НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до курсового проекту з дисципліни
Бази даних та інформаційні системи

на тему: «WEB-сервіс для пошуку майстрів за геолокацією»

Студенти групи КМ-81
Горбач Костянтин Ігорович
Юр’єва Ксенія Геннадіївна
Піткевич Ілля Віталійович
Цурковський Сергій Олександрович

Керівник проекту Асистент Шияк Б. А.
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Київ – 2021

ЗМІСТ

РЕЗЮМЕ

У курсовому проєкті було розроблено веб сервіс, який відображає геолокацію останнього замовлення майстра (сантехніка, електрика, налаштувальника принтерів і Wi-Fi тощо) на карті, а замовники, в свою чергу, можуть обрати майстра, який закінчує (або закінчив) роботу неподалік. Таким чином майстрам не треба витрачати багато часу на пересування між замовниками, а замовникам не треба довго чекати на майстра.

Business Drivers

1. Люди витрачають багато часу на те, аби дочекатися виконання певної роботи вдома і їм треба дзвонити і домовлятися про послуги з майстрами.
2. Майстри (люди, що надають послуги на дому) витрачають багато часу на пересування між замовниками.

Business Goals

1. Скоротити час, який витрачає майстер на пересування між замовниками.
2. Пришвидшити виклик майстра додому.
3. Зменшити час очікування на майстра.

Business Objectives

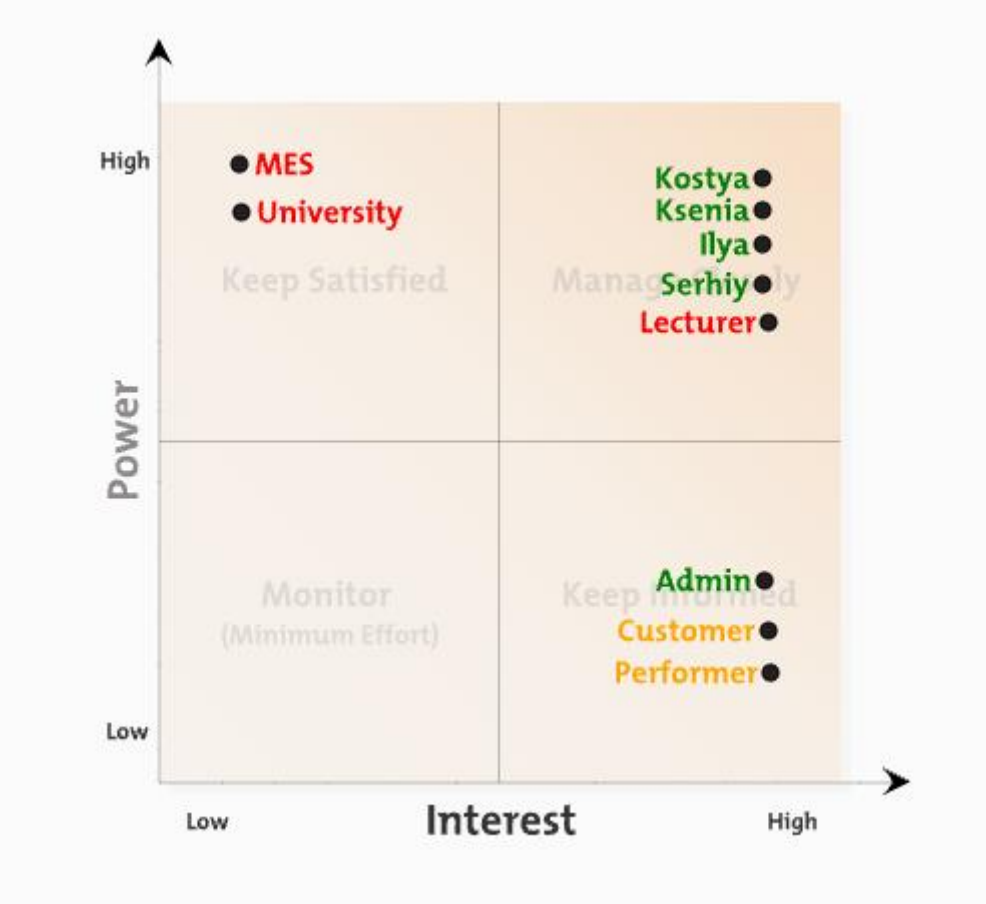
Створити веб сервіс, де майстри зможуть відмічати геолокацію, де вони закінчили останнє замовлення, а замовники зможуть замовляти послуги майстрів, що знаходяться поруч з ними.

