Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни “Веб-програмування та Веб-дизайн”
Тема “Розробка веб-сервісу для придбання пакетів послуг”

Студента (ки) III курсу АІ-182 групи
спеціальності 122 – «Комп’ютерні науки»
Костенко Г.П.
(прізвище та ініціали)
Керівник ст. викл. Червоненко П.П.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)

м. Одеса – 2020 рік

Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

Костекно Георгій Петрович (прізвище, ім'я, по батькові) *АІ-182* (група)

1. Тема роботи: *Розробка веб-сервісу для придбання пакетів послуг*

2. Термін здачі студентом закінченої роботи: _____

3. Початкові дані до проекту (роботи): *CloudIPSP*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): *Постановка задачі, проектування програми, вибір програмного забезпечення, створення програми, маніпулювання даними*

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

Завдання видано _____ ст. викладач Червоненко П.П.

Завдання прийнято до виконання _____ студентка Костенко Г.П.

ЗМІСТ

ЗМІСТ……3
ВСТУП……4
1. ПОСТАНОВА ЗАДАЧІ……..…5
2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ……6
2.1 ПОРІВНЯННЯ PYTHON ТА ІНШИХ МП…….6
2.2 FLASK OR DJANGO FRAMEWORK……..7
2.3 ВИБІР СИСТЕМИ КОНТРОЛЮ ДАНИХ ТА БАЗИ ДАНИХ……8
2.4 CSS FRAMEWORK BOOTSTRAP……..……...…9
2.5 ПЛАТІЖНА СИСТЕМА FONDY……10
3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ…….12
3.1 USE CASE DIAGRAM……..…..12
3.2 USER STORY……13
3.3 DATA BASE SCHEMA……..……..13
3.4 IDEA TA VIRTUAL ENVIRONMENT……14
3.5 РОЗРОБКА ПРОГРАМНОГО ПРОДКТУ……..……15
ВИСНОВОК……..……19
ПОСИЛАННЯ…….…….20
ДОДАТОК А ……..…..……21
USE CASE DIAGRAM……29
ПОСИЛАННЯ НА GITHUB……30

ВСТУП

Ще пару десятків років назад людина не могла собі уявити, що комерційні відносини між кожним з нас стануть можливі у буд'який час та у буд'якому місці.

Із розвитком інтернету та підвищенням його виркористування серед звичайних користувачів, з'явлення різноманітних інетрнет-магазині, онлайн-банкінгів і т.і. було лише питанням часу.

У 1990 році Тім Бернерс-Лі створив перший веб-сервер і браузер. Він був відкритий для комерційного використання в 1991 році. У 1994 році відбулися інші досягнення, наприклад, онлайн-банкінг та відкриття інтернет-магазину піци «Pizza Hut». У тому ж році Netscape представила SSL-шифрування даних, переданих в мережі, яке стало необхідним для безпеки інтернет-магазинів. Крім того, в 1994 році німецька компанія Intershop представила свою першу систему інтернет-магазинів. У 1995 році Amazon запустила свій інтернет-магазин, а в 1996 році з'явився eBay.

Метою даної курсової роботи стала розробка веб-сервісу за домопогою якого користувачі матимуть можливість пропонувати та купувати різноманітні пакети послуг задля заробітку чи простоого покращення свого життя.

1. ПОСТАНОВА ЗАДАЧІ

У ході даної курсової роботи був розробелний программний продукт «Tinkerer», що представляє собою веб-ресурс призначений для купівлі та продажу користувачами різни «пакетів послуг», котрі самі користувачі й можуть додавати на цей ресурс.

Для реалізації такого ресурсу було вирішено використовувати мову програмування Python(3.8) задля використання фреймворку Flask, що був розробленій для створення back-end у різноманітних веб-сервісах.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Flask — мікрофреймворк для веб-додатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Поширюється відповідно до умов ліцензії BSD.

Flask називається мікрофреймворком, оскільки він не вимагає спеціальних засобів чи бібліотек.[7] У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх бібліотек. Однак, Flask має підтримку розширень, які забезпечують додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Метою роботи є розробка веб-сервісу для розміщення, продажу та купівлі користувачами різноманітних «пакетів послуг»

Завдання роботи є:

1. Побудова алгоритму - найбільш ефективна математична модель якові можна реалізуваті у виде алгорітмічної моделі. Для цього буде Використана мова блок-схем;
2. Програмування включає в себе наступні види робіт: вибір мови програмування; уточнення способів организации даних; запису алгоритмів на вібраній мові програмування;
3. Вибір та реалізація бази даних;
4. Розробка зручного інтерфейсу для користувача;
5. Відлагодження и тестування програми.

У сучасному світі існує безліч різноманітних мов програмування, що можуть бути використані для створення back-end на веб-ресурсах. Для реалізації даного програмного продукту був обраний саме Python після порівняння можливостей різних мов програмування.

