МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Сервіс відвідувань спортивних занять Bouncefit»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Проценко О.Б.**

Студента групи ІН – 73 – 9 — **Півень Є.О.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-73-9 спеціальності "Комп'ютерні науки" денної форми навчання Півня Євгенія Олеговича.

Тема: «Сервіс відвідувань спортивних занять "Bouncefit"»

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) вступ; 2) інформаційний огляд; 3) постановка завдання; 4) вибір методу рішення; 5) програмна реалізація; 6)висновок; 7) література; 8) додаток.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____Проценко О.Б.

Завдання прийняв до виконання _____ Півень Є.О.

РЕФЕРАТ

Записка: 55 ст., 24рис., 1 табл., 1 додаток, 15 джерел.

Об’єкт дослідження — веб-сервіс резервацій місць спортивних занять.

Мета роботи - розробити веб-сервіс, згідно поставленим завданням.

Методи дослідження - в процесі проведення досліджень застосовувались технології клієнт-серверу та проектування баз даних.

Результати — розроблений власний веб-сервіс, на основі шаблону проектування MVC. Інтерфейс користувача орієнтований на початковий рівень входження, дуже простий і зручний у використанні.

ВЕБ-СЕРВІС, CODEIGNITER, MVC, APACHE, POSTGRESQL, PHP, JAVASCRIPT, JQUERY, HTML, CSS, DJANGO, PYTHON

Зміст

Вступ

Більшість нашого сучасного життя неможливо уявити без великого різноманіття веб-сервісів, що оточують нас всюди. Вони використовуються багатьма компаніями для максимального полегшення взаємодії користувача з продуктом та надання більш якісних послуг. Як результат, власники бізнесу отримують більш прибуток, з можливістю гнучкого розширення, а користувачі високу якість послуг.

Але в будь-якому разі розробка веб-сервісу не є простою задачею. Для отримання максимальної функціональності послуг необхідна розробка серверної частини, розробка дизайну та мобільних додатків, при чому підключення необхідних інструментів, таких як об’єктних сховищ, баз даних чи систем оплати, потребують додаткових витрат коштів. Тому дана розробка переважно є кращим варіантом для великих компаній, що мають фінансові можливість на оплату розробки та довгострокову підтримку.

Веб-сервіс спортивних занять має на меті максимально полегшити взаємодію користувача при резервації місць на події, додатково до цього маючи в своєму розпорядженні різноманітний функціонал аккаунтів, пакетів для оплати та бонусів.

1 Інформаційний огляд

1.1 Загальний огляд веб-сервісів

Веб-сервіс – це унікальна програмна розробка, що забезпечує реалізацію бізнес задач будь-якого рівня складності. Даний вид необхідний для розширення бізнесу, надання йому відповідної гнучкості та значного розширення цільової аудиторії. Їх ключовою особливістю є використання клієнт-серверної технології.

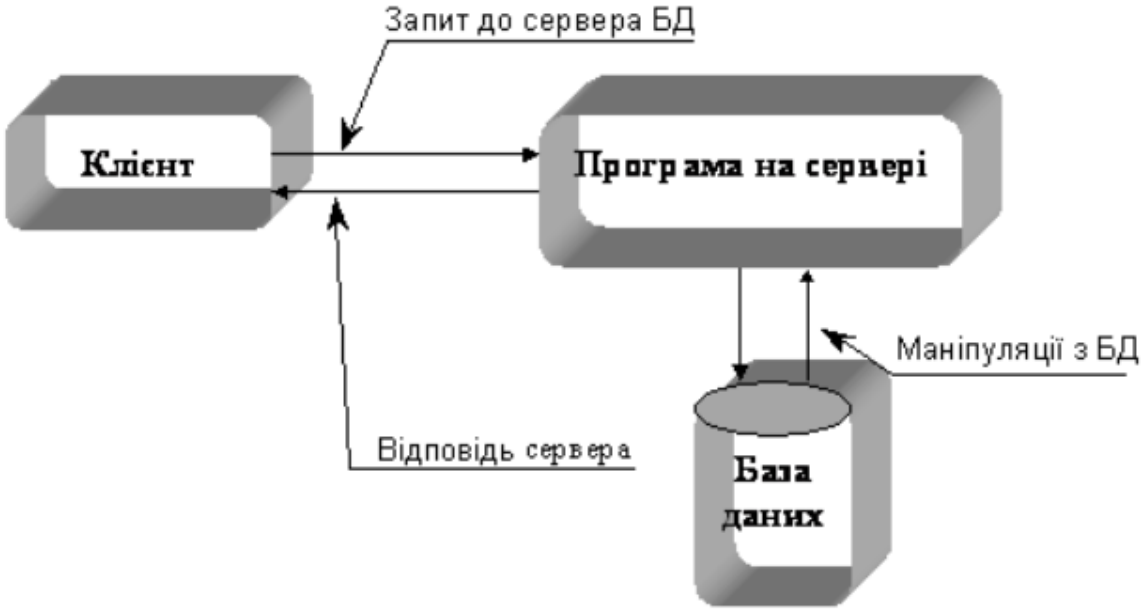

Рисунок 1.1 – Структура клієнт-серверної технології

Даний вид розробки має велику кількість переваг, що стануть незамінними для будь-якого виду бізнесу, серед них:

- високий рівень масштабованості, за допомогою чого сервіси можна додавати як окремі модулі до головного, що не потребує глобальної зміни коду в проекті;
- різноманітність вибору технологій розробки, що дозволяє використовувати на різних частинах веб-сервісу різні мови програмування та інструменти, при чому всі вони можуть знаходиться на різних вузлах мережі. Така оптимальність використання інструментів

розробки дозволить обрати найкращий варіант для своєї частини веб-сервісу;

- спрощена взаємодія дозволяє виконувати обмін даними між різними частинами веб-сервісу, що можуть бути написані на різних мовах та використовувати різні операційні системи;
- зручна інтегрованість дозволяє без особливих проблем підключати зовнішні сервіси, або будь-який інструментарій для розширення функціоналу.

Не дивлячись на великі перспективи використання даної розробки для більшості бізнес проектів, вони також мають і свої недоліки, які необхідно обов'язково враховувати, для виключення певних проблем в майбутньому. Серед таких недоліків:

- відсутність готових рішень, що значно ускладнює розробку будь-якого веб-сервісу з нуля;
- велика вартість розробки та підтримки, через що можливість розробки такого роду рішень будуть мати лише компанії з достатнім забезпеченням коштів;
- проблеми забезпечення відповідного рівня безпеки, так як враження будь-якого вузла веб-сервісу може ставити під загрозу безпеку всього проекту.

Як можемо бачити, будь-який власник своєї справи повинен детально перебачити можливі переваги та недоліки даної розробки, так як можлива переоцінка своїх можливостей може призвести до негативних наслідків, чи навіть банкрутства. Але, щоб краще оцінити реальні перспективи використання веб-сервісів, можна лише звичайним чином поглянути на частоту використання даних розробок серед бізнесу, та кількість замовлень на ринку праці. Їхня кількість незрівнянно велика, порівняно з усіма видами IT-проектів, як серед власників бізнесу, так і серед замовлень роботи. Навіть більше, наше сьогоднішнє

повсякденне життя повністю оточене різного роду веб-сервісами, які надають велику кількість різноманітних послуг та спрощують життя великій кількості людей [7].

1.2 Огляд подібних рішень

Під реалізацією веб-сервісу може бути велике різноманіття проектів, наприклад:

- дошка оголошень, що використовуються для розміщення різноманітних оголошень;
- шукачі цін та товарів, які збирають товари з різних магазинів, за допомогою чого користувач може обрати кращий товар з кращою ціною;
- системи бронювання, які допоможуть здійснити бронювання місць будь-якого роду послуг, наприклад місця відвідування тренувальних занять, або квиток на концерт;
- торгові площадки, що можуть використовуватись для продажу різноманітних товарів.

Для прикладу представимо декілька реальних проектів веб-сервісів та детально їх опишемо, для їх кращого розуміння.

Першим буде каталог для продажу власної продукції кави - Starbucks. Даний проект є досить простим та неперевантаженим в плані функціональності. В своєму складі він має веб-сайт та мобільні IOS та Android додатки.

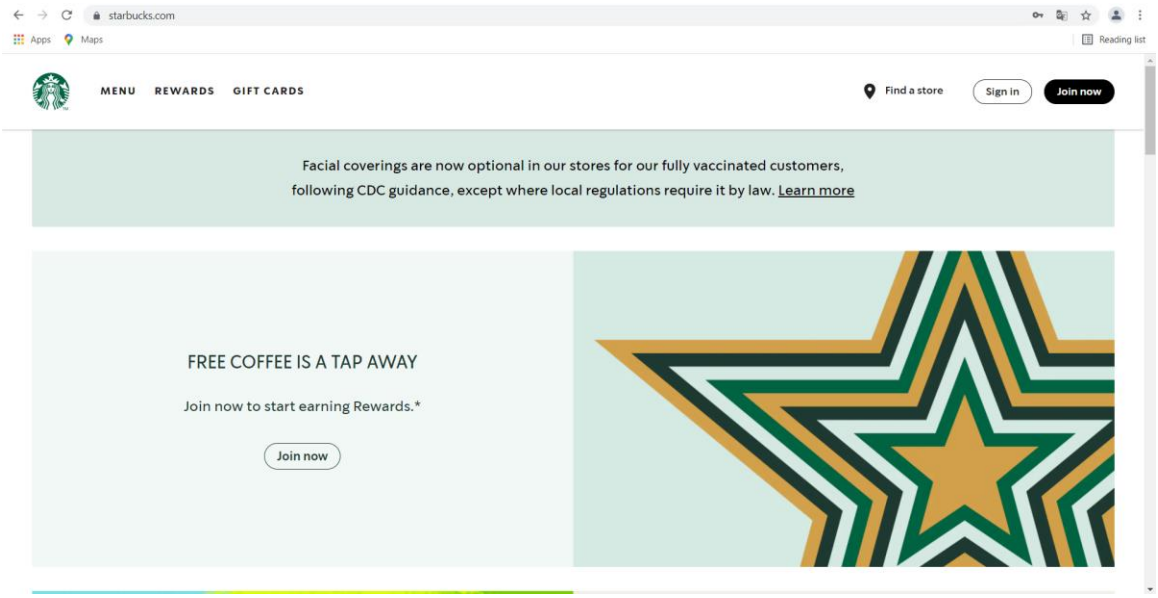

Рисунок 1.2 – Головна сторінка сервісу Starbucks

Використовуючи веб-сервіс користувач має можливість отримати детальну інформацію про наявність товару, його опис, здійснити відповідну купівлю за допомогою вбудованої системи оплати та карт визначення місцезнаходження.

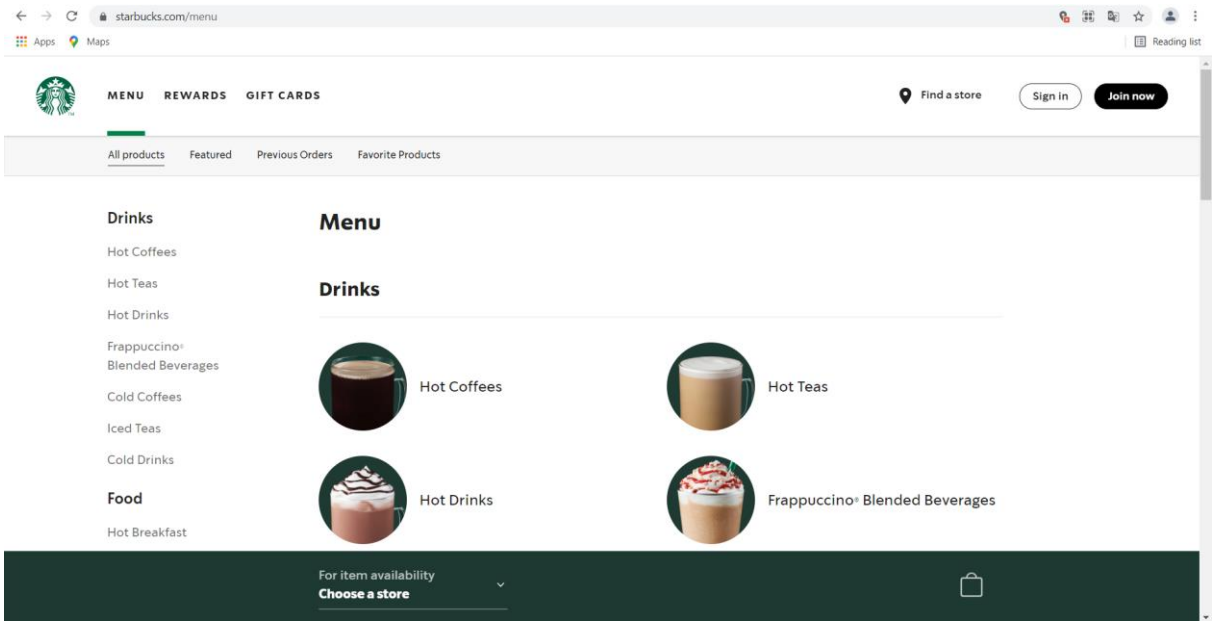

Рисунок 1.3 – Сторінка меню сервісу Starbucks

Або наприклад дізнатись останні новини компанії по асортименту продукції чи по будь-яких можливих акціях. Також в системі присутня можливість реєстрації користувача, за допомогою чого можна відслідковувати історії транзакцій та отримувати персоналізовані пропозиції, або створити подарунковий сертифікат, який можна подарувати будь-якому іншому користувачу.