ВСТУП

Терміни, Акроніми, Абревіатури

Термін	Визначення
Performer	Людина, що виконує роботу на дому (сантехнік, електрик, налаштувальник принтерів і Wi-Fi тощо)
Customer	Замовник
Order	Замовлення
Complaint	Скарга на результат виконання замовлення

Стейкхолдери



Перевірка на плагіат	University, MES
Архітектура веб-застосунку	Kostya, Ilya, Kseniia, Serhiy, Lecturer
Строки виконання	All, except MES
Середовище розгортання	Kostya, Ilya, Kseniia, Serhiy
Архітектура БД	Lecturer, Kostya, Ilya, Kseniia, Serhiy
Документація	Lecturer, Kostya, Ilya, Kseniia, Serhiy
Вибрані технології	Lecturer, Kostya, Ilya, Kseniia, Serhiy
Керівництво користувача	Admin, Customer, Performer
Організація кодової бази	Kostya, Ilya, Serhiy, Kseniia
Інсталяція та модернізація	Kostya, Ilya, Serhiy, Kseniia

Viewpoints

MES	Functional
University	Functional
Lecturer	Functional, Context, Information
Kostya	Development, Deployment, Context, Information, Concurrency, Operational
Serhiy	Development, Deployment, Context, Information, Concurrency, Operational

Kseniia	Development, Deployment, Context, Information, Concurrency, Operational
Ilya	Development, Deployment, Context, Information, Concurrency, Operational
Admin	Operational, Context
Customer	Context
Performer	Context

Мета

Люди витрачають багато часу на те, аби дочекатися виконання певної роботи вдома і їм треба дзвонити і домовлятися про послуги з майстрами. Майстри витрачають багато часу на пересування між замовниками.

Метою даного проєкту є створення веб сервісу, за допомогою використання якого зменшиться час, який майстер витрачає на пересування між замовниками, пришвидшиться та спроститься процедура виклику майстра додому, зменшиться час очікування на майстра.

ВИМОГИ

Функціональні вимоги

Ідентифікатор функціональної вимоги	Опис вимоги	Входить в MVP
FUN-01	Реєстрація користувачів	+
FUN-02	Пошук виконавця за критеріями	+
FUN-03	Розгляд виконавцем заявки	+
FUN-04	Створення угоди	+
FUN-05	Закінчення угоди	+
FUN-06	Створення скарги	
FUN-07	Виконавець може бути заблокований	
FUN-08	Замовник може бути заблокований	
FUN-09	Користувачі мають змогу редагувати власні дані	+
FUN-10	Перегляд власного кабінету користувачем	+
FUN-11	Додавання коментарів до замовлення	+
FUN-12	Вхідні дані перевіряються, при потребі екрануються	
FUN-13	Реалізоване API	+
FUN-14	Перегляд усіх виконавців на інтерактивній мапі	+

Quality attribute scenarios

	Source	Stimulus	Environment	Artifact	Response	Response measure
Ефективність	Майстер	Додавання геолокації	Під час звичайної роботи	Система	Геолокація обробляється	Із середньою затримкою у 10 секунд
Безпека	Авторизований користувач	Намагається змінити адресу	При нормальній роботі	Дані в системі	Перевірка введених даних	Правильні дані встановлюються протягом 10 секунд
Юзабіліті	Авторизований користувач	Вивчає особливості системи	Під час виконання	Система	Інтерфейс можна використовувати в незнайомому контексті	Виклик майстра займає до 5 хвилин