2.1 ПОРІВНЯННЯ PYTHON ТА ІНШИХ МП

C #. Microsoft дійсно молодці, вони зробили .NET Core і всіляко її просувають. Але, по-перше, це нова кроссплатформенная технологія, і там ще не все гладко. По-друге, це дійсно дорого, розробників C # мало - просто тому, що вона непопулярна.

Java. Це складно. Зробити нормальний сайт на Java - це не 10 рядків коду, як на Python. Це багато коду, це фреймворки, і потрібно знати специфіку настройки Java-серверів. Загалом, суцільні біль і страждання.

PHP. В останніх версіях він чудовий. Я навіть так скажу: PHP 7.2 не гірше Python. Але не можна просто так взяти і використовувати PHP 7.2. Якщо звичайний, не топовий розробник робить сайт на PHP, він не буде

писати тільки на 7.2: все одно доведеться читати якісь підручники, туторіали, всюди купа legacy-коду, і це не дуже добре.

JavaScript і Node.js. Це чудово і дуже сучасно, коли одна мова і на фронтенді, і на бекенд. Тільки не дуже стабільно. Node.js - хороша штука, але проблематично розгорнути її в продакшені так, щоб вона не падала і працювала нормально. Плюс, якщо ми хочемо писати якісний код на JavaScript, нам потрібен не JavaScript, а TypeScript. А ось TypeScript несподівано складний, побачивши нього у рядового розробника скипають мізки.

Таким чином було прийняте рішення використовувани мову програмування пайтом через її гнучкість, простоту та читабельність коду, що дозволить без зайвих зусиль модернізувати програмний продукт у майбутньому.

2.2 FLASK OR DJANGO FRAMEWORK

Flask і Django - обидва дорослі, які розгортаються веб фреймворка, які, в корені своєму, надають аналогічний функціонал в обробці запитів, підтримки документів, але розрізняються в масштабі відповідальності.

Велика частина відмінностей між двома фреймворками є наслідком різних підходів, інші - з відмінних основних проектних рішень. Ось невеликий список ключових відмінностей, які можуть вплинути на ваше рішення:

- Об’єкт Request - Flask використовує локальні потоки, а Django передає запит там, де це потрібно.
- Форми - Django доступний з вбудованими формами, які інтегруються з ORM і адмінкой сайту. Flask не підтримує форми за замовчуванням, але ви можете використовувати WTForms, щоб заповнити цю прогалину.
- Бази даних - Django доступний з вбудованою ORM і системою міграції, яка може керувати базами даних. Flask не може цим похвалитися, проте є інструменти, такі як SQLAlchemy, які надають аналогічний функціонал (або навіть більше).

• Аутентифікація і привілеї користувачам - Django надає додаток аутентифікації, яке надає реалізацію за замовчуванням для призначеного для користувача керування і привілеїв. Flask надає безпечні куки в якості інструменту вашої власної реалізації.

• Панель адміністратора - Django включає в себе повністю інтегрований адмін-інтерфейс для управління даними додатка. Flask не має таких функцій, але Flask-Admin - дуже популярне розширення, яке можна використовувати для створення аналогічного адміністративного інструменту.

Урахувавши усі «за та против», було прийнято рішення використовувати Flask framework для розробки прогромноо продукту «Tinkerer».

2.3 ВИБІР СИСТЕМИ КОНТРОЛЮ ДАТИНХ ТА БАЗИ ДАНИХ

Задля реалізації програмного продукту «Tinkerer» було прийняо рішення використовувати SQLite через її зручну інтеграцію через Flask SQLAlchemy у веб-сервіси, що використовують Python та Flask.

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок.

Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

2.4 CSS FRAMEWORK BOOTSTRAP

Одною з найголовніших візуаьних складових любого веб-сервісу є CSS, що дозволяє видозмінювати наповнення front-end, аби зацікавити користувача та зробити його досвід використання даного ресурсу макимально приємним та комфортних

Для створення front-end був використаний CSS framework - Bootstrap.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap — це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub. Серед інших, його використовують NASA і MSNBC.

Bootstrap має модульну структуру і складається переважно з наборів таблиць стилів LESS, які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проекту.

Основні інструменти Bootstrap:

- Сітки (grid) — наперед задані, готові до використання колонки
- Шаблони (template) — фіксовані чи адаптивні шаблони сторінок
- Типографіка (typography) — опис та визначення класів для шрифтів, таких як шрифти для коду, цитат тощо
- Мультимедіа (media) — засоби управління зображеннями та відео
- Таблиці (table) — засоби оформлення таблиць, які зокрема забезпечують сортування

- Форми (form) — класи для оформлення як форм, так і деяких подій
- Навігація (nav, navbar) — класи для оформлення вкладок, сторінок, меню і панелей навігації
- Сповіщення (alert) — класи для оформлення діалогових вікон, підказок і спливаючих вікон
- Іконочний шрифт (icon font) — набір іконок у вигляді шрифту, складається майже з 500 компонентів.