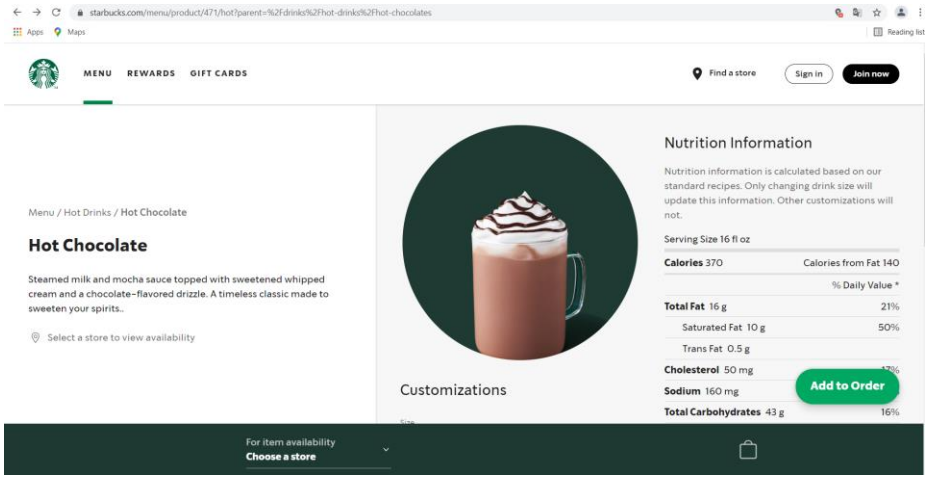

Рисунок 1.4 – Окрема сторінка продукту сервісу Starbucks

Мобільні додатки в той же час мають такі самі можливості, як і звичайний веб-сайт, але більш адаптовані під мобільні пристрої. В певній мірі навіть можна сказати, що телефонні додатки більш краще використовувати в даному випадку, аніж сторінку в браузері.

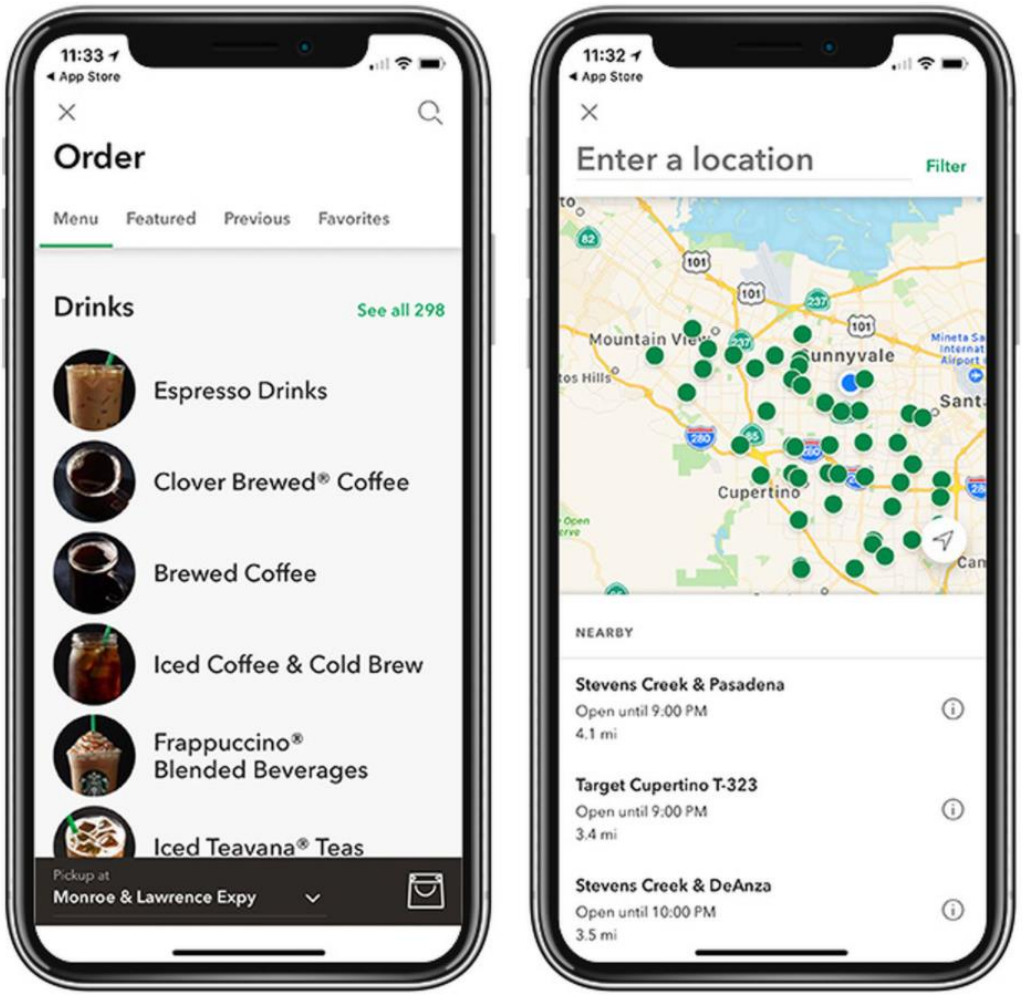

Рисунок 1.5 – Вигляд мобільного додатку сервісу Starbucks

Наступним додатком обираємо онлайн систему бронювання житла – Booking.com.

У своєму складі веб-сервіс має веб-сайт та мобільні IOS та Android додатки. Порівняно с попереднім проектом, він має набагато більше функціональності, але і вимоги до проекту також більші.

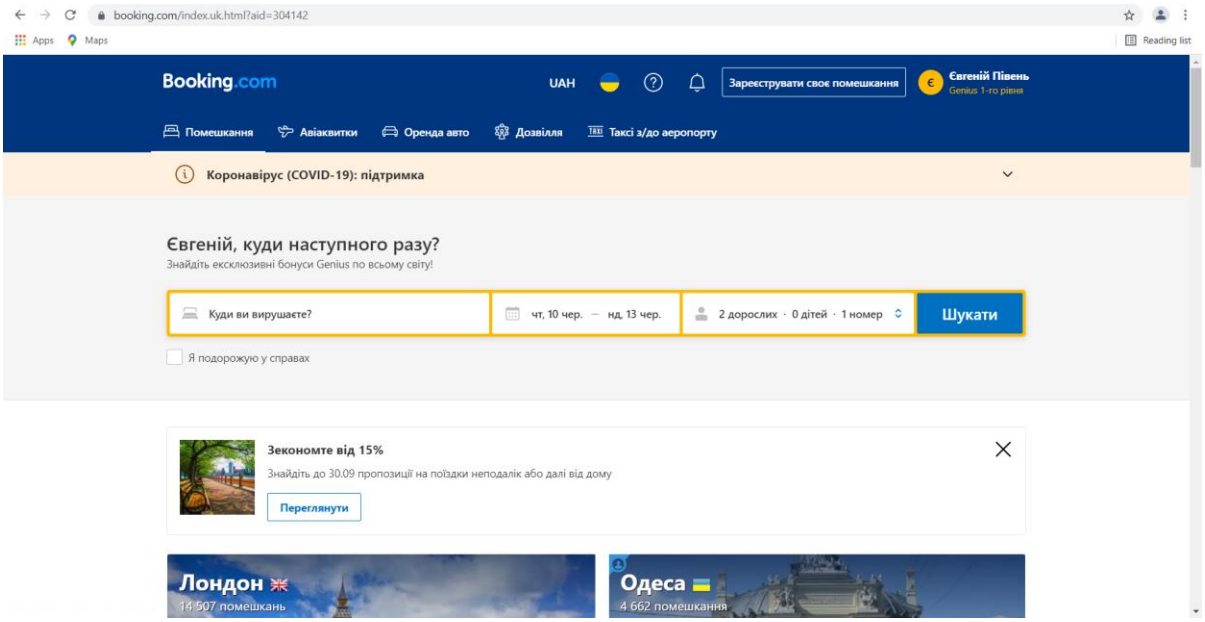

Рисунок 1.6 – Головна сторінка сервісу Booking.com

При використанні даного веб-сервісу користувач може здійснити бронювання житла в будь-якому місці України або точці світу.

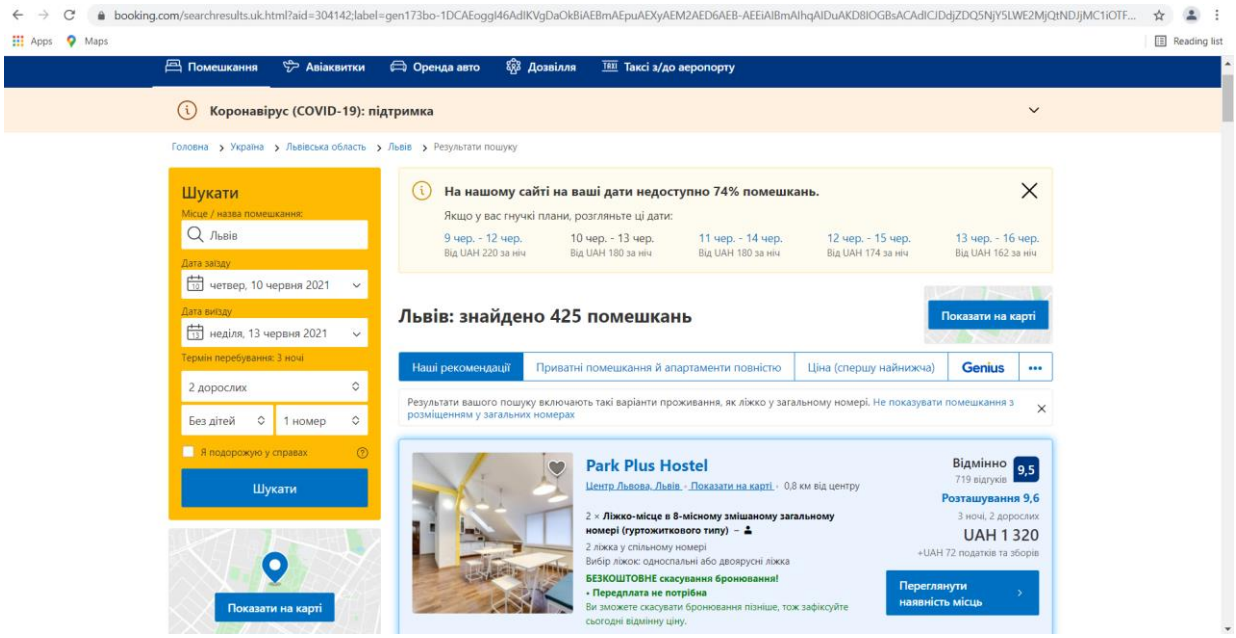

Рисунок 1.7 – Сторінка пошуку помешкання сервісу Booking.com

Для цього надається велика різноманіття фільтрів, детальної інформації та рейтинг, щоб користувач міг максимально зручно та ефективно здійснити бронювання. Додатково до цього функціоналу користувач має можливість зробити замовлення авіаквитків та оренди авто.

