Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«СИСТЕМА УПРАВЛІННЯ РЕЗЕРВУВАННЯМ РЕСУРСІВ (ЗА ПРИКЛАДОМ СЕРВІСУ AIRBNB) (DEVOPS)»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп’ютерні науки»

Виконавці проекту

№	П.І.Б.	Група
1	Карпов Д.Р.	КН-П-201
2	Лупашко П.П.	КН-П-201
3	Козлов Я.Я.	КН-Д-202
4	Воробйов А.І.	КН-Д-202
5	Гольденберг В.О	КН-А-201

Автор звіту:

_____ Гольденберг Віктор Олександрович

Одеса – 2024

АНОТАЦІЯ

Проект був започаткований для створення нової системи онлайн-бронювання, яка базується на існуючій платформі airbnb.com. Функціональні можливості та інтерфейс користувача були модернізовані. Розробка нової системи розпочалася з урахуванням вимог керівників департаментів та основної концепції поточної системи airbnb.com.

Ця робота описує дослідження, розробку та тестування системи бронювання, що має на меті полегшити процес купівлі-продажу послуг між користувачами та компаніями. Система пропонує широкий спектр функцій, зокрема:

- **Реєстрація та авторизація:** Користувачі та компанії можуть легко створити облікові записи та отримати доступ до системи.
- **Аутентифікація:** Надійні механізми аутентифікації гарантують безпеку користувачів та захист їхніх даних.
- **Оренда:** Користувачі можуть шукати та бронювати послуги, пропоновані компаніями, на платформі.
- **Додавання та управління готелями:** Компанії можуть додавати інформацію про свої готелі, керувати ними та приймати онлайн-бронювання.
- **Доступ до інформації:** Користувачі та компанії мають доступ до інформації про свої бронювання, клієнтів та зароблені кошти.

Головними завданнями були покращення зручності використання інтерфейсу системи. У цій роботі об'єктом дослідження є алгоритм, що забезпечує комплексну оцінку результатів за численними критеріями в умовах невизначеності початкових даних. Предмет дослідження включає фактор, який дозволяє користувачеві легко подорожувати та комбінувати об'єкти і рейси найпростішим можливим способом.

Цілями дослідження є:

- Огляд систем бронювання, що дозволяють користувачам взаємодіяти з іншими користувачами та компаніями без необхідності проходити складні процедури.
- Розробка програмного забезпечення з урахуванням досягнутих раніше результатів, а також проведення його тестування в різних умовах і на різних пристроях.
- Впровадження програмного забезпечення для захисту даних у систему з метою підвищення рівня інформаційної безпеки створеної системи.
- Методи дослідження, що застосовуються для досягнення цілей, базуються на комбінації методів системного аналізу, моделювання та когнітивної методології.

Для розробки клієнтської частини було використано веб-фреймворк Angular разом з HTML, TypeScript і SCSS. Ця дослідницька робота зосереджена на створенні системи бронювання з підтримкою специфічного пошукового фільтру.

ЗМІСТ

ВСТУП

Сьогодення показує нам, що все більше йде діджиталізація та все більше пришвидшується темп життя. Люди все менше прив’язані до певного місця, будь це квартира, власний будинок, офіс тощо. Ми все більше переносимо роботу, хобі, відпочинок та розваги у віртуальну сферу. Відсоток робітників, які обирають дистанційний формат співпраці, роботу через фріланс-платформи або власну справу в інтернет-просторі зростає з кожним роком. Цим людям відкривається можливість більш вільно переміщатися як по місту так і за його межами. Це у свою чергу призводить до можливості працювати з різних місць країни й навіть світу. Ці люди спонукають розвиток платформ для оренди житла. При чому розвивається як і короткочасна (туризм, ділові відрядження) так і довгострокова оренда (постійне проживання, вимушене переміщення у зв’язку з геополітичними обставинами).

Виходячи з вищевказаної проблематики та факторів впливу, платформи потребують зручного та гнучкого інструментарію для користувачів та надавачів послуг. Даний проект направлений на розробку платформи для онлайн бронювання житла, яка буде більш адаптована для терен нашої країни. А саме враховувати особливості туризму, надання послуг оренди й озираючись на ситуацію в країні. Але розробка виконується з розрахунком на подальший розвиток на зовнішньому ринку і виходу за межі нашої країни.