2.5 ПЛАТІЖНА СИСТЕМА FONDY

Кожний кліенто-орієнтований веб-ресурс що ставить основною задачею продаж та купівлю певних послуг має бути підєднаний до платіжної системи.

Електронні платіжні системи, або системи електронних платежів (electronic payment systems), призначені для здійснення платіжних операцій у інтернеті. За допомогою платіжної системи можна здійснювати розрахунок за товари та послуги проектів і сервісів. Наприклад, оплачувати мобільний зв'язок, комунальні послуги, кабельне або супутникове телебачення, послуги провайдерів, покупки в інтернет-магазинах.

Як платіжні засоби в електронних платіжних системах використовуються електронні (цифрові) гроші, вони є аналогом готівки і, при необхідності, зазвичай миттєво передаються з одного електронного гаманця на інший. Всі електронні платіжні системи за способом доступу до електронного рахунку можна розділити на дві групи:

- системи, що мають веб-інтерфейс для керування електронним гаманцем;
- системи, що вимагають встановлення додаткового програмного забезпечення для керування електронним гаманцем.

Збільшення використання платіжних систем неминуче, оскільки вони мають дуже важливі і незаперечні переваги, такі як:

- доступність — будь-який користувач має можливість безкоштовно відкрити власний електронний рахунок;
- простота використання — для відкриття та використання електронного рахунку не потрібно яких-небудь спеціальних знань, всі наступні дії інтуїтивно зрозумілі;
- мобільність — незалежно від місця свого знаходження користувач може здійснювати зі своїм рахунком будь-які фінансові операції;
- оперативність — переказ коштів з рахунку на рахунок відбувається протягом декількох секунд;
- безпека — передача інформації ведеться з використанням SSL протоколу з кодовим ключем 128-bit або іншими криптографічними алгоритмами.

3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

Розробка додатка складалася з чотирьох етапів:

1. Створення ідеї проекту;
2. Формування цілей і завдання проекту;
3. Розробка концептуальної схеми;
4. Розробка програмного продукту.

Перші єтапи(Створення ідеї проекту та Формування цілей и завдання проекту) нами вже були пройдені.

Перейдемо до створення концептуальних схем для нашого програмного продукту.

3.1 USE CASE DIAGRAM

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

USE CASE діаграма представлена у «додатку А».

3.2 USER STORY

User Story - формат концептуального проектування що дозволяє створити «Story» - ситуації що будуть описувати функціонал програмного продукту з точки зору користувача.

User Story:

- Як користувач «Tinkerer» я можу перегляати пропозиції різних пакетів послуг на сайті.
- Як користувач «Tinkerer» я можу додавати нові пакети послуг.
- Як користувач «Tinkerer» я можу придбати пакети послуг.
- Як користувач «Tinkerer» я можу сплатити пакети послуг через інтегровану систему оплати Fondy.

3.3 DATA BASE SCHEMA

Схема баз даних — це структура системи баз даних описана формальною мовою, яка підтримується системою керування баз даних (СКБД) і відноситься до організації даних для створення плану побудови бази даних з розподілом на таблиці. Формально схема баз даних є набором формул (правил), які називаються обмеженнями цілісності. Обмеження цілісності забезпечують сумісність між усіма частинами схеми. Всі обмеження виражаються однією мовою.

Поняття схеми бази даних відіграє ту ж роль, що і поняття теорії у численні предикатів. Модель цієї «теорії» точно відповідає базі даних, яку можна побачити в будь-який момент часу як математичний об'єкт. Таким чином, схема може містити формули, що представляють обмеження цілісності спеціально для застосунків і обмеження спеціально для типу бази даних, які виражені однією мовою баз даних.

В реляційній базі даних, схема визначає таблиці, поля, відношення, індекси, пакети, процедури, функції, черги, тригери, типи даних, послідовності, матеріалізовані уявлення, синоніми, посилання баз даних, каталоги, Java, XML-схеми та інші елементи.

Схема, як правило, зберігається в словнику даних. Хоча схема визначена в тексті мовою бази даних, цей термін часто використовується для графічного позначення структури бази даних. Іншими словами, схема — це структура бази даних яка визначає об'єкти в базі даних.

Схема бази даних представлена у «додатку А».

3.4 IDEA TA VIRTUAL ENVIRONMENT

Для створення будь-якого програмного продукту потрібні такі складові як текстовий редактор(або IDEA), мова програмування, фремворки, база даних,а також певні пакети та бібліотеки.

Для створення «Tinkerer» була використана IDEA PyCharm від помпанії JetBrains.

PyCharm — інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний зневаджувач, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена російською компанією JetBrains на основі IntelliJ IDEA.

PyCharm працює під операційними системами Windows, Mac OS X і Linux.

Можливості:

- Статичний аналіз коду, підсвічування синтаксису і помилок.
- Навігація серед проектів і сирцевого коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використаннями методів.
- Рефакторинг: перейменування, витяг методу, введення змінної, введення константи, підняття і опускання методу тощо.
- Інструменти для веб-розробки з використанням фреймворку Django та Flask
- Вбудований зневаджувач для Python
- Вбудовані інструменти для юніт-тестування
- Розробка з використанням Google App Engine

- Підтримка систем контролю версій: загальний користувацький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття

Також при розробці программного продукту було використано стандартне Virtual-Environment для мови програмування Python.