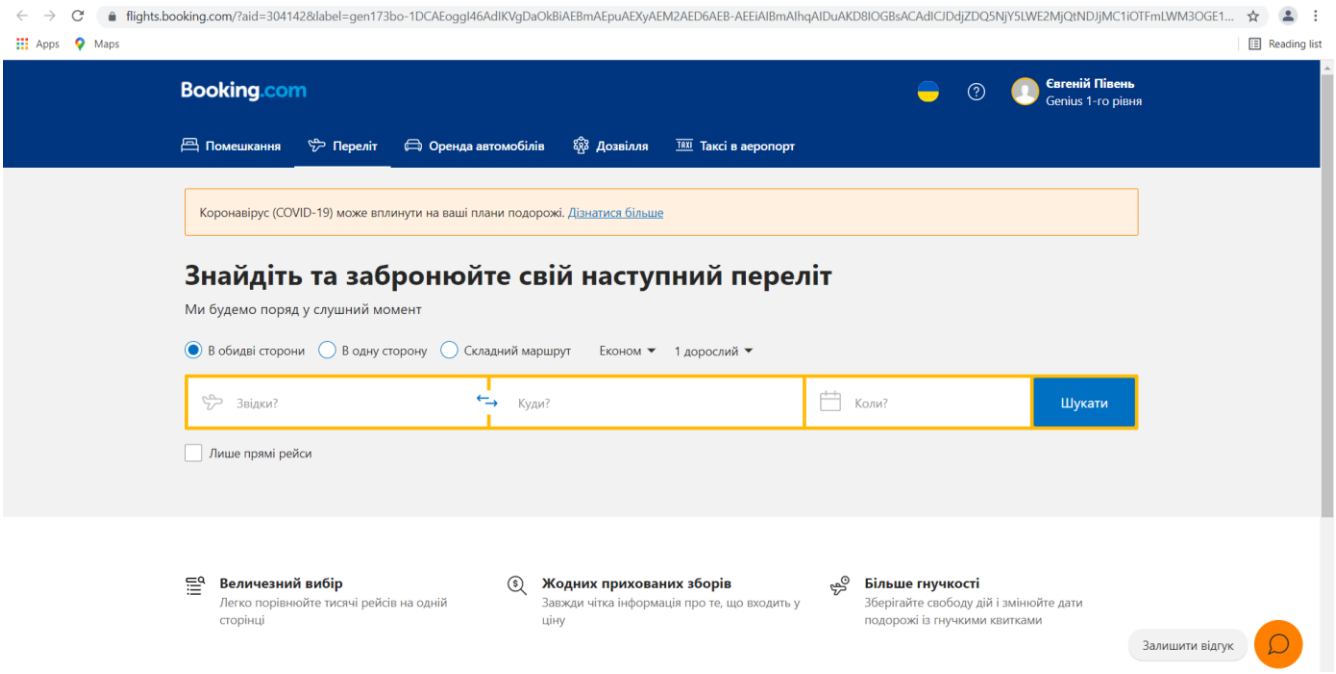

Рисунок 1.8 – Сторінка пошуку авіаквитка сервісу Booking.com

Сервіс має в своєму розпорядженні систему аккаунтів, за допомогою чого користувач може відслідковувати історію бронювання та замовлень, а також налаштувати вподобання відповідно до мови та валюти оплати.

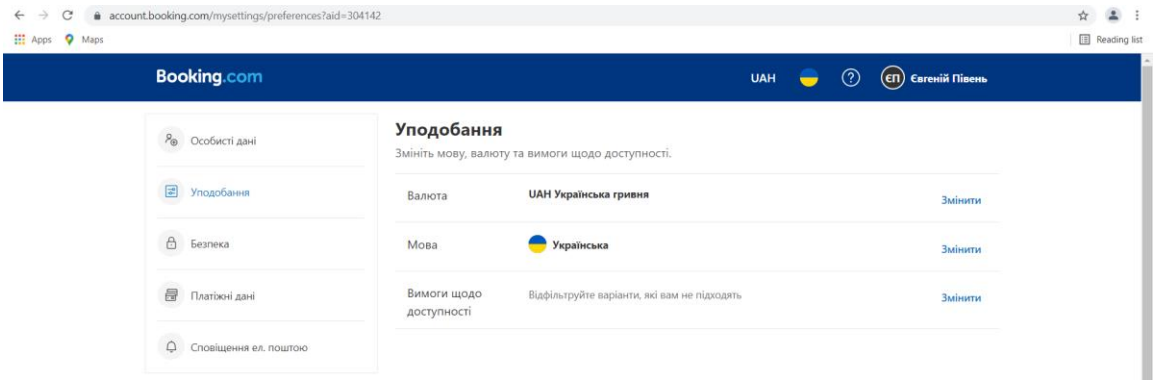

Рисунок 1.9 – Сторінка налаштувань сервісу Booking.com

Мобільний додаток та само має такий же функціонал, як і основна сторінка, тільки більш адаптовані під мобільні девайси.

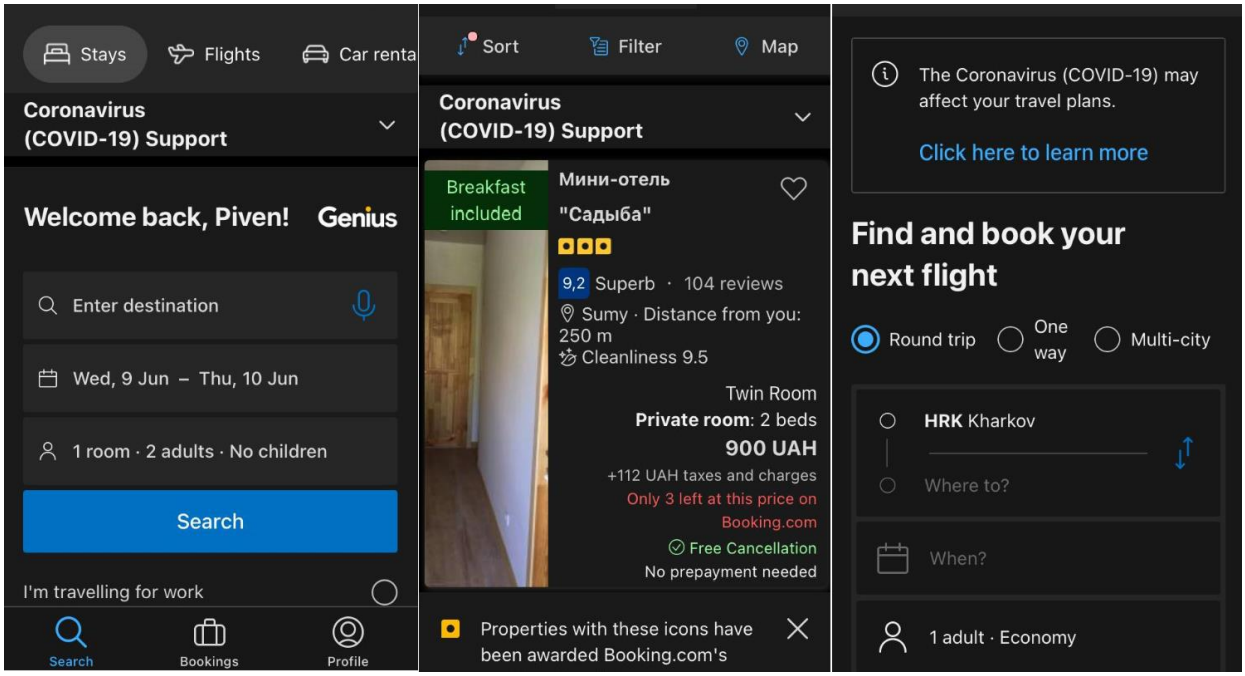

Рисунок 1.10 – Вигляд мобільного додатку сервісу Booking.com

1.3 Технології, які використовуються та мови програмування

Для розробки будь-якого веб-сервісу можуть використовувати досить різноманітні технології та мови програмування, але в першу чергу їх можна згрупувати в залежності від вузла розробки. Загалом, більшість веб-серверів використовують клієнт-серверну технології, тому спробуємо їх згрупувати за реалізацією бек частини проекту, та фронт частини. Також необхідно врахувати, що фронт частина може включати в себе окремо веб-сайт та мобільні додатки, які використовуються різні мови програмування[7].

Серверні вузли в більшості випадків використовують такі мови програмування:

- Python;
- Perl;
- C;

- Java;
- Ruby;
- PHP.

Для більш кращого розуміння даних мов програмування приведемо переваги та недоліки декількох з них.

Python – це інтерпретована мова загального призначення високого рівня. Філософія даної мови полягає у використанні значних відступів для якомога кращої читабельності коду. Конструкції мови та об’єктно-орієнтовний підхід мають в собі ціль допомогти розробникам писати чіткий і логічний код для малих та великих проектів[5].

Переваги:

- легка читабельність коду;
- невелика кількість коду;
- код виконується лінія за лінією, що спрощує захоплення помилок;
- динамічна типізація;
- підтримка великої кількості бібліотек;
- платформо-незалежна мова програмування.

Недоліки:

- низька швидкість;
- неефективне використання пам’яті;
- слабкий якість використання баз даних, порівняно з JDBC та ODBC;
- висока частота помилок виконання, через динамічну типізацію, так як змінні.

PHP – це мова сценаріїв загального призначення, що часто використовується при веб-розробці[1].

Переваги:

- висока швидкість розробки веб-додатків;
- спрощена підтримка веб-додатків;

- ефективна робота з базою даних;
- платформо-незалежна мова програмування;
- підтримка великої кількості бібліотек;
- ефективно комбінується з різними мовами програмування.

Недоліки:

- слабкий рівень безпеки, через її open-source сутність;
- низька швидкість;
- має недостатню кількість інструментів для обробки помилок;
- є слабо типізованою мовою програмуванням.

Java – це високорівнева, об’єктно-орієнтовна мова програмування, головною метою якої є якомога менша кількість залежностей при розробці, наскільки це можливо. Це необхідно для того, скомпільований код можна було писати один раз, і запускати на будь-якій платформі.

Переваги:

- простота написання коду;
- об’єктно-орієнтовна мова програмування;
- безпечність, через відсутність явних указників на області пам’яті, а також використання віртуальної машини, що відокремлює пакети локальної файлової системи, від тих, що були завантажені з мережі;
- платформо-незалежна мова програмування;
- ефективне використання пам’яті;
- багатопотоковість.

Недоліки:

- низька швидкість;
- значне використання пам’яті;
- високі вимоги до характеристик комп’ютерного обладнання;
- є слабо типізованою мовою програмуванням.

- в мові відсутні функції delete() або free() для очищення пам’яті, що робить її не контрольованою.

В більшості випадків серйозної різниці між реалізаціями сервісів на різних мовах програмування особливо немає. Іноді є особливі потреби замовників до вимог проекту, які можливо реалізувати використовуючи лише окрему мову програмування, але в більшості випадків обирають більш дешевші варіанти, або ті, які мають більшу кількість можливих спеціалістів для реалізації розробки[14].

Також не менш важливим моментом є обрання відповідного фреймворку для розробки веб-сервісу, так як вони значно покращують основні можливості мов програмування, маючи в своєму розпорядженні досить непогані обгортки з зручним інструментарієм для розробки. Але також треба враховувати, що для кожної мови програмування існують власні фреймворки. Нижче перелічимо ті фреймворки, які відносяться до описаних раніше мов:

- Python: Django, Flask, Web2py, CherryPy, Pylons, Pyramid;
- PHP: Laravel, CodeIgniter, Symfony, Zend, Yii;
- Java: Spring, Hibernate, Struts, Play, Google web Toolkit, Grails, Blade.

Також можна виокремити деякі фреймворки, що використовуються для Android та IOS розробки, серед них:

- Android: Java, Kotlin, C/C++, JavaScript, Dart, C#;
- IOS: Swift, C/C++, C#, Java.

1.4 Постановка задачі

У випускній роботі потрібно зробити та програмно реалізувати веб-сервіс резервації спортивних місць. Ресурс повинен бути схожим на ресурси, вказані раніше в пункті 1.2.

Ресурс повинен бути забезпеченим наступним функціоналом:

1. мати можливість реєстрації та авторизації користувачів;
2. наявність 2-х частин сайту: користувацька та панель адміністратора;
3. наявність списку класів;

4. фільтр класів;
5. наявність списку власних резервацій та резервацій очікування;
6. наявність сторінки всіх відео;
7. сторінка профілю с детальною інформацією користувача;
8. блок купівлі кредитів на головній сторінці;
9. на головній сторінці розмістити слайдер останніх новин та акцій;
10. можливість адміністратором додавати, редагувати та видаляти класи, а також мати можливість керування даними проекту;
11. забезпечити блок контактів, адресу та способи оплати резервацій.