Нефункціональні вимоги

Код вимоги	Опис вимоги	
NFR-1	Manual test повинен покривати всю систему.	Тестування
NFR-2	Rest API системи має мати документацію.	Документація
NFR-3	Авторизований користувач має мати можливість викликати майстра за 5 кліків.	Юзабіліті
NFR-4	Інтуїтивно зрозумілий дизайн.	Юзабіліті
NFR-5	Паролі користувачів повинні бути захешовані.	Безпека
NFR-6	Майстер має отримувати інформацію не пізніше ніж через хвилину після замовлення послуги користувачем.	Час відгуку
NFR-7	Система має додавати дані про місцезнаходження майстра протягом 10 секунд	Час завантаження
NFR-8	Система повинна коректно працювати в браузерах Google Chrome, Opera, Mozilla Firefox, Internet Explorer, Safari.	Сумісність
NFR-9	В проекті використовується Rest API для взаємодії між серверною та Front-End частинами.	Інтегрування
NFR-10	Сторінка має завантажуватися ~ 2с.	Час відгуку
NFR-11	Система аутентифікації повинна бути побудована з використанням JWT	Безпека

NFR-12	Система має запитувати дозвіл у користувача для використання геоданих.	Згода
NFR-13	Система має працювати лише через https.	Безпека

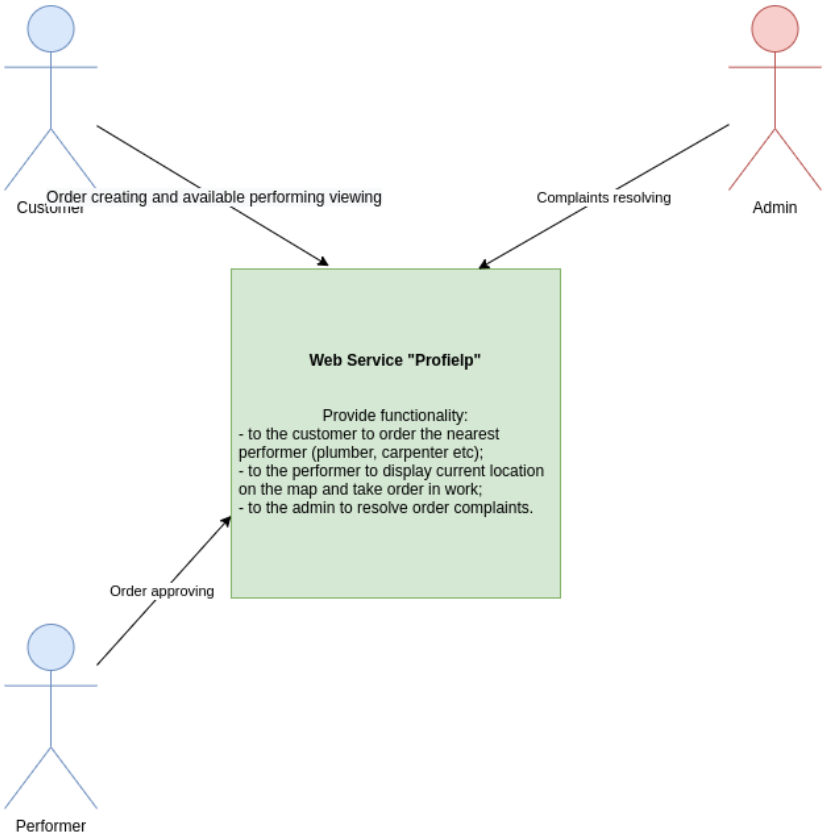
АРХІТЕКТУРНО ЗНАЧУЩІ ЯКІСНІ АТРИБУТИ

1. Система має бути розгорнута на Heroku (PCO)
2. Майстер має отримувати повідомлення не пізніше ніж через хвилину після замовлення послуги користувачем (NFR-6, безпосередньо впливає на зручність використання та швидкість роботи сервісу)
3. Система повинна коректно працювати в браузерах Google Chrome, Opera, Mozilla Firefox, Internet Explorer, Safari (NFR-8)

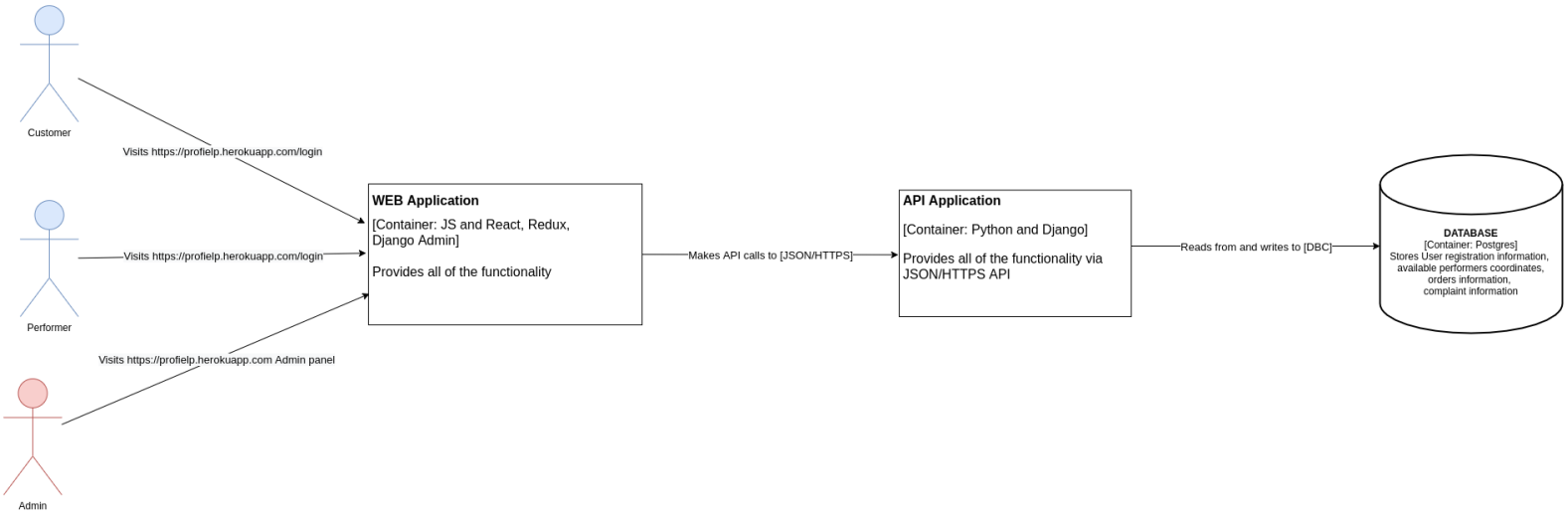
ПРЕДСТАВЛЕННЯ

C4 model

Context Diagram



Container Diagram



Component Diagram

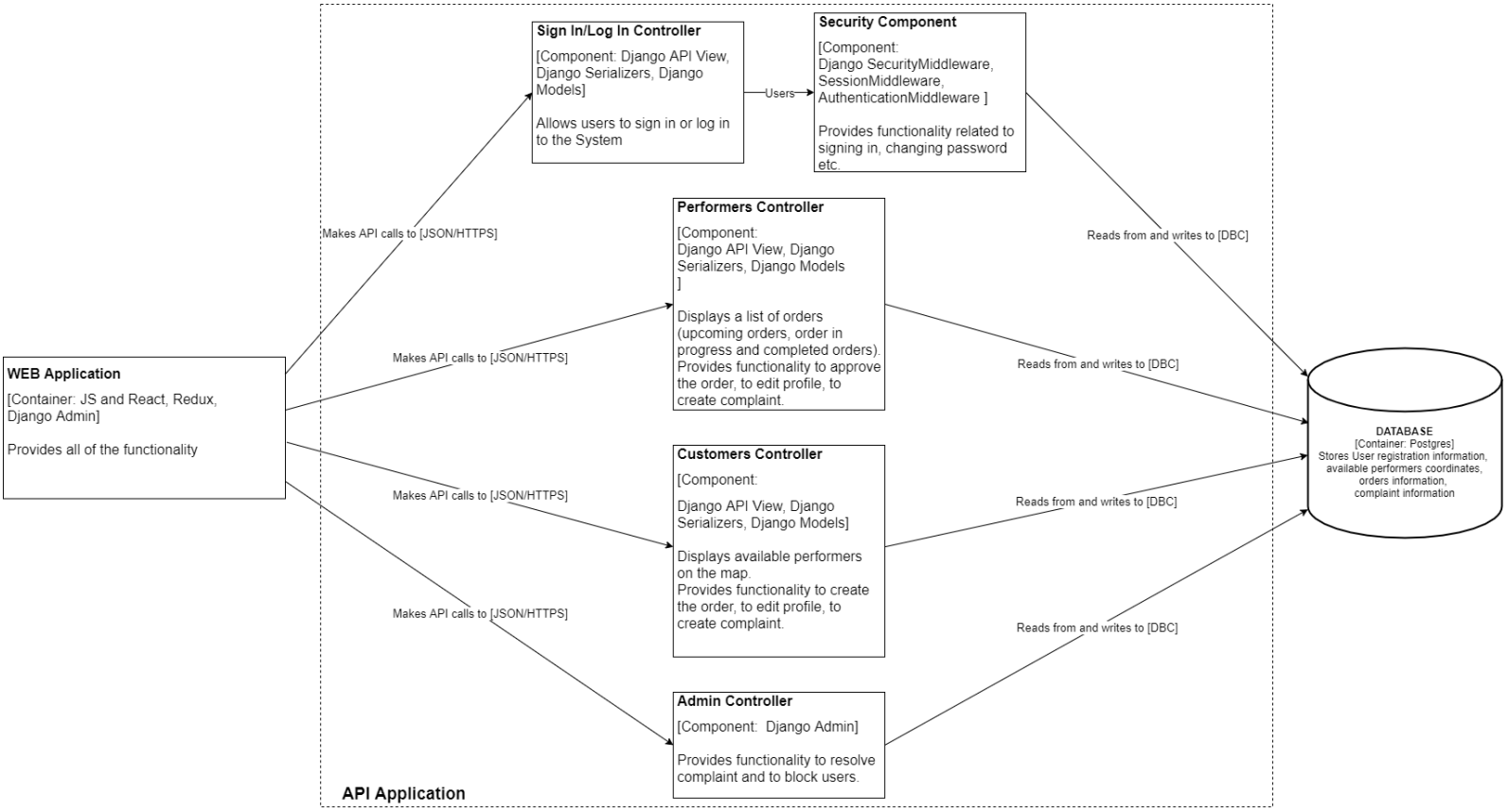
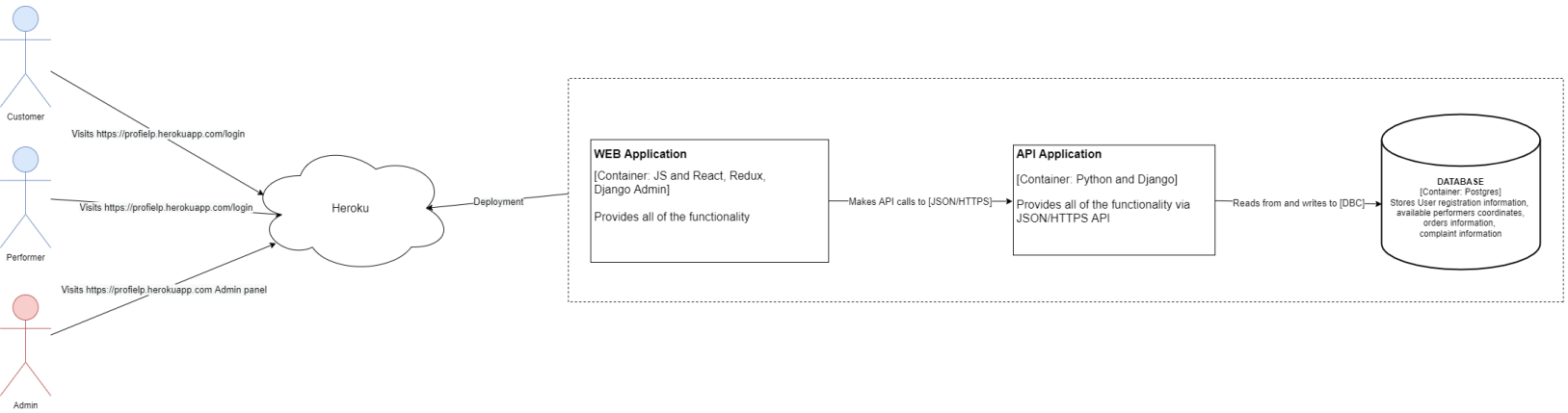


Схема розгортання системи



КОМПОНЕНТИ СИСТЕМИ

WEB Application

Опис	Надає всю Web UI функціональність для взаємодії з API Application.
Технологічний стек	JS and React, Redux, Django Admin
Пов'язані компоненти	Sign In/Log In Controller, Performers Controller, Customers Controller, Admin Controller
Покриті функціональні вимоги	FUN-01 – FUN-12, FUN-14

Sign In/Log In Controller

Опис	Дозволяє користувачам реєструватися та входити в систему.
Технологічний стек	Django API View, Django Serializers, Django Models
Пов'язані компоненти	WEB Application, Security Component
Покриті функціональні вимоги	FUN-01, FUN-13

Security Component

Опис	Надає функції, пов’язані з входом, зміною пароля тощо.
Технологічний стек	Django SecurityMiddleware, SessionMiddleware, AuthenticationMiddleware
Пов'язані компоненти	Sign In/Log In Controller, DATABASE
Покриті функціональні вимоги	FUN-09, FUN-12

Performers Controller

Опис	Відображає список замовлень (нові замовлення, замовлення в процесі виконання та виконані замовлення). Надає функціональні можливості для підтвердження замовлення, редагування профілю, створення скарги.
Технологічний стек	Django API View, Django Serializers, Django Models
Пов'язані компоненти	WEB Application, DATABASE
Покриті функціональні вимоги	FUN-03, FUN-05, FUN-06, FUN-09, FUN-10, FUN-13

Customers Controller

Опис	Відображає доступних виконавців на карті. Надає функціонал для створення замовлення, редагування профілю, створення скарги.
Технологічний стек	Django API View, Django Serializers, Django Models
Пов'язані компоненти	WEB Application, DATABASE
Покриті функціональні вимоги	FUN-02, FUN-04, FUN-05, FUN-06, FUN-09, FUN-10, FUN-11, FUN-13, FUN-14

Admin Controller

Опис	Надає функції для розгляду та вирішення скарги та блокування користувачів.
Технологічний стек	Django Admin
Пов'язані компоненти	WEB Application, DATABASE
Покриті функціональні вимоги	FUN-07, FUN-08

DATABASE

Опис	Зберігає реєстраційну інформацію користувача, координати доступних виконавців, інформацію про замовлення, інформацію про скарги.
Технологічний стек	PostgreSQL
Пов'язані компоненти	Security Component, Performers Controller, Customers Controller, Admin Controller
Покриті функціональні вимоги	FUN-01 – FUN-12, FUN-14

СТРУКТУРА ДЕКОМПОЗИЦІЇ РОБІТ

Номер WBS	Задача	Виконавець	Прогнозований час	Витрачений час
1	Front-end			
1.1	Розробка загального дизайну сайту	Ксенія	3 дні	2 дні
1.2	Створення сторінок реєстрації та власного кабінету	Ілля	4 дні	2 дні
1.3	Реалізація функції пошуку виконавця за критеріями	Ілля, Костя	3 дні	1 день
1.4	Створення сторінки з картою	Ілля	2 дні	1 день
1.4.1	Розробка інтерактивного функціоналу карти	Ілля	2 дні	3 дні
1.5	Створення сторінки для заключення угоди	Ілля	2 дні	1 день
1.6	Реалізація функції додавання коментарів до замовлення	Костя	2 дні	1 день
1.7	Реалізація функціоналу отримання списку угод	Ксенія	2 дні	2 дні
1.8	Реалізація функціоналу створення скарги	Ксенія	3 дні	2 дні
1.9	Розробка інтерфейсу адміністратора	Ілля	2 дні	1 день
2	Back-end			
2.1	Визначення структури API	Костя	2 дні	3 дні
2.2	Затвердження серверного функціоналу сайту	Костя	1 день	1 день
2.2.1	Реалізація функціоналу реєстрації користувачів	Ілля	2 дні	2 дні
2.2.2	Реалізація запиту пошуку виконавця	Ілля	3 дні	2 дні

2.2.3	Реалізація збереження локації виконавця	Ілля	2 дні	2 дні
2.2.4	Реалізація функціоналу отримання списку угод	Ксенія	2 дні	2 дні
2.2.5	Реалізація функції заключення та закриття угоди	Костя, Ксенія	3 дні	3 дні
2.2.6	Реалізація збереження скарги	Ксенія	2 дні	3 дні
2.3	Реалізація функції блокування користувачів адміністратором	Ілля	2 дні	0,5 дня
3	Database			
3.1	Розробка структури бази даних	Сергій	3 дні	4 дні
3.2	Розробка функціоналу бази даних	Сергій	3 дні	3 дні
3.3	Розширеня структури бази даних для нового функціоналу	Сергій	3 дні	3 дні
3.4	Мірація бази даних на нову структуру	Сергій	3 дні	3 дні
4	Тестування			
4.1	Ручне тестування функціоналу карти	Ксенія, Сергій	1 день	1 день
4.2	Ручне тестування функції реєстрації	Ксенія, Сергій	1 день	1 день
4.3	Ручне тестування функції заключення угоди	Ксенія, Сергій	1 день	1 день
4.4	Тестування всього функціоналу на клауд хостингу	Ксенія, Сергій	2 дні	2 дні
5	Завантаження сайту на хостинг			
5.1	Завантаження основних файлів	Ілля, Костя	1 день	1 день
5.2	Під'єднання бази даних	Сергій	1 день	1 день
5.3	Завантаження файлів нової версії	Ілля, Костя	1 день	1 день

6	Створення Docker контейнера			
6.1	Створення Docker контейнера для сайту	Ілля	1 день	0,5 дня
6.2	Створення Docker контейнера для бази даних	Ілля	1 день	0,5 дня
7	Документація			
7.1	Підключення Swagger API	Костя	1 день	1 день
7.2	Написання пояснювальної записки	Ксенія, Сергій	1 день	3 дні
7.3	Створення презентації	Ксенія	2 дні	1 день

РЕЗУЛЬТАТИ ПЕРЕВІРКИ ВІДПОВІДНОСТІ ВИМОГАМ

- ✓ Розроблена система відповідає принципам "12 factor apps"

1. Кодова база

Одна кодова база, що відслідковується в системі контролю версій та має багато розгортань.

2. Залежності

Існує файл з залежностями requirements.txt

3. Конфігурація

Існує файл зі змінними середовища для Prod та Dev Envs (.prod.env, .dev.env)

4. Сторонні служби

Підключена база даних PostgreSQL, яка підключається за допомогою Database URL.

5. Збірка, реліз, виконання

Використовується мова, що інтерпретується, а не компілюється.

6. Процеси

Застосунок запускається за допомогою ProcFile з використанням Gunicorn.

7. Прив’язка портів

Heroku прив’язує порти автоматично

8. Конкурентність

Heroku автоматично масштабує застосунок за допомогою процесів.

9. Утилізовуваність

Застосунок запускається швидко та коректно вимикається за рахунок того, що він розгорнутий на Heroku та є Stateless.

10. Dev/prod паритет

Development та production середовища є досить ідентичними, оскільки dev гілка зливається (merge) з гілкою master та перед розгортанням на production сервіс тестується локальна за допомогою Heroku CLI.

11. Журналювання

У даному Web-застосунку є журналювання на рівні Heroku та на рівні застосунку.

12. Задачі адміністрування

Наприклад, міграція бази даних здійснюється в тому ж середовищі, що і застосунок, за допомогою однієї команди.

- ✓ Реалізація в системі паралельної роботи декількох користувачів:

Реалізовані 4 ролі (customer, performer, administrator) та SuperUser може створювати адміністраторів з різними рівнями доступів.

- ✓ Реалізація в системі валідації даних на рівнях UI, backend, DB:

Уся інформація користувачів (номер телефону, email тощо) перевіряється на коректність на стороні UI та Back-end.

- ✓ Реалізація в системі REST API:

Rest API level 2 - під кожен ресурс є окремий URL, який підтримує різні HTTP методи (POST, GET, DELETE, PATCH, PUT, OPTIONS).

Додаток 1. Посилання

- Посилання на GitHub: https://github.com/Wipersee/profielp
- Посилання на WEB-застосунок: https://profielp.herokuapp.com