3.5 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

Завдяки використанню мови програмування Python та фреймворку Flask, а також інтеграції SQLite та платіжної системи Fondy було розроблено веб-ресурс «Tinkerer».

Программний код «Tinkerer» представлений у «додатку А», а також на GitHub за даним посиланням:

https://github.com/luckybomj/Tinkerer

Нижче будуть представлені зображення виконання функцій програмного продукту «Tinkerer»

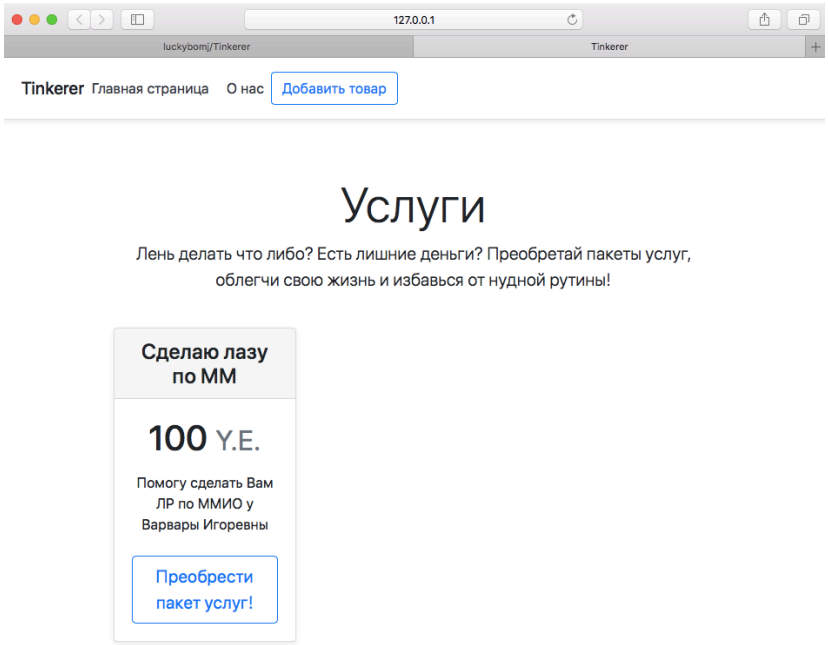

Рис.1 - Головна сторінка «Tinkerer»