2 Вибір методів рішення задачі

2.1 Вибір мов програмування

В даному сервісі використовується декілька мов програмування, це PHP та Python. В пункті 1.3 наведені їх окремі переваги та недоліки. В цьому пункті приведемо деякі їх більш детальні особливості, що можна застосувати при реалізації веб-сервісу.

Рисунок 2.1 – Основний логотип мови Python

Python. Останнім часом дана мова отримала високу популярність серед розробників. Вона підтримує об’єктно-орієнтоване та процедурне програмування, а також має в собі функцію динамічного очищення пам’яті. Порівняно з іншими мовами має дуже легкий та інтуїтивний синтаксис, що в певній мірі можна порівняти з читанням тексту на англійській мові, і відповідно високу читабельність, що дозволить пізнати основи написання коду мови за дуже невеликий проміжок часу. В той же час є абсолютно безкоштовною, та має досить простий інсталятор, що можна порівняти з будь-яким іншим на платформі Windows[7].

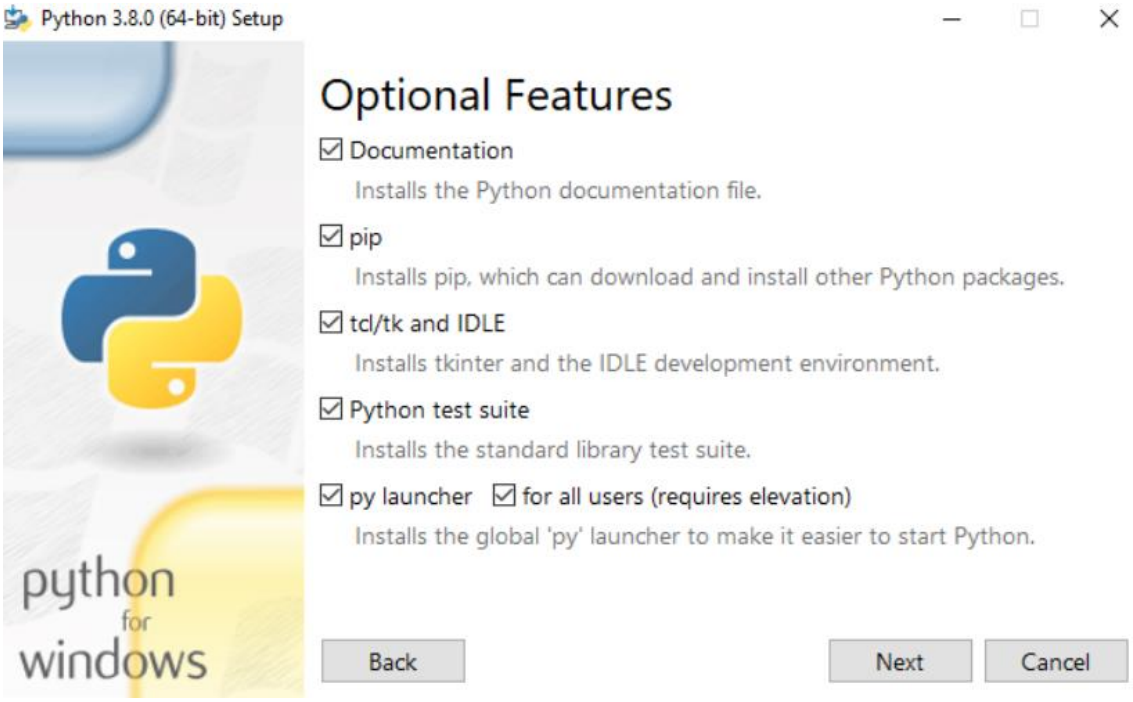

Рисунок 2.2 – Вигляд інсталятора Python

Об’єктно-орієнтовне програмування включає в себе повний комплекс концептів, включаючи класи, наслідування та інкапсуляцію. Декларація класу відбувається за допомогою використання ключового слова “class”. Порівняно з Java, запуск конструктора відбувається через функцію __init__(), замість виклику ім’я класу[6].

Окрім її платформо-незалежності, вона має також таку особливість, як можливість вбудовування, що дозволяє вставляти частини коду в інші мови, наприклад C++. Це дуже цікава можливість, так як дозволяє користувачам або розробникам гармонічно використовувати її скриптові можливість в різних джерелах будь-яких мов програмування[6].

Серед наукової спільноти використання Python є потужним інструментом, так як в своєму складі мають такі бібліотеки як Pandas, NumPy, TensorFlow та інші. Вони включають в себе велику кількість функцій, що можуть застосовуватись для обрахування багатьох математичних формул, діаграм тощо[7].

В своєму складі мова Python має велику кількість вбудованих структур даних, таких як:

- списки, що еквівалентні масивам;

- словник, який зберігає пари ключ-значення;

- кортежі, за допомогою яких можна зберігати послідовності незмінюваних об’єктів;

- стеки та черги для використання у окремих потребах відповідно до визначення даних понять.

Мови такі як C, C++ та Java повинні компілювати код перед виконанням, який внутрішньо конвертує головний код в код машинного рівня, також відомий як байт код. Але в Python код не компілюється перед виконанням. Це означає, що немає потреби виконувати ресурснозатратні дії, такі як підключення бібліотек чи пакетів для компіляції[8].

Послідовне виконання – це метод, що наслідується Python при виконанні. Саме це надає їй так званої інтерпретованості та зручне для розробки середовище. Але не дивлячи на це, даний філософія має в собі деякі недоліки, так як набагато повільніше виконує код, порівняно з Java чи C++. Хоча різноманітна підтримка функцій та бібліотек переважає, порівняно з недоліками[8].

PHP. Для кращого розуміння використання та відповідно переваг мови програмування можемо просто побачити її відсоток використання на ринку. Він складає 83% всіх веб-сервісів. Серед них присутні Spotify, Slack та BlaBlaCar та багато інших лідерів ринку. Даний факт аргументується багатьма чинниками, перелічені в наступному тексті[1].

Першими чинниками є її безкоштовність, open-source природа та велика підтримка спільноти, що дозволяє постійно оновлювати мову програмування. Як результат, відбувається постійне покращення швидкості її роботи та отримується новий функціонал.

PHP був особливо оптимізований саме для швидкого створення веб-сервісів. Він має такі вбудовані функції, як GET, POST і роботи з HTML та URLs. Для бізнесу це є ключовою перевагою, так як зменшується час на розробку IT-продукту, і відповідно зменшуються затрати на розробку.

Рисунок 2.3 – Логотип мови PHP

Великою перевагою даної мови є її значна універсальність та гнучкість. PHP код сумісний з усіма головними платформами (Linux, Unix, Windows), більшістю серверів (Xitami, Netscape, Microsoft IIS, Apache), а також більш ніж двадцяти баз даних (MySQL, PostgreSQL, MongoDB). Саме це змушує використовувати її в більшості випадків для крос-платформних додатків. Окремо увагу потрібно приділити також її можливості вбудовування. Вона успішно інтегрується з такими мовами як JavaScript, XML, WML та інші[14].

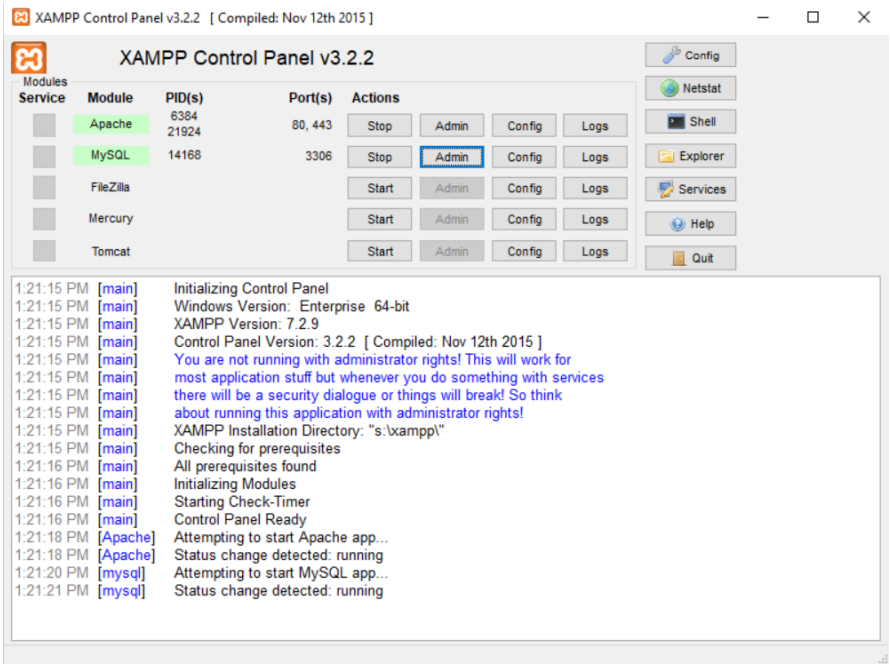

Рисунок 2.4 – Контрольна панель серверу XAMPP (Apache)

Цікавим моментом PHP є її універсальність серед веб-хостингів. Більшість з них автоматично пропонує підтримку мови PHP, навіть не потребуючи додаткової оплати для цього, включаючи також безкоштовні набори планів та ім’я домену, що дозволяє без проблем та витрат коштів створити окрему сторінку[4].

Швидкість завантаження сторінка для будь-якого сервісу чи веб-сайту є ключовою особливістю, так як напряму впливає на потенціальну кількість аудиторії. PHP надає такі можливість за допомогою функцій швидкої обробки даних, значному потенціалу налаштування та інтеграції з різноманітними спеціальними системи управління. Початково вона була створена для генерування динамічних веб-сторінок. Це і є головна причина, чому в цій мові вирішення такої задачі відбувається набагато швидке. PHP легко вбудовується в HTML, тому розробники без особливих проблем конвертувати існуючий статичний код сторінки в динамічний, додавши до нього PHP код. Власне, відповідно до цього це є найкраща мова для створення сторінок на базі HTML[1].

2.2 Вибір фреймворків та їх аналоги

Загалом сервіс складається з двох частин: серверу та веб-сайту.

Веб-сайт, на якому відбувається резервація занять, та взаємодія користувача з функціоналом написаний у поєднанні мови PHP та фреймворку CodeIgniter. Використовується архітектура MVC (модель-відображення-контролер), що надає будь-якому проекту модульність та інтуїтивність, яка дозволить ефективно організувати код та зробити його більш зрозумілим при нарощуванні функціоналу.

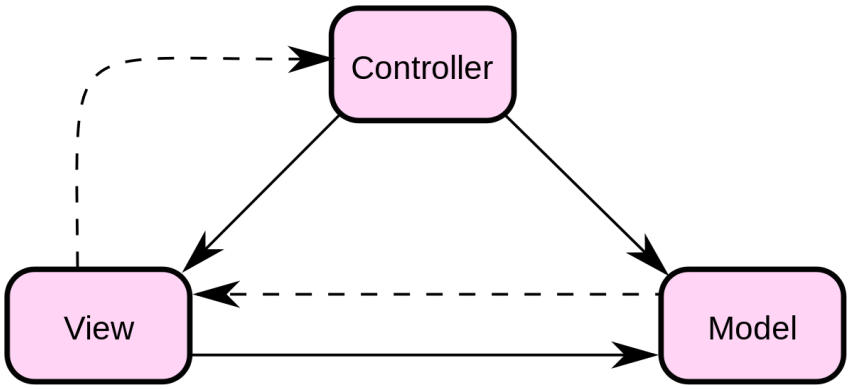

Рисунок 2.5 – Схема архітектури MVC

Для даних потреб фреймворк CodeIgniter є досить гарним рішенням, так як є легким та оптимізованим для розробки веб-сайту з помірною завантаженістю. Дизайн сайту та організація його структури організовані за допомогою мов програмування HTML і CSS. Додаткові анімації та зв’язок front частини проекту з контролерами виконані за допомогою JavaScript. В якості ресурсу бази даних використовуються дані з серверу, які отримуються за допомогою curl запитів, у вигляді json масиву.

