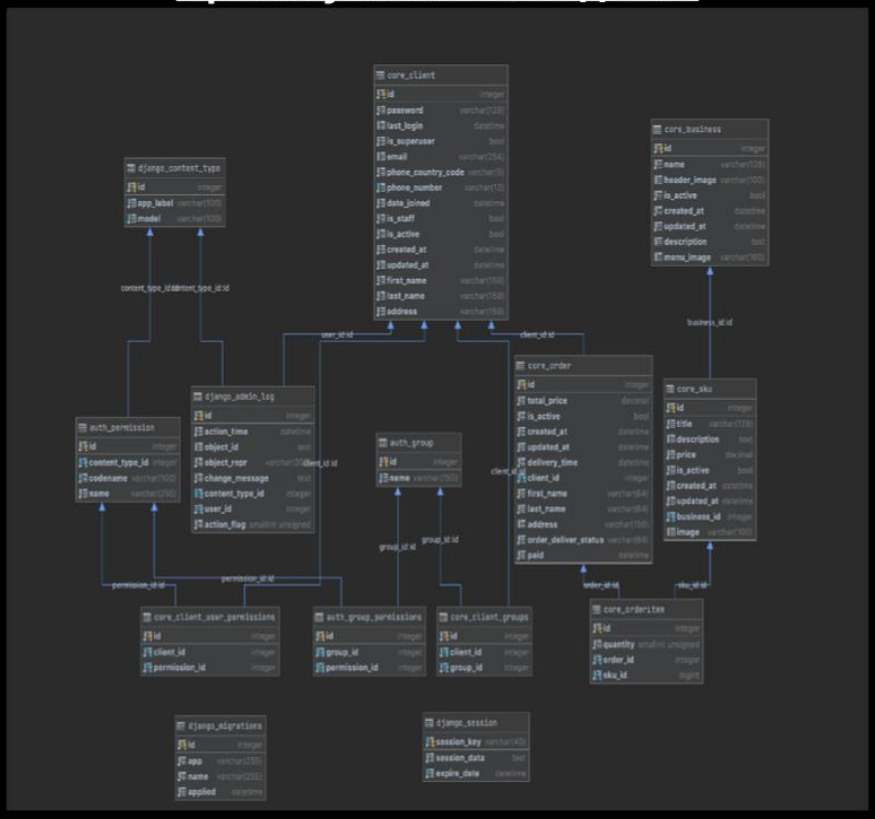
Templates django

PostgreSQL

JS

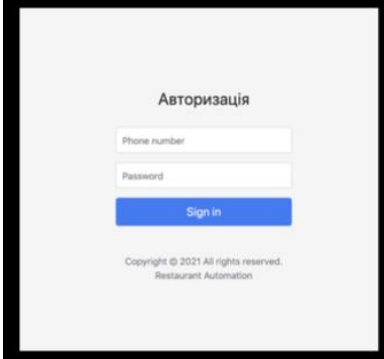
Bootstrap 4

Проектування бази даних

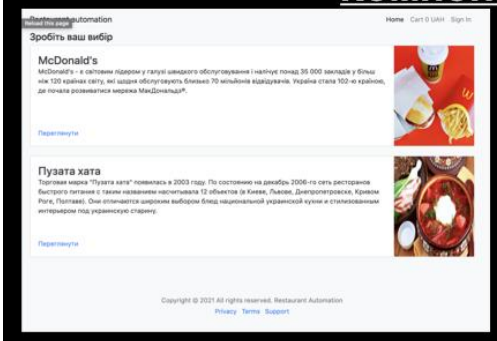


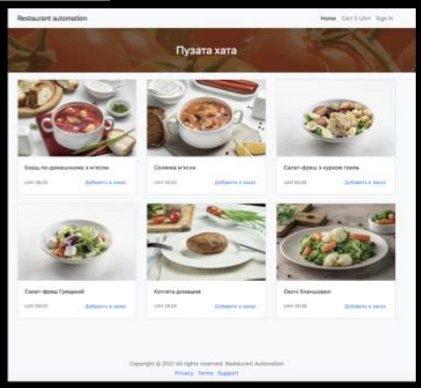
Початкові налаштування роботи системи

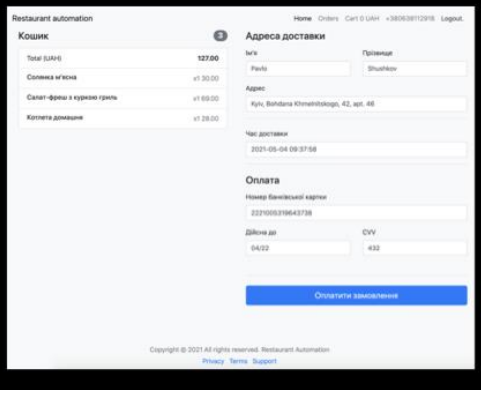
Реалізація авторизації та реєстрації

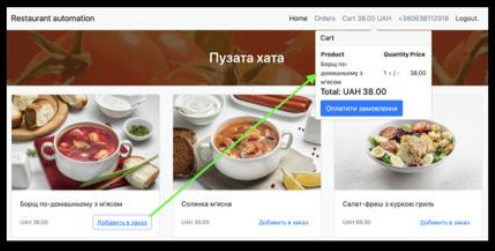


Реалізація відображення меню ресторану & Реалізація компонента замовлення

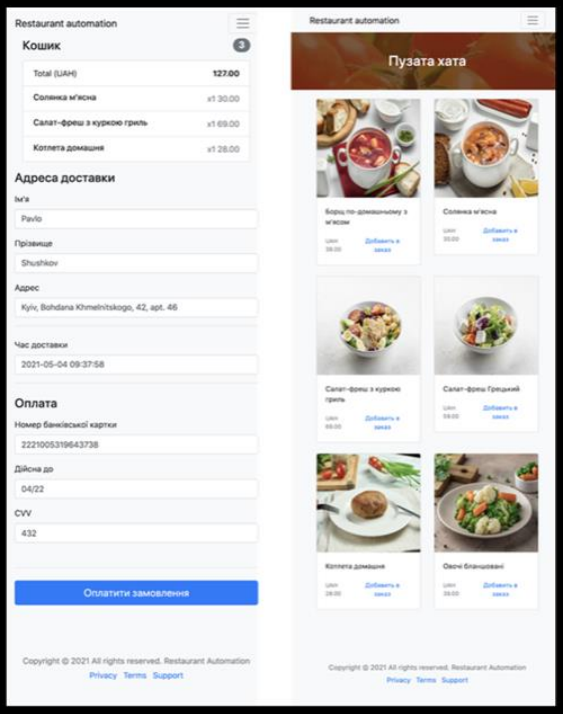








Приклади адаптивной верстки



ВИСНОВКИ