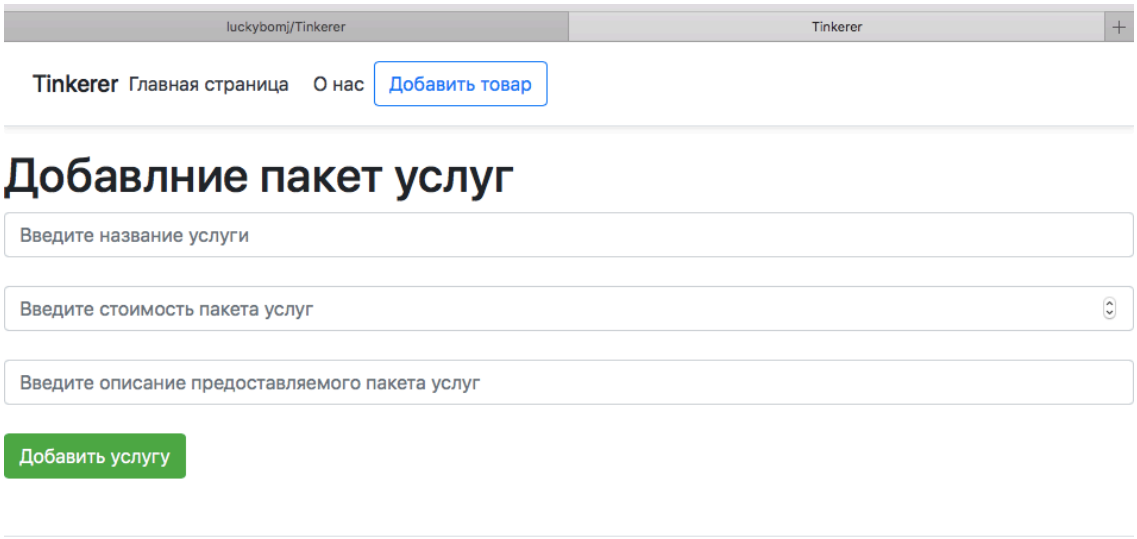

Рис.2 - Додання користувачем нового пакету послуг

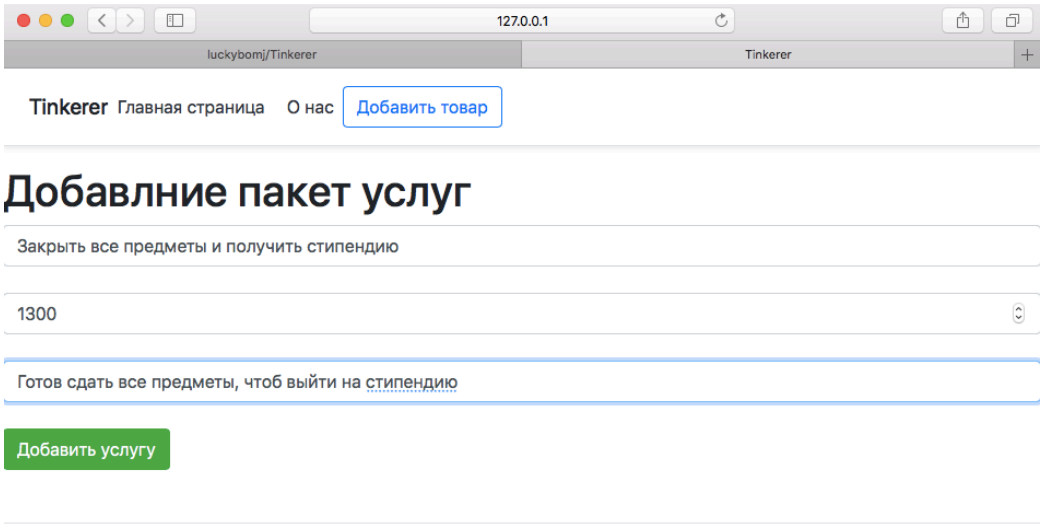

Рис.3 - Заповненння форми для додання пакету послуг

Після додання нового пакету послуг за допомогою функції redirect() виконується редірект на головну сторінку програмного продукту, та оновлюється її вид.

Дані з форми були додані у базу даних за допомогою фрункції post(), а після виконуєть виведення нових даних на головну сторінку .