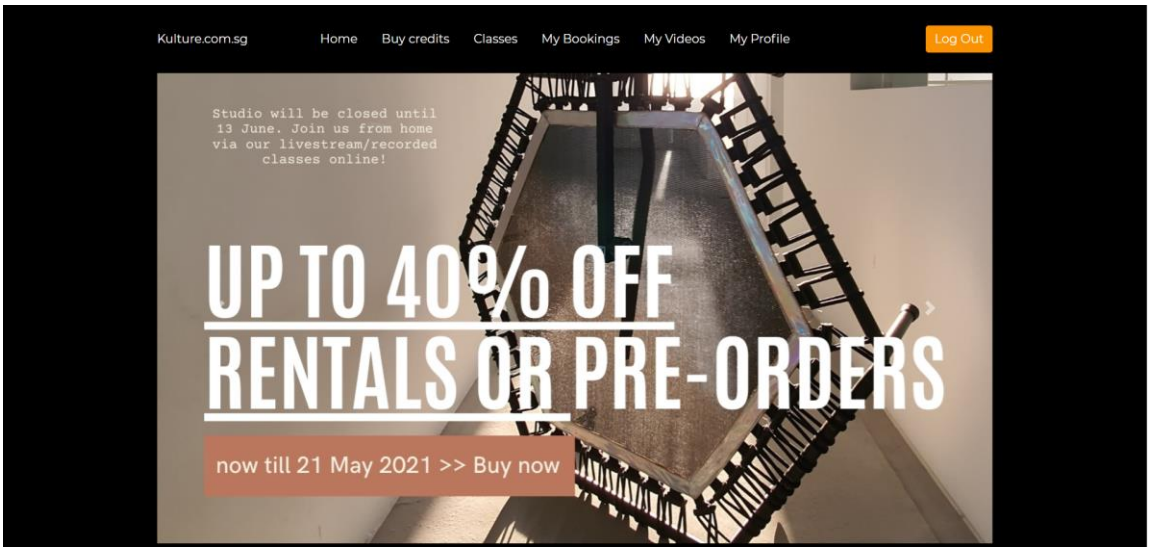

Рисунок 2.6 – Головна сторінка клієнтської частини веб-серверу

Аналогічним варіантом реалізації сайту могло б бути поєднання мови програмування PHP з фреймворком Laravel. В даній ситуації треба розуміти, що Laravel є нащадок фреймворку CodeIgniter та вважається в певній мірі більш оновленим та кращим варіантом ніж попередній[15].

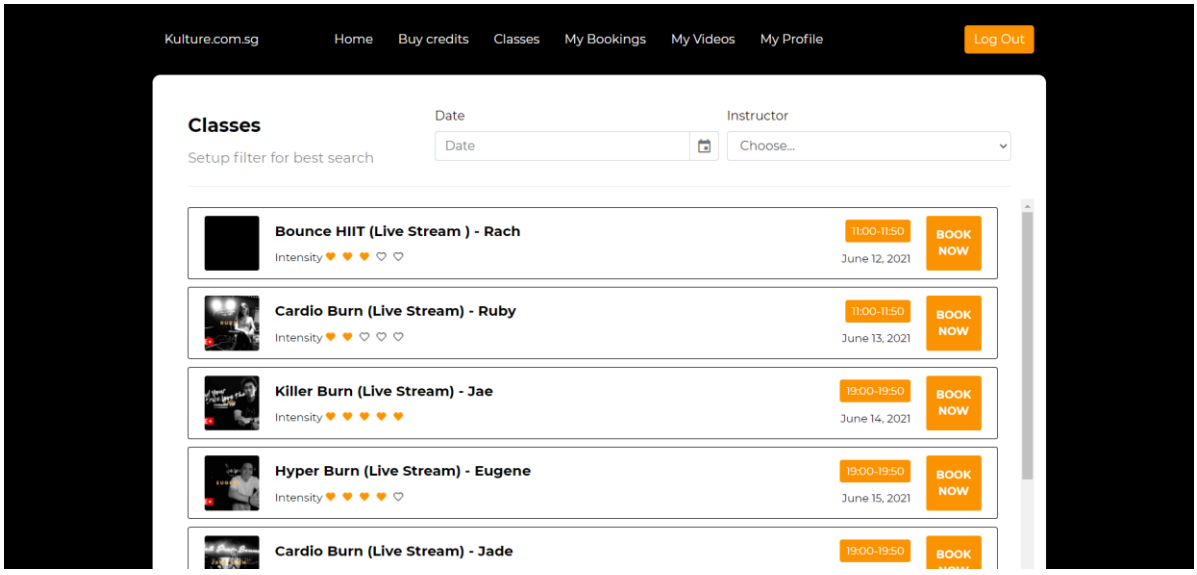

Рисунок 2.7 – Сторінка класів клієнтської частини веб-серверу

Але в даному випадку сайт не вимагає значних навантажень, або якихось специфічних вимог, які неможливо було б виконати у певному фреймворку, тому з існуючою задачею CodeIgniter справляється достатньо.

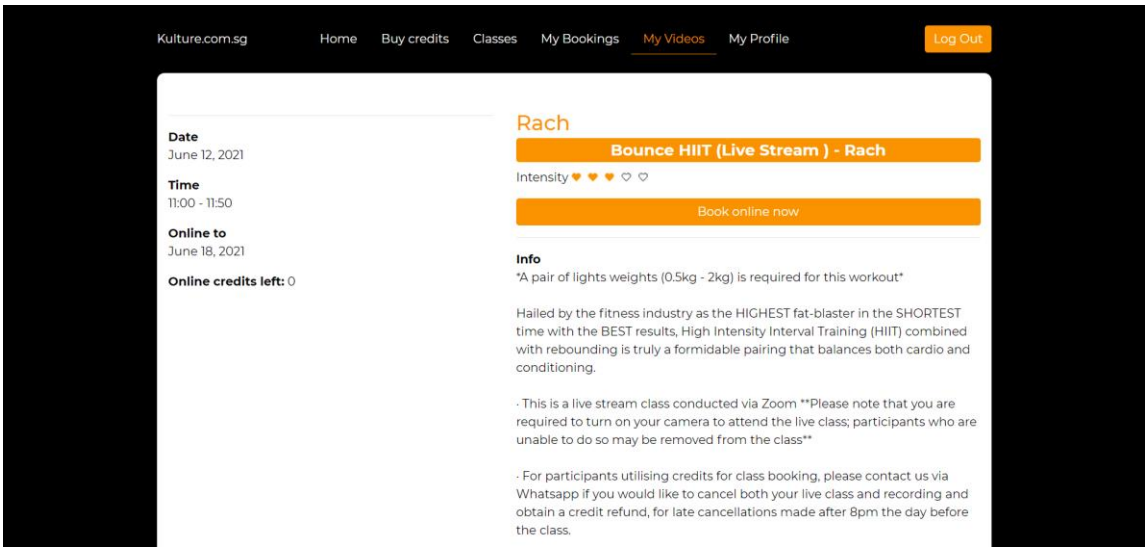

Рисунок 2.8 – Окрема сторінка класу клієнтської частини веб-серверу

Сервер написаний на мові програмування Python з використанням фреймворку Django. Організований на архітектурі MVC. Окремі шаблони для повідомлень та сторінки використовують HTML та JavaScript. Останній в більшій мірі використовується для організації динамічного оновлення даних на певних сторінках.

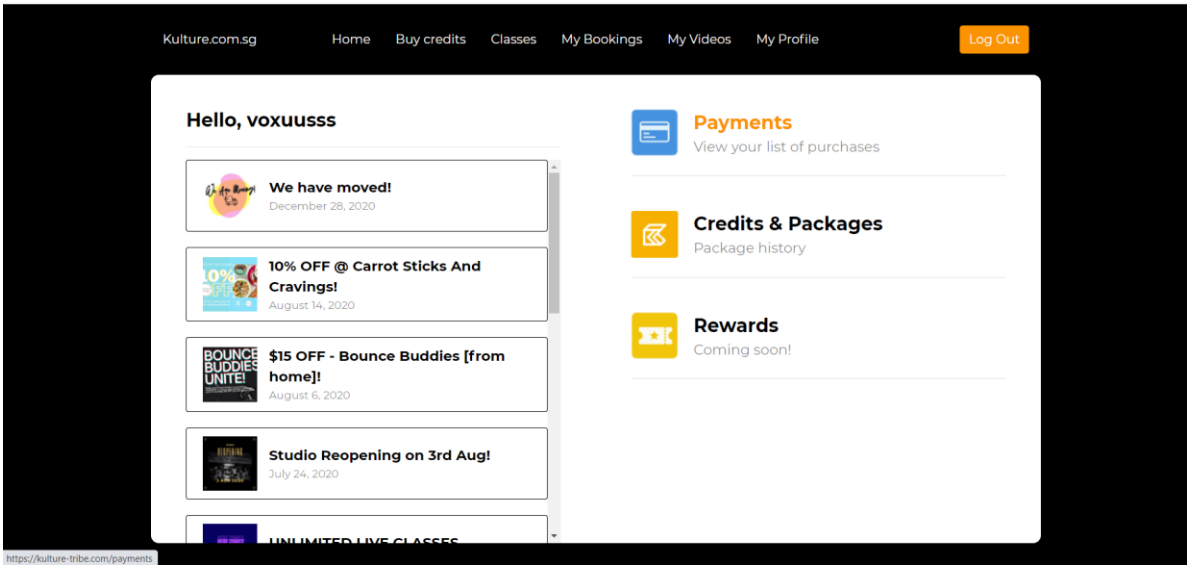

Рисунок 2.9 – Сторінка профілю клієнтської частини веб-серверу

Django є дуже гарним вибором для серверних рішень, так як має в своєму розпорядженні вбудовану адміністративну панель, яку дуже зручно налаштовувати під потреби користувача[3].

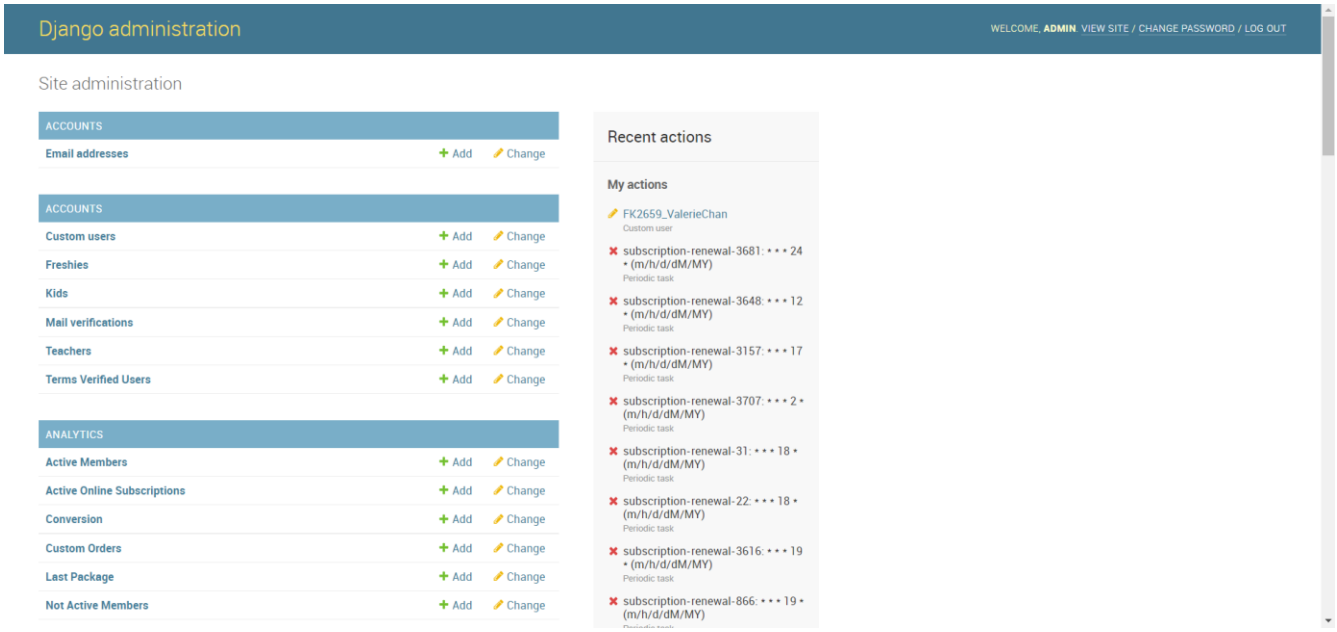

Рисунок 2.10 – Сторінка адміністративної панелі

Особливістю фреймворку Django є те, що функціонал даного проекту можна розбивати на групи. Для прикладу загальний інструментарій користувача, його моделі та API відділені в окремому файлі. Тому можна виокремити інтуїтивність інтерфейсу даного фреймворку, його швидкодію, легкість та зручність у нашаруванні функціоналу[10].

2.3 Вибір СУБД

У якості бази даних веб-сервісу використовується PostgreSQL, яка знаходиться на Amazon S3 сервері. Дане підключення можливе за допомогою даних, які прописані всередині файлу налаштувань проекту.