Платформа виконує задачу зведення в одному місці пропозиції від надавачів послуг оренди. Це допомагає бізнесу розвиватись та спонукає до здорової конкуренції та регулювання регіональної цінової політики. Для клієнтів поліпшує пошук, прогнозування, бронювання та взаємодію з обраним житлом.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

СИСТЕМА УПРАВЛІННЯ РЕЗЕРВУВАННЯМ
РЕСУРСІВ (за прикладом сервісу airbnb.com)

Загальний опис.

Програмне забезпечення (далі – ПЗ) призначене для створення розподіленої системи керування ресурсами, що забезпечує можливість проактивного управління, включаючи резервування. Функції системи покривають повний життєвий цикл управління ресурсами: створення, актуалізація, використання та видалення. Головна особливість системи полягає в тому, що пріоритет використання обмежених ресурсів залежить від терміну їх резервування.

Призначення:

Програмне забезпечення орієнтоване на створення контейнера для управління обмеженими ресурсами на основі аналізу їх резервування. Обробка конкурентних запитів на обмежені ресурси повинна враховувати показники пріоритетності, серед яких важливим є часовий показник моменту резервування. ПЗ розробляється за прикладом сервісу бронювання обмеженої кількості місць, але має бути універсальним і здатним змінювати своє цільове призначення.

Вимоги до функціоналу. Обов'язкова частина

Головна сторінка

На головній сторінці системи забезпечено відображення фільтрів, пошуку і каталогу апартаментів, що значно полегшує користувачам доступ до необхідної інформації та сприяє їхній ефективній навігації по сервісу. Завдяки цим функціональним можливостям користувачі можуть швидко

знаходити потрібні об'єкти для бронювання, використовуючи зручні фільтри і швидкий пошук.

Крім того, на головній сторінці знаходяться інструменти для авторизації та реєстрації нових користувачів або посилання на них, що робить процес взаємодії з системою ще більш зручним і простим. Це дозволяє новим користувачам швидко і без зайвих зусиль зареєструватися та отримати доступ до усіх функцій сервісу.

Реєстрація користувачів:

Користувач ПЗ має змогу зареєструватись як за допомогою авторизаційних даних (логіну/паролю), так і за допомогою облікових записів популярних соціальних мереж.

При реєстрації обов'язково зазначається засіб оперативного контактування (далі – ЗОК) - телефон або електронна пошта. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

Авторизація та особистий кабінет користувача:

За фактом успішної авторизації користувачу надається доступ до особистого кабінету. Це окремий інформаційний ресурс (сторінка), що являє собою основне робоче місце користувача ПЗ. Він повинен забезпечити усі необхідні засоби для створення замовлень.

Також засобами особистого кабінету має бути передбачено можливість зміни персональних даних.

Процес бронювання:

Користувач повинен мати можливість перейти на сторінку будь-якого із запропонованих на головній сторінці готелів, отримати на ній ціну за ніч, опис, фото та всю іншу інформацію про готель. Користувач має бути забезпечений інтерфейсом для вибору дат і кількості гостей для бронювання на сторінці апартаментів. Після заповнення цих даних, має бути сторінка з необхідною інформацією для здійснення бронювання, а саме інформація про банківську картку, повідомлення компанії готелю, фото і номер телефону.

Вимоги до функціоналу. Опціональна частина

Розміщення та управління

Компанія має можливість додати свій готель або апартаменти через спеціальну сторінку, де вона заповнює всю необхідну інформацію про об'єкт. Ця процедура не лише спрощує процес реєстрації нових об'єктів, але й забезпечує повну інтеграцію з системою управління, що дозволяє компанії ефективно керувати всіма аспектами своєї гостьової діяльності.

Крім цього, компанія має доступ до спеціальної сторінки хостингу, де вона може здійснювати моніторинг і управління своїми готелями і клієнтами. Ця функція не тільки сприяє оптимальному використанню ресурсів і підвищує ефективність обслуговування гостей, але і забезпечує компанії зручний інтерфейс для взаємодії з її власною мережею готелів.

Архітектура

ПЗ має бути спроектоване за розподіленою клієнт-серверною архітектурою з чітко відокремленими розподіленими вузлами системи:

Mockup - візуальна модель інтерфейсу користувача із зазначенням зв'язків і переходів між різними частинами інтерфейсу.

Backend - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.

Web App - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій.

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- Backend
- Web
- Mockup (вихідні файли або посилання на зовнішній проект)

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Web - Angular, TypeScript, SCSS, HTML

Mockup - визначається ментором з дизайну

Команда проєкту

Для виконання поставлених завдань передбачається робота команди

розробників окремо за кожним з модулів проекту.

Мінімальна команда проекту - два учасники, у такому складі один учасник виконує модуль Backend, другий - Frontend.

Рекомендована команда проекту - чотири учасники, у такій команді кожен з учасників відповідає за одну з частин: Mockup, Backend, Frontend та Devops відповідно.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проекту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носієві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

Автореферат містить описову частину, у якій розкриваються основні концепти проекту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Архітектура платформи складається з таких частин:

«Frontend»,

«Design»,

«Backend»,

«DevOps».

«DEVOPS»

В частині проекту DevOps за основу були взяті наступні системи та практики:

- Розподілення ресурсів
- CI/CD
- Контейнеризації
- Мікросервісність
- Контроль версій (GitHub)
- Хмарні сервіси

Розподілення ресурсів

Наразі для більшої стресостійкості та відмовостійкості різні частини проекту розгортають на різних серверах або в окремих контейнерах. Це дозволяє робити резервування ресурсу на інших серверах (дзеркалювання), або надавати резервний доступ до платформи. Також це дає змогу частково змінювати проект без впливу на інші частини всієї конструкції та не перериваючи доступ до сайту. В даній роботі це представлено розподіленням архітектури на різних серверах. В роботі використовується 3 сервера:

- Сервер з Jenkins
- Сервер з СКБД (Системою контролю баз даних) та моніторингом

- Сервер з 2 контейнера в яких розподілений Frontend та Backend проекту.

CI/CD – набір інструментів, що використовуються для автоматизації процесу розроблення ПЗ.

Розшифрування абревіатур:

CI (Continuous Integration) - Безперервна інтеграція;

CD (Continuous Deployment) - Безперервне розгортання.

Контейнеризація – технологія архівації мінімально потрібного ПЗ для запуску коду.

Плюси контейнеризації:

- Швидке розгортання сервісу;

- Контейнер завжди зберігає один і той самий код, що корисно коли потрібно дотримуватись однієї версії ПЗ;

- В контейнер можна покласти всі потрібні програми та залежності між ними, що допомагає комплексному розгортанню всього необхідного без впливання людського фактору (неправильного налаштування або елементарного загублення зв’язку між програмами);

- Контейнери ізольовані один від одного і від системи, де були розгорнуті. Це додає стабільності при збоях.

Мікросервіси – практика, при якій використовується набір різних програм для досягнення поставленої мети. Дає можливість гнучко налаштовувати систему під різноманітні задачі та не руйнує архітектуру при заміні будь-якого з сервісів.

Контроль версій – звірка між білдами кода та відображення змін, що допомагає перевіряти стабільність роботу коду та швидко відкатувати зміни за потреби. Також це дозволяє ввести окремі гілки проекту. Зазвичай їх мінімум 2: тестова та головна, в яку вносяться протестовані зміни

Хмарні сервіси – віртуальні сервери, які запускаються на потужностях відповідних сервісів. Вони підходять для тестування фірмами, у яких немає бюджету для власних серверів та навчальних білдів. За допомогою цього можна просто і гнучко масштабувати розміри серверів та їхніх ресурсів.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ DEVOPS ЧАСТИНИ

Опис розробки

При отриманні ТЗ був обраний варіант мікросервісної архітектури. Для даного проекту були обрані наступні сервіси: GitHub, Jenkins, Mysql, Prometheus, Grafana Docker AWS EC2, Oracle Cloud