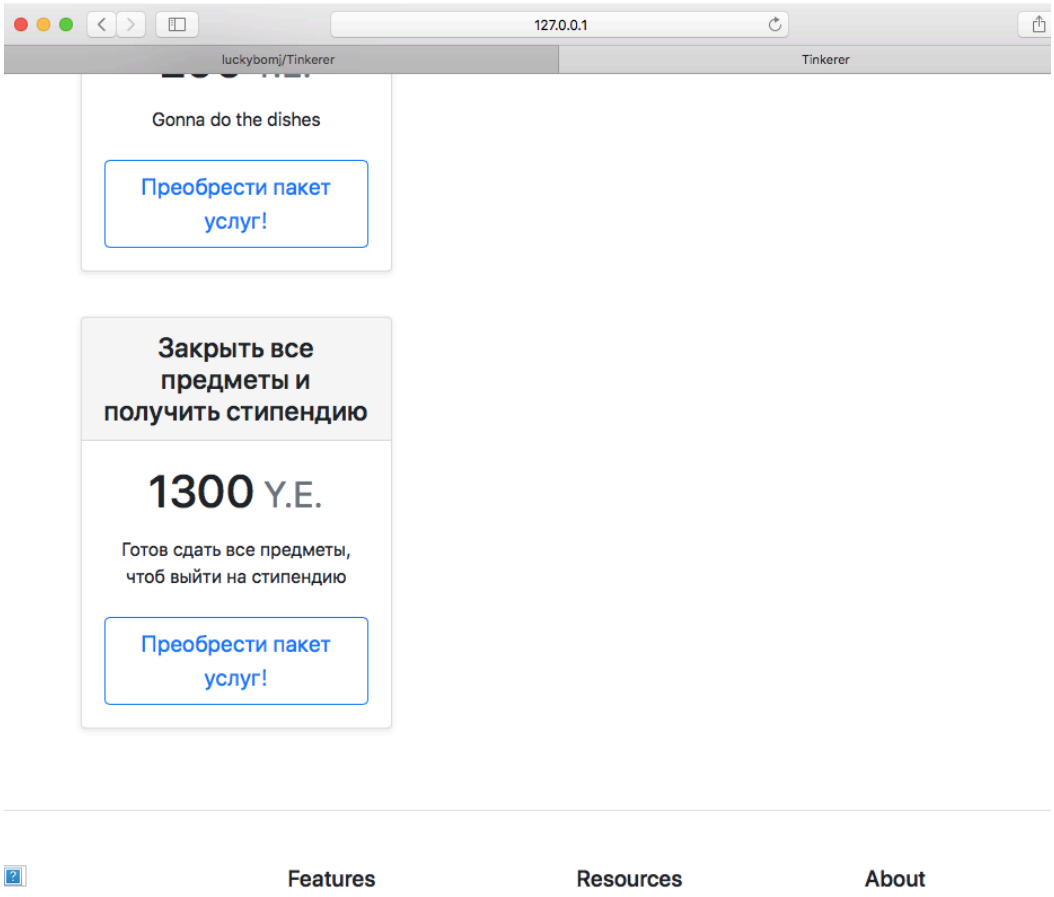

Рис.4 - Головна сторінка після додання нової послуги

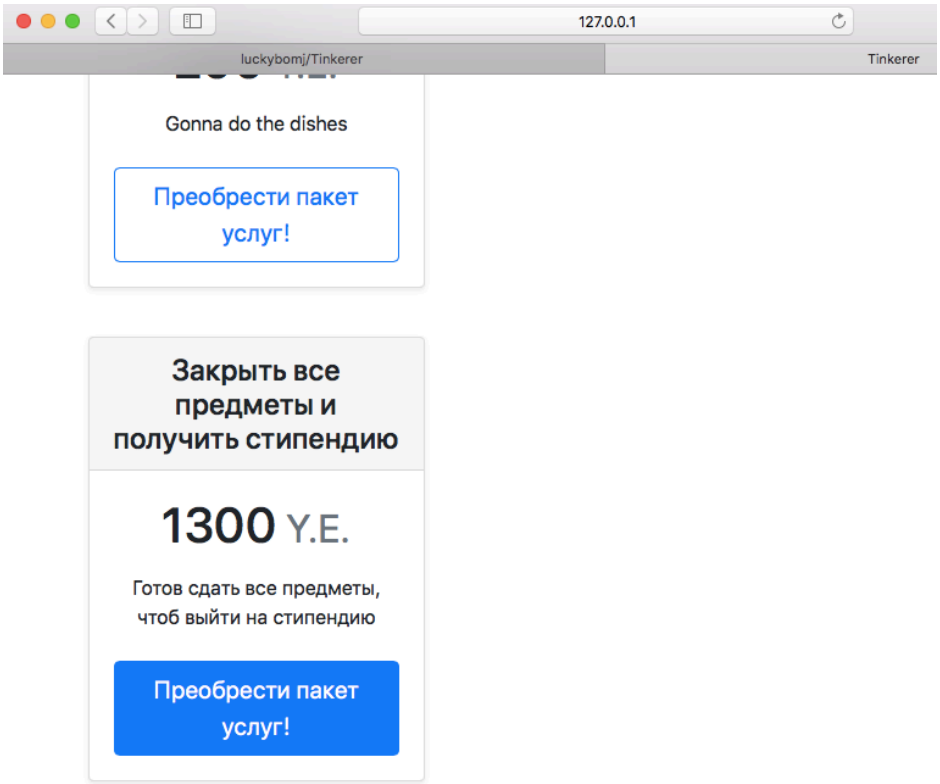

Рис.5 - Можливість оплати пакету послуг

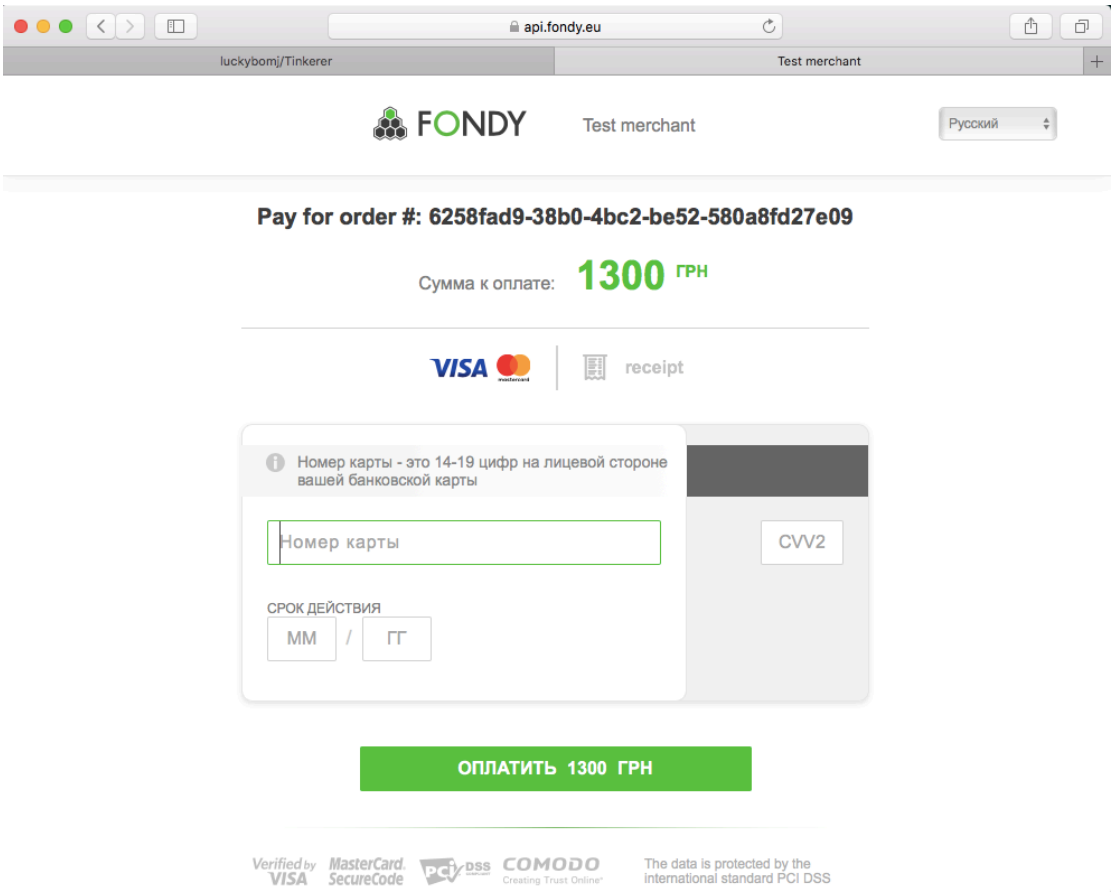

Рис.6 - Редірект користувача на сторінку тестової оплати

Після закінчення тестування та відладки програмного коду, проект підходить до свого завершення, а сам продукт може бути випущений.

По закінченню проекта, в якості особливостей проекту «Tinkerer» виступатимуть наступні функції:

1. Можливість купівлі пакетів послуг;
2. Вільне додання пакетів послуг;
3. Перегляд існуючих рішень запропонованих іншими користувачами;
4. Зручна система оплати.

ВИСНОВОК

В ході розробки курсової роботи з дисципліни «Веб-програмування та веб-дизайн» був створений веб сервіс «Tinkerer» для онлайн купівлі, чи продажу, різноманітних пакетів послуг, що дозволять користувачам за певну ціну збавитись від рутинної роботи та полегшити собі життя. Світ змінюється дуже залишатися в ритмі життя витрачаюи свій час саме на найбільш корисні для себе речі.

Даний програмний продукт був розробленій із застосування мови програмування Python та його веб-фреймворку Flask.

У якості бази даних було використано SQLite через її простоту та гнучкість, а також можливість інтеграції з Flask.

Необхідними пакетами та бібліотеками також стали CloudIPSP для інтеграції онлайн-купівлі, Flask-SQLAlchemy для вза’ємодії за базою даних на SQLite, а також фреймворк для CSS - Bootstrap.

Услі онлайн придбання реалізовані на базі Fondy через уже названу CloudIPSP.

Готовий програмний продукт може стати у нагоді багатьом користувачам, чий розморядок для не дозволяє витрачати багато часу на рутині справи, у той же час дозволяючи іншим допомогти їм з повсякденними справами за певну ціну.

ПЕРЕЛІК ПОСИЛАНЬ

1.Flask Extensions[Електронний ресурс].-http://flask.pocoo.org/ extensions/

2. Коэльё Л. П., Ричерт В. Построение систем машинного обучения на языке Python. — Перевод с английского. — М.: ДМК Пресс, 2015. — с. — ISBN 978-5-9706-0330-7.

3. Марк Саммерфилд. Программирование на Python 3. Подробное руководство. — Перевод с английского. — СПб.: Символ-Плюс, 2009. — 608 с — ISBN 978-5-93286-161-5