PostgreSQL є об’єктно-реляційною системою управління бази даних, що включає в себе комбінацію додатків, різних утиліт та бібліотек, і використовує SQL як головну мову запитів[13].

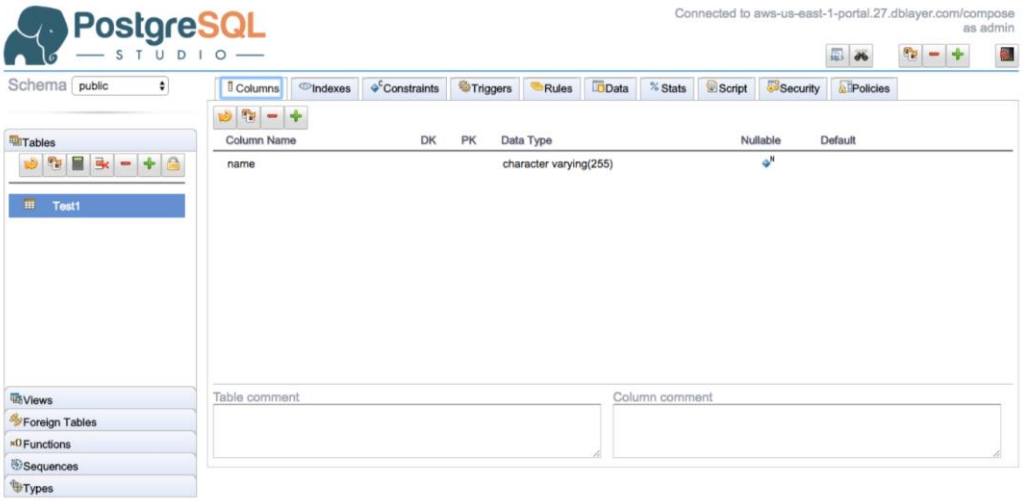

Рисунок 2.11 – Адміністративна панель PostgreSQL

Вона підтримує багато типів даних, таких як рядок, цифрові значення, дату і час я в MySQL. В особливих випадках існує підтримка типів геометричних фігур, картинок, мережевих адрес, бітових рядків тощо.

Ключові особливості бази даних:

- підтримка механізму блокування;
- безкоштовність та підтримка open-source філософії;
- має стійкість до помилок;
- не вимагає значного обслуговування;
- підтримує керування паралельним доступом (багатоверсійність);
- запускається на всіх операційних системах;
- підтримує визначені користувачем типи даних.

PostgreSQL є досить старою системою керування бази даних. Ми можемо інтегрувати її с будь-якою мовою програмування, наприклад Java, C, C++ тощо. Ця особливість дозволяє визначати особисті функції користувача. Її структурована мова має багато функцій, що можна знайти в інших базах даних[12].

Унікальні особливості PostgreSQL:

- табличний простір;
- точка відновлення;

- можливість оновити тип колонки;

Переваги:

- легка у використанні;
- має підтримку визначених користувачем типів;
- велика спільнота;
- є можливість використання збережених процедур.

Недоліки:

- для кожного окремого клієнта, що використовує базу даних створюються окремі сервіси, і результат виникає високий рівень використання пам’яті;
- у порівнянні з іншими базами, має низьку швидкість;
- не дивлячись на велику аудиторії, вона все одно поступається конкурентам;
- досить не проста інсталяція, як для початкового користувача.

3 Програмна реалізація

3.1 Інформаційна модель

Для кращого представлення мети завдання представимо її у вигляді інформаційної моделі. Модель була створена за допомогою програми Paint. На малюнку 3.1 представлена UML діаграма зв’язку користувачів: адміністратор, зареєстрований користувач, незареєстрований користувач.

Рисунок 3.1 – Діаграма UML зв’язку користувачів

Додатково до цього існує ще тріальний користувач, який не представлений в схемі, так як має ідентичні можливості до незареєстрованого користувача, лише окрім можливості відвівати заняття за запрошення зареєстрованого користувача.

Як можемо бачити зі схеми, незареєстрований користувач має можливість лише реєструватись та переглядати існуючі класи для бронювання. Купувати

кредити, бронювати місця та переглядати класи, тобто іншими словами має весь повний функціонал, що є присутнім в системі. Адміністратор в той же час має доступ до адміністративної панелі, тим самим маючи можливість змінювати дані проекту.

3.2 Розробка бази даних

В процесі створення веб-сервісу була розроблена відповідна база даних на основі вимог проекту. Створення відбувалося за допомогою обгортки міграцій фреймворку Django.

Згідно вимог, що були поставлені для необхідного функціонування системи резервацій були створена наступна таблиця 3.1:

Таблиця 3.1 – Опис таблиць бази даних

Таблиця	**Призначення**
teacher	Містить дані про вчителів занять
custom_user	Містить дані про користувачів
trial_custom_user	Зберігає інформацію про тріальних користувачів
forgot_password_keys	Проміжна таблиця, що використовується для фунціоналу забули пароль
analytics_model	Проміжна таблиця, що використовується для аналітики проекту
application	Містить дані про налаштування проекту
celery_tasks	Містить дані про celery задачі
hide_bingo	Проміжна таблиця для функціоналу гри бінго

faq	Містить дані тексту частих запитань
notification_page	Містить дані останніх оголошень
client_versions	Містить дані версії клієнту
videos	Містить дані відео занять
class_videos	Містить презентаючі відео класів
bingo_settings	Містить дані налаштувань гри бінго
online_package_banner	Містить дані банеру головної сторінки
classes	Містить дані класів занять
payment_items	Містить дані пакетів для здійснення бронювання
classes_photos	Містить дані фото класів
day_class	Містить дані класів
classes_dayclass_waiting_list	Містить дані черги очікування класів
reservation	Містить дані резервацій
user_history	Містить дані історії користувача
price	Містить дані ціни послуг
payments	Містить дані транзакцій користувачів

При проектуванні інформаційної системи бази даних була використана ER модель, яка дозволяє описати концептуальну схему за допомогою конструкцій блоків. На малюнку 3.2 розміщена ER модель, що складається з декількох найголовніших частин проекту.

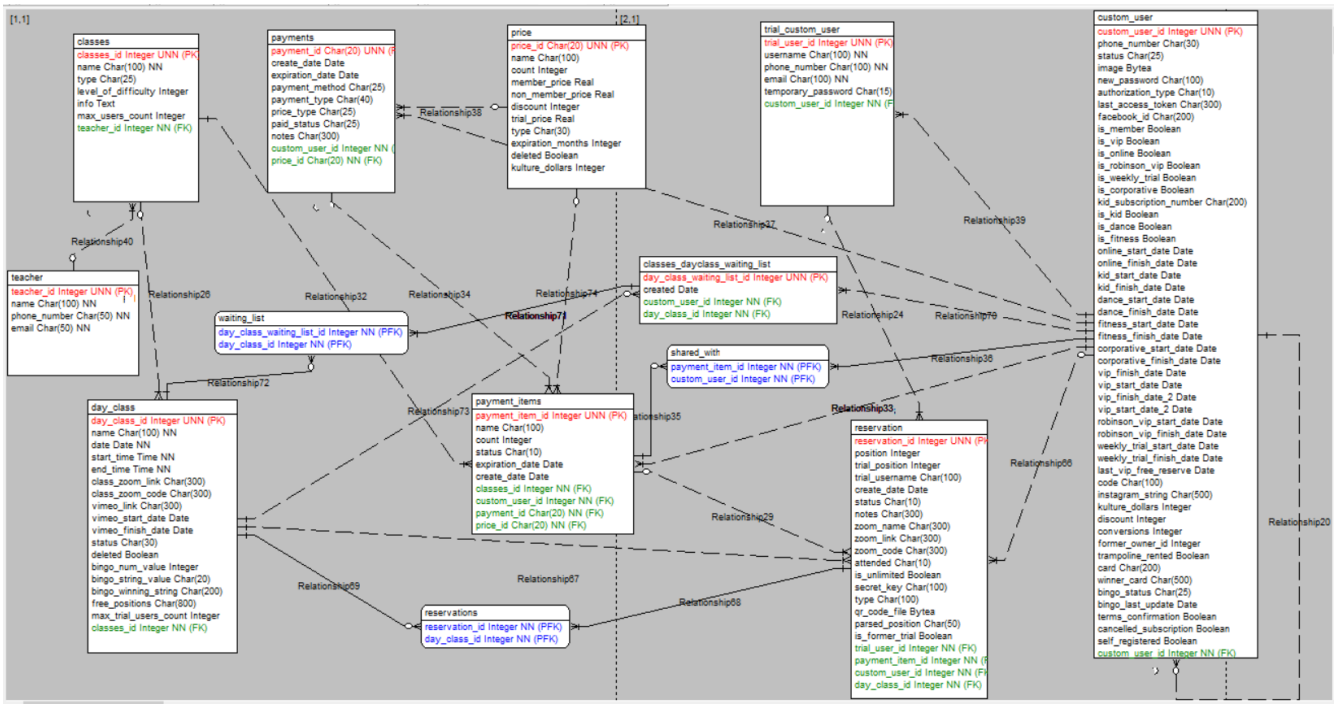

Рисунок 3.2 – ER модель функціоналу резервацій

3.3 Опис коду

Більша частина функціоналу даного сервісу реалізована на серверній частині, тому саме їй будемо приділяти найбільшу увагу. Організація файлів серверу відображена на малюнку 3.3. Наступний опис коду виконання програми буде наведений в додатку.

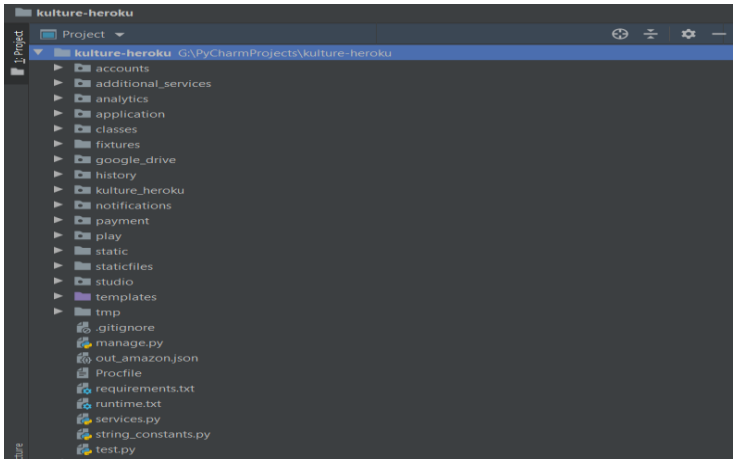

Рисунок 3.3 – Структура файлів серверу Bouncefit.

На початку в більшій мірі хотілось би приділити увагу функціоналу резервацій занять. Як говорилось раніше, першим етапом користувачу необхідно виконати купівлю пакету, або оформити підписку. Виконати це можливо на сайті додатку. За дію купівлі пакету відповідає запит create_transaction, що розташований в payment додатку проекту. Код контролеру CreateTransaction запиту наведений в додатку.

В даний запит необхідно передавати наступні параметри:

payment_method_nonce – параметр, що можливо отримати лише на front частині сервісу, після виконання успішної транзакції за допомогою braintree бібліотеки, яка використовується для організації платіжної системи;

price_id – id пакету, що відповідає типу пакету, який купив користувач;

with_discount – параметр, що передається як визначення того, чи необхідно використовувати системи бонусів користувача, або ні.

Виконання даного коду відбувається наступним чином:

Збереження параметру data, в який зберігається масив даних, які передаються в запит;

Збереження параметру user, в якому зберігається модель користувача CustomUser за допомогою функції getCurrentCustomUser, яка дозволяє отримати об’єкт користувача через токен авторизації, що передається в запиті. Код даної функції:

```
def getCurrentCustomUser(request):
    token = request.META.get('HTTP_AUTHORIZATION').replace('Token ', '')
    current_user = getUserWithToken(token)
    return current_user
```

Надалі можемо обмежитись загальним описом. На даному етапі відбувається збереження об’єкту пакету, що обрав користувач, його типу та ціни, яка закладена в даний пакет.