4.Список привязок SQLite для других языков [Електронний ресурс]. - http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers

5.И. А. Хахаев. Практикум по алгоритмизации и программированию на Python. Учебник. — М.: Альт Линукс, 2010. — 126 с. — (Библиотека ALT Linux). — ISBN 978-5-905167-02-7.

6.Дэвид М. Бизли. Python. Подробный справочник, 4-е издание. — Перевод с английского. — СПб.: Символ-Плюс, 2010. — 864 с — ISBN 978-5-93286-157-8

ДОДАТОК А

1. Зміст файлу main.py:

```
from flask import Flask, render_template, request, redirect
from flask_sqlalchemy import SQLAlchemy

from cloudipsp import Api, Checkout

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///products.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)



class Item(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    price = db.Column(db.Integer, nullable=False)
    isActive = db.Column(db.Boolean, default=True)
    service = db.Column(db.Text, nullable=False)

    def __repr__(self):
        return self.title



@app.route('/')
def index():
    items = Item.query.order_by(Item.price).all()
    return render_template('index.html', data=items)
```

```
@app.route('/about')
def about():
    return render_template('about.html')


@app.route('/buy/<int:id>')
def buy(id):
    item = Item.query.get(id)

    from cloudipsp import Api, Checkout
    api = Api(merchant_id=1396424,
              secret_key='test')
    checkout = Checkout(api=api)
    data = {
        "currency": "UAH",
        "amount": str(item.price) + "00"
    }
    url = checkout.url(data).get('checkout_url')
    return redirect(url)


@app.route('/create', methods=['POST','GET'])
def create():
    if request.method == "POST":
        title = request.form['title']
        price = request.form['price']
        service = request.form['service']

        item = Item(title = title, price = price, service = service)
        try:
```

```
                db.session.add(item)
                db.session.commit()
                return redirect('/')
            except:
                return "Что-то пошло не так. Ошибка добавления пакета услуг"
        else:
            return render_template('create.html')



if __name__=="__main__":
    app.run(debug=True)
```

2. Зміст файлу index.html:

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

{% block title  %}
<title>Tinkerer</title>
{% endblock %}


{% block body  %}
<main class="container">
  <div class="cart-deck mb-3 text-center">
        <div class="pricing-header px-3 py-3 pt-md-5 pb-md-4 mx-auto text-
center">
    <h1 class="display-4">Услуги</h1>
```

```
            <p class="lead">Лень делать что либо? Есть лишние деньги? Преобретай пакеты услуг, облегчи свою жизнь и избавься от нудной рутины! </p>
        </div>
        <ul class="custom-grid">
        {% for el in data%}

        <li class="row row-cols-1 row-cols-md-3 mb-3 text-center">
         <div class="col">
          <div class="card mb-4 shadow-sm">
          <div class="card-header">
           <h4 class="my-0 fw-normal">{{ el.title }}</h4>
          </div>
          <div class="card-body">
                    <h1 class="card-title pricing-card-title">{{el.price}} <small class="text-muted">Y.E.</small></h1>
           <ul class="list-unstyled mt-3 mb-4">
            <li>{{el.service}}</li>
           </ul>
                <a href="/buy/{{ el.id }}" class="w-100 btn btn-lg btn-outline-primary">Преобрести пакет услуг!</a>
          </div>
         </div>
         </div>
        </li>
        {% endfor %}
        </ul>
        </div>
       </main>
       {% endblock %}
```

3. Зміст файлу base.html:

```
<!doctype html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/
4.5.0/css/bootstrap.min.css">
  {% block title %} {% endblock %}
</head>
<body>


<header class="d-flex flex-column flex-md-row align-items-center p-3 px-
md-4 mb-3 bg-white border-bottom shadow-sm">
  <p class="h5 my-0 me-md-auto fw-normal">Tinkerer</p>
  <nav class="my-2 my-md-0 me-md-3">
    <a class="p-2 text-dark" href="/">Главная страница</a>
    <a class="p-2 text-dark" href="/about">О нас</a>

  </nav>
  <a class="btn btn-outline-primary" href="create">Добавить товар</a>
</header>
{% block body %} {% endblock %}
  <footer class="pt-4 my-md-5 pt-md-5 border-top">
    <div class="row">
      <div class="col-12 col-md">
```

```
                <img class="mb-2" src="/docs/5.0/assets/brand/bootstrap-logo.svg"
alt="" width="24" height="19">
                <small class="d-block mb-3 text-muted">© 2017-2020</small>
              </div>
              <div class="col-6 col-md">
                <h5>Features</h5>
                <ul class="list-unstyled text-small">
                  <li><a class="link-secondary" href="#">Cool stuff</a></li>
                  <li><a class="link-secondary" href="#">Random feature</a></li>
                  <li><a class="link-secondary" href="#">Team feature</a></li>
                  <li><a class="link-secondary" href="#">Stuff for developers</a></li>
                  <li><a class="link-secondary" href="#">Another one</a></li>
                  <li><a class="link-secondary" href="#">Last time</a></li>
                </ul>
              </div>
              <div class="col-6 col-md">
                <h5>Resources</h5>
                <ul class="list-unstyled text-small">
                  <li><a class="link-secondary" href="#">Resource</a></li>
                  <li><a class="link-secondary" href="#">Resource name</a></li>
                  <li><a class="link-secondary" href="#">Another resource</a></li>
                  <li><a class="link-secondary" href="#">Final resource</a></li>
                </ul>
              </div>
              <div class="col-6 col-md">
                <h5>About</h5>
                <ul class="list-unstyled text-small">
                  <li><a class="link-secondary" href="#">Team</a></li>
                  <li><a class="link-secondary" href="#">Locations</a></li>
                  <li><a class="link-secondary" href="#">Privacy</a></li>
                  <li><a class="link-secondary" href="#">Terms</a></li>
```

```
            </ul>
          </div>
        </div>
      </footer>
    </main>

    </body>
    </html>
```

4. Зміст файлу create.html:

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

{% block title  %}
<title>Tinkerer</title>
{% endblock %}



{% block body %}
<div class="conteiner">
    <h1>Добавлние пакет услуг</h1>
    <form method="post">
                <input type="text" class="form-control" name="title" id="title"
placeholder="Введите название услуги"></br>
            <input type="number" class="form-control" name="price" id="price"
placeholder="Введите стоимость пакета услуг"></br>
            <input type="text" class="form-control" name="service" id="service"
placeholder="Введите описание предоставляемого пакета услуг"></br>
             <button class="btn btn-success" type="submit">Добавить услугу</
button>
    </form>
```

```
</div>
{% endblock %}
```

USE CASE DIAGRAM

Змін	Лист	№ докум.	Підпис	Дата	ІС КР 122 АІ182 ПЗ	Лист 29

Посилання на GitHub repository:
https://github.com/luckybomj/Tinkerer