Код визначення типу пакету:

```
def get_price_type(self, user):
    is_trial = Payments.objects.filter(price=self, user=user).count() == 0
    if is_trial and self.trial_price != None:
        return self.PRICE_TYPE_TRIAL
    else:
        return self.PRICE_TYPE_MEMBER if user.is_member else
self.PRICE_TYPE_NOT_MEMBER
```

Код визначення ціни пакету для користувача:

```
def get_price_value_for_user(self, user, prices_ids_used=[]):
    if self.trial_price is None:
        return self.member_price if user.is_member else self.non_member_price
    else:
        if prices_ids_used:
            is_trial = self.id in prices_ids_used
        else:
            is_trial = not Payments.objects.filter(price=self, user=user).exists()
        if is_trial:
            return self.trial_price
        else:
            return self.member_price if user.is_member else self.non_member_price
```

Також відбувається перевірка наявності параметру with_discount. Якщо він присутній, частину ціни користувач оплачує своїми бонусами (якщо вони є в наявності). Якщо даний код був успішно виконаний, виконується функція транзакції braintree.Transaction.sale(), що повертає в результаті виконання масив, в яку передаються наступні параметри:

- amount – ціна купівлі пакету;

- payment_method_nonce – параметр, що був отриманий на front частині;

- масив options з параметром submit_for_settlement: True, що означає неявне виконання транзакції, тобто без його подальшого підтвердження вручну.

Далі перевіряємо успішність транзакції, через перевірку параметрів, що були збережені в масиві результату виконання функції braintree.Transaction.sale(). Якщо умова виконується, код виконується звичним чином, якщо ні, з даного масиву збирається масив, в якому зберігаються помилки транзакції та повертаються у вигляді Response.

При успішності виконання умови, здійснюється додаткова перевірка наявності отриманого id транзакції в списку транзакцій braintree. Якщо умова не виконується, повертається Response у вигляді тексту “non-success”, якщо виконується, відбувається додавання необхідних даних в базу даних, а саме:

Запис успішності транзакції в історію користувача за допомогою створення моделі UsersHistory з прив’язуванням відповідного користувача, додавання пакету до користувача у вигляді прив’язування об’єкту PaymentItems з необхідною кількістю кредитів і часом валідності, та створення об’єкту Payments, що збереже виконання даної транзакції в історії системи і дозволить прослідкувати за відповідністю всіх параметрів купівлі. В разі успішного виконання всіх умов, запит повертає Response з текстом “success”.

Наступним функціоналом є методи купівлі vip підписки buy_vip та online підписки buy_online.

Контролер BuyVip запиту buy_vip наведений в додатку.

До нього необхідно передавати наступні параметри:

payment_method_nonce – параметр, що можливо отримати лише на front частині сервісу, після виконання успішної транзакції за допомогою braintree бібліотеки, яка використовується для організації платіжної системи;

with_discount – параметр, що передається як визначення того, чи необхідно використовувати системи бонусів користувача, або ні.

Відповідний запит є певною мірою копію попереднього, тому не потребує детальної уваги. Його головними відмінностями є визначення ціни транзакції, якщо в попередньому запиті вона визначалась через спеціальну функція, відповідно до обраного пакету, то в цьому випадку вона отримується зі спеціального об’єкту Application, в якому зберігаються параметри, що відіграють роль системних налаштувань. І в тому числі відповідно в ньому зберігається ціна vip підписки. Наступною відмінністю цього запиту є дана частина коду:

```
    if user.vip_finish_date == None:
   user.vip_start_date = datetime.datetime.utcnow().date()
   user.vip_finish_date = datetime.datetime.utcnow() +
timedelta(days=application.vip_time_interval)
else:
   user.vip_start_date = datetime.datetime.utcnow().date()
   user.vip_finish_date = user.vip_finish_date +
timedelta(days=application.vip_time_interval)
user.save()
```

В попередньому запиті на цьому місці відбувалося прив’язування пакету з відповідною кількістю кредитів та часом валідності користувача, але так як в цьому випадку ми виконуємо купівлю підписки, то відповідно тут відбувається визначення часу початку валідності дії vip підписки та час її закінчення.

Також важливо звернути уваги на додаткову умову в даному коді, що відсутня в попередньому запиті купівлі пакету:

```
    if user.is_vip:
```

Вона стоїть перед виконанням транзакції, і означає необхідність наявності значення True у користувача параметру is_vip. Якщо умова не виконується, повертається Response з текстом помилки “This user is not vip yet”.

Запит buy_online є точною копією запиту buy_vip, тому не потребує окремого опису.

Наступним кроком користувач може здійснити резервацію заняття, тому значну увагу приділимо запиту резервації reserve_day_class, що знаходиться в додатку classes сервісу. Код контролера ReserveDayClass наведено в додатку.

В даний запит необхідно передавати наступні параметри:

user_id – id користувача, для якого відбувається резервація. Передається у випадку, якщо резервація виконується для повноправного користувача;

trial_user_id – id тріального користувача, для якого відбувається резервація. Передається у випадку, якщо резервація виконується для тріального користувача;

day_class_id – id класу, а іншими словами заняття, в якому користувач хоче зарезервувати місце;

position – позиція, номер якої користувач хоче зарезервувати в класі;

type – тип резервації. Може приймати або “online”, або “bounce_fit” значення.

Дані параметри необхідно передавати у типі словник в загальний масив. При необхідності можна передавати одночасно декілька таких словників в загальний масив, якщо є потреба виконати одразу декілька резервацій одночасно.

Для прикладу:

```
[{"user_id": 1,
  "position": 1,
  "day_class_id": 1,
  "type": "online"
}]
```

Виконання даного коду відбувається наступним чином:

Збереження параметру data, в який зберігається масив даних, які передаються в запит;

Збереження параметру user, в якому зберігається модель користувача CustomUser за допомогою функції getCurrentCustomUser, що згадувалась раніше;

Дані, що були збережені в параметр data та параметр user у вигляді контекстних параметрів (додаткових) передаються в серіалізатор ReserveDayClassSerializer.

Даний серіалізатор скадається з двох функцій: validation() та save(). Метод validation() використовується на самому початку для валідації даних, що передаються в запиті. В ньому включені перевірки відповідності позиції до тих, що присутні в класі, відповідності id користувача, чи id тріального користувача, відповідності id класу та наявності вільних позицій в класі.

Метод save() використовується в кінці запиту для створення нової резервації, відповідно до даних, які були передані.

Далі перевіряється умова успішності валідації параметрів, якщо умова не виконується, в Response повертається текст помилки, отриманий в серіалізаторі, в іншому випадку код продовжує виконуватись звичним чином.

Надалі обмежимось загальним описом запиту. Наступним кроком відбувається парсинг json даних, переданих в запит та перебір кожного об’єкта словника, в залежності від їх кількості в загальному масиві. При виконанні циклу відбувається присвоєння параметру класу заняття, користувача, параметру classes для подальшого присвоєння класу, що дозволить надалі визначати його тип. Після цього відбувається перевірка можливості резервації користувача в даному класі, чи не вийшов час валідності класу. Якщо дана перевірка не виконується, повертається Response у вигляді тексту “You cannot reserve yet”. Якщо помилок не виникло, тоді визначається кількість доступних кредитів в пакетах користувача для даного типу резервації та наявність будь-яких підписок за допомогою методу free_reservations_with_subscription_count.

В ньому перевіряються наявності будь-якої доступною підписки, що є в системі та їх поточна валідність.

В наступному блоці коду виконується умова, якщо сумарна кількість пакетів користувача та кількість додаткових кредитів користувача, що можуть бути отримані з методу free_reservations_with_subscription_count більше нуля, а також сумарна кількість кредитів пакетів та кількість додаткових кредитів користувача більше нуля, то код продовжує своє виконання. В іншому випадку повертаються помилки “No valid payment items”, або “Not enough payment items”. При успішному проходженні умов відбувається створення нових резервацій, а надалі, відповідно до типу резервації зняття одного кредиту з відповідного типу пакету. Також окремо присутній блок виконання резервації, при наявності trial_user_id, при якому не відбувається ніяких дій з пакетом. В разі успішного виконання всіх умов, запит повертає Response з текстом “success”.

Висновки

Під час виконання випускної роботи було розроблено та реалізовано проект, що знаходиться в експлуатації в реальному часі та отримує помірний рівень завантаженості, відповідно до кількості користувачів, що в середньому його відвідують за день. При його розробці на клієнті була використана мова програмування PHP у поєднанні з фреймворком CodeIgniter та JavaScript, в якості взаємодії фронт частини з контролерами і організації роботи сторінки.

На серверній частині була обрана мова Python у поєднанні з фреймворком Django. Згідно вказаних вимог, веб-сервіс був забезпечений відповідним функціоналом, після чого було пройдено відповідне тестування. Результатом тестування є успішна робота проекту та відповідність тим вимогам, які були вказані замовником.

Список літератури

1. David Carr, Markus Gray. Beginning PHP: Master the latest features of PHP 7 and fully embrace modern PHP development. – Birmingham: Packt Publishing, 2018. - 214 с.
2. Steve Prettyman. Learn PHP 8: Using MySQL, JavaScript, CSS3, and HTML5 – New York City: Apress, 2020. - 452 с.
3. Владимир Д. Django 2.1. Практика создания веб-сайтов на Python. – СПб: BHV, 2019. - 672 с.
4. Kevin Tatroe. Programming PHP: Creating Dynamic Web Pages. – California: O`relly Media, 2020. - 544 с.
5. Luciano Ramalho. Fluent Python: Clear, Concise, and Effective Programming. – California: O`relly Media, 2015. - 792 с.
6. Mark Lutz. Programming Python: Powerful Object-Oriented Programming. – California: O`relly Media, 2011. - 1632 с.
7. Alien B. Downey. Think Python: How to Think Like a Computer Scientist. – California: O`relly Media, 2016. - 292 с.
8. Bill Lubanovic. Introducing Python: Modern Computing in Simple Packages. – California: O`relly Media, 2019. - 630 с.
9. Rob Foster. CodeIgniter Web Application Blueprints. – Birmingham: Packt Publishing, 2015. - 330 с.
10. William S. Vincent. Django for Beginners: Build Website with Python and Django. – California: O`relly Media, 2020. - 292 с.
11. Kenneth Reitz. The Hitchhiker`s Guide to Python: Best Practices for Development 1st Edition. – California: O`relly Media, 2016. - 338 с.
12. Hans-Jurgen Schonig. Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13, 4th Edition. – Birmingham: Packt Publising, 2020. - 476 с.

13. Luca Ferrari, Enrico Pirozzi. Learn PostgreSQL: Build and manage high-performance database solutions using PostgreSQL 12 and 13. – Birmingham: Packt Publishing, 2020. - 650 c.

14. Luke Welling, Laura Thomson. PHP and MySQL Web Development (Developer`s Library) 5th Edition. – Boston: Addison-Wesley Professional, 2016. – 688 c.

15. Adam Omelak. Laminas: MVC Framework for PHP. – Birmingham: Packt Publishing, 2020. – 466 c.

Додаток

Код контролера CreateTransaction:

```
class CreateTransaction(GenericAPIView):
serializer_class = CreateTransactionSerializerDoc

def post(self, request):
    """
        Create transaction
        ---
        Request header(body):
            - name: Authorization
            - value: Token xxxxxxxxxxxxxxxxxx

        Request parameters(url):
            - name: payment_method_nonce
            - type: String
            - description: payment method nonce string

            - name: price_id
            - type: int
            - description: price id

            - name: with_discount
            - type: boolean
            - description: buying with discount or not

        Response status(int):
            - 200 - success
    """
    data = request.data
    user = getCurrentCustomUser(request)
    serializer = self.serializer_class(data=data, context={'user':user})
    if serializer.is_valid():
        price = Price.objects.get(id=data['price_id'])
        price_type = price.get_price_type(user=user)
```

```
        amount = str(price.get_price_value_for_user(user=user))
        with_discount = data['with_discount']
        if with_discount:
            if user.kulture_dollars != 0:
                amount = str(price.get_price_value_for_user(user=user) - user.kulture_dollars)
        result = braintree.Transaction.sale({
            'amount': amount,
            'payment_method_nonce': data['payment_method_nonce'],
            'options': {
                "submit_for_settlement": True
            }
        })

        if result.is_success and result.transaction:
            result, result_dict = _check_transaction_id(transaction_id=result.transaction.id)
            if result:
                if with_discount:
                    if user.kulture_dollars != 0:
                        user.kulture_dollars = 0
                        user.save()
                        UserHistory.objects.create(user=user,

type=UserHistory.USER_HISTORY_TYPE_PURCHASE_WITH_DISCOUNT,
                                                   current_kulture_dollars=user.kulture_dollars,
                                                   user_bonus=amount * -1)
                classes = price.classes_prices.all()[0]
                payment_item = PaymentItems.objects.create(user=user,
                                                           classes=classes,
                                                           count=price.count,
                                                           price=price,
                                                           expiration_date=datetime.datetime.utcnow() +
datetime.timedelta(days=price.expiration_days))
                payment = Payments.objects.create(user=user, price=price, price_type=price_type,
expiration_date=payment_item.expiration_date)
                payment_item.payment = payment
                payment_item.save()
```

```
            return Response('success', status.HTTP_200_OK)
        return Response('non-success', status.HTTP_200_OK)
    else:
        errors = []
        for x in result.errors.deep_errors:
            errors.append('Error: %s: %s' % (x.code, x.message))
        if len(errors) == 0:
            errors.append(result.message)
        return Response({'message': errors, }, status=status.HTTP_400_BAD_REQUEST)
else:
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Код контролера BuyVip:

```
class BuyVip(GenericAPIView):
serializer_class = BuyVipSerializerDoc

def post(self, request):
    """
    Buy vip
    ---
    Request header(body):
        - name: Authorization
        - value: Token xxxxxxxxxxxxxxxxxx

    Request parameters(url):

        - name: payment_method_nonce
        - type: String
        - description: payment method nonce string

        - name: with_discount
        - type: Boolean
        - description: buying with discount or not

    Response status(int):
        - 200 - success
```

```
        """
        data = request.data
        user = getCurrentCustomUser(request)
        serializer = self.serializer_class(data=data, context={'user':user})
        application = Application.objects.all()[0]
        price = application.un_trial_vip_price if 'un_trial' in data else application.vip_price
        if serializer.is_valid():
            with_discount = data['with_discount']
            if with_discount:
                if user.kulture_dollars != 0:
                    price = price - user.kulture_dollars if price > user.kulture_dollars else 0
            if user.is_vip:
                result = braintree.Transaction.sale({
                    'amount': str(price),
                    'payment_method_nonce': data['payment_method_nonce'],
                    'options': {
                        "submit_for_settlement": True
                    }
                })
                if result.is_success or result.transaction:
                    result, result_dict = _check_transaction_id(transaction_id=result.transaction.id)
                    if result:
                        if with_discount:
                            if user.kulture_dollars != 0:
                                if price > user.kulture_dollars:
                                    user.kulture_dollars = 0
                                    user.save()
                                else:
                                    user.kulture_dollars -= price
                                    user.save()
                                UserHistory.objects.create(user=user,

type=UserHistory.USER_HISTORY_TYPE_PURCHASE_WITH_DISCOUNT,
                                                           current_kulture_dollars=user.kulture_dollars)
                        if user.vip_finish_date == None:
                            user.vip_start_date = datetime.datetime.utcnow().date()
```

```
                    user.vip_finish_date = datetime.datetime.utcnow() +
timedelta(days=application.vip_time_interval)
                else:
                    user.vip_start_date = datetime.datetime.utcnow().date()
                    user.vip_finish_date = user.vip_finish_date + timedelta(days=application.vip_time_interval)
                user.save()

                Payments.objects.create(user=user, payment_type=Payments.PAYMENT_TYPE_VIP,
expiration_date=user.vip_finish_date)

            return Response('success', status.HTTP_200_OK)
        else:
            errors = []
            for x in result.errors.deep_errors:
                errors.append('Error: %s: %s' % (x.code, x.message))

            return Response({'message':errors, }, status=status.HTTP_400_BAD_REQUEST)
      else:
        return Response("This user is not vip yet", status=status.HTTP_400_BAD_REQUEST)
    else:
      return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Код контролера ReserveDayClass:

```
class ReserveDayClass(APIView):
  def get_serializer_context(self, *args, **kwargs):
    request = self.request
    return {"request": request}

  @staticmethod
  def _is_reservation_valid_for_user(user, day_class):
    is_valid_vip = user.is_valid_vip
    application = Application.objects.all()[0]
    reservation_time_interval = application.vip_reservation_time_interval if is_valid_vip else
application.membership_reservation_time_interval
```

```
    edge_date = datetime.date.today() + timedelta(days=reservation_time_interval)
    if user.is_child:
        if (day_class.classes.type != Classes.CLASS_TYPE_KIDS) or not user.parent.is_valid_kid:
            return False
    else:
        if user.is_valid_corporative:
            return True
    return day_class.date < edge_date

# @staticmethod
# def _are_free_reservations_available(user, day_class):
# return user.is_free_reservations_with_subscription(day_class)



def post(self, request, format=None):
    """
        Reserve me and my fiends
        ---

        Request header(body):
            - name: Authorization
            - value: Token xxxxxxxxxxxxxxxxxx

        Request parameters(body):

                - name: data
                - type: JSON
                - description: reserve day class bloc
                - example: [{
                             "user_id": 1,
                             "position": 1,
                             "day_class_id": 1,
                             "type": "online"
                           },
                           {
                             "user_id": 2,
```

"position": 2,
"day_class_id": 1,
"type": "online"
}]
or
[{
"trial_user_id": 1,
"trial_position": 1,
"day_class_id": 1,,
"type": "online",
"trial_username": "FirstTrialUser"
},
{
"trial_user_id": 2,
"trial_position": 2,
"day_class_id": 1,,
"type": "online",
"trial_username": "SecondTrialUser"
}]

Reserve day class bloc:

Multiple

- name: user_id
- type: int
- description: user id

or

- name: trial_user_id
- type: int
- description: trial user id

Optional

- name: position
- type: int

```
            - description: position

            - name: trial_position
            - type: int
            - description: trial position

            - name: day_class_id
            - type: int
            - description: day_class_id

            - name type
            - type String
            - description: type of reservation

        YOU CANT USE DOCUMENTATION FOR TEST REQUEST

    Response parameters(String):
        - 'success' -- success
        - 'fail' -- fail

    Response status(int):
        - 201 - success. created
        - 400 - fail. wrong parameters
        - 405 - fail. bad parameters
"""
data = request.data
serializer = ReserveDayClassSerializer(data=data,
                                       context={'user': getCurrentCustomUser(request), "request": self.request})
validation_result, validation_message = serializer.validation()
if not validation_result:
    return Response(validation_message, status=status.HTTP_405_METHOD_NOT_ALLOWED)
else:
    json_data = json.loads(data['data'])
    for item in json_data:
        day_class = DayClass.objects.get(id=item['day_class_id'])
        if day_class.classes.type in [Classes.CLASS_TYPE_BOUNCE_FIT,
```

```
Classes.CLASS_TYPE_FITNESS, Classes.CLASS_TYPE_ONLINE]:
            classes = Classes.objects.filter(type__in=[Classes.CLASS_TYPE_FITNESS,
Classes.CLASS_TYPE_BOUNCE_FIT, Classes.CLASS_TYPE_ONLINE])
        else:
            classes = Classes.objects.filter(type=day_class.classes.type)

        user = CustomUser.objects.get(id=item['user_id']) if 'user_id' in item else
getCurrentCustomUser(request)
        if ReserveDayClass._is_reservation_valid_for_user(user=user, day_class=day_class):
            # payment_items =
PaymentItems.objects.filter(Q(expiration_date__gte=datetime.datetime.utcnow())|Q(expiration_date=None),
user=user, classes__in=classes, status=PaymentItems.PAYMENT_ITEM_STATUS_ACTIVE)
            if user.is_child:
                payment_items = []
            else:
                payment_items = PaymentItems.get_shared_payment_items(user=user)
                payment_items = payment_items.filter(
                    Q(expiration_date__gte=datetime.datetime.utcnow() + datetime.timedelta(days=1)) | Q(
                        expiration_date=None) | Q(expiration_date=day_class.date))
            # additional_free_reservations = 1 if ReserveDayClass._are_free_reservations_available(user=user,
day_class=day_class) else 0
            # additional_free_reservations = 1 if user.is_free_reservations_with_subscription(day_class) else 0
            additional_free_reservations = user.free_reservations_with_subscription_count(day_class)
            length_of_payment_items = len(payment_items)
            if 'trial_user_id' in item:
                length_of_payment_items = 10000
            if user.is_valid_corporative:
                additional_free_reservations = 10000
            if length_of_payment_items + additional_free_reservations > 0:
                sum_count = 0
                if len(payment_items) > 0:
                    sum_count = payment_items.aggregate(Sum('count'))['count__sum']
                if sum_count is None:
                    sum_count = 0
                if 'trial_user_id' in item:
                    sum_count = 10000
```

```
                if sum_count + additional_free_reservations >= len(json_data):
                    reservations = serializer.save(item)
                    click_action = "reservation"

                    for reservation in reservations:
                        if additional_free_reservations == 0 and not 'trial_user_id' in item:
                            for payment_item in payment_items:
                                if payment_item.count > 0:
                                    if 'type' in item:
                                        if (item['type'] == "online" and item['type'] == payment_item.classes.type and
item['type'] in [Classes.CLASS_TYPE_BOUNCE_HYBRID, Classes.CLASS_TYPE_FITNESS,
Classes.CLASS_TYPE_BOUNCE_FIT, Classes.CLASS_TYPE_ONLINE]) or \
                                                (item['type'] == "bounce_fit" and item['type'] == payment_item.classes.type
and day_class.classes.type in
                                                 [Classes.CLASS_TYPE_BOUNCE_HYBRID,
Classes.CLASS_TYPE_KIDS, Classes.CLASS_TYPE_FITNESS, Classes.CLASS_TYPE_BOUNCE_FIT]):
                                            if payment_item.count <= 1:
                                                payment_item.count = 0
                                                payment_item.status =
PaymentItems.PAYMENT_ITEM_STATUS_DELETED
                                            else:
                                                payment_item.count = payment_item.count - 1
                                            payment_item.save()
                                            reservation.payment_item_used = payment_item
                                            reservation.save()
                                            break
                                        else:
                                            if (len(payment_items) - 1) == (*payment_items,).index(payment_item):
                                                return Response({'message': "Online class must be reserved with online
package, or live stream subscription"}, status=status.HTTP_405_METHOD_NOT_ALLOWED)
                                    else:
                                        return Response({'message': "Reservation type have to be inputted."},
status=status.HTTP_405_METHOD_NOT_ALLOWED)
                                    # else:
                                    #     if payment_item.classes.type == "online":
                                    #         if reservation.position:
```

```
                    #         return Response({'message': "Online package uses only for online
reservations"},
                    #                         status=status.HTTP_405_METHOD_NOT_ALLOWED)
                    #     if (payment_item.classes.type == "online" and payment_item.classes.type !=
day_class.classes.type) or (day_class.classes.type == "online" and payment_item.classes.type !=
day_class.classes.type):
                    #         continue
                    #     if payment_item.count <= 1:
                    #         payment_item.count = 0
                    #         payment_item.status =
PaymentItems.PAYMENT_ITEM_STATUS_DELETED
                    #     else:
                    #         payment_item.count = payment_item.count - 1
                    #     payment_item.save()
                    #     reservation.payment_item_used = payment_item
                    #     reservation.save()
                    #     break
                else:
                    if not 'trial_user_id' in item:
                        additional_free_reservations = additional_free_reservations - 1
                        reservation.is_unlimited = True
                        reservation.save()
                        user.last_vip_free_reserve = datetime.datetime.utcnow().date()
                        user.save()
                    else:
                        reservation.is_unlimited = True
                        reservation.save()
                if reservation.user != user:
                    message_body = '%s  has booked you in for %s class. Your trampoline number is %d' % (
                        user.username, day_class,
                        reservation.position if reservation.position is not None else -1) \
                        if day_class.classes.type not in [Classes.CLASS_TYPE_BOUNCE_FIT,
                                                          Classes.CLASS_TYPE_FITNESS] \
                        else '%s invited you to the %s class. Your position %d' % (
                    user.username, day_class,
                    reservation.position if reservation.position is not None else -1)
```

```
                    send_notification(message_title='reservation', message_body=message_body,
                                      user=reservation.user, data_day_classes_id=day_class.id,
                                      data_click_action=click_action, click_action=click_action)

                if day_class.reservations_count() == day_class.classes.max_users_count:
                    send_class_full_mail_notification(
                        object_url=ClassesAdmin.get_admin_url_with_url(day_class),
                        class_name=str(day_class))
                else:
                    return Response({'message': "Not enough payment items"},
status=status.HTTP_405_METHOD_NOT_ALLOWED)
            else:
                return Response({'message': "No valid payment items"},
status=status.HTTP_405_METHOD_NOT_ALLOWED)
        else:
            return Response({'message': "You can not reserve yet"},
status=status.HTTP_405_METHOD_NOT_ALLOWED)
        return Response({'message': 'success'}, status=status.HTTP_200_OK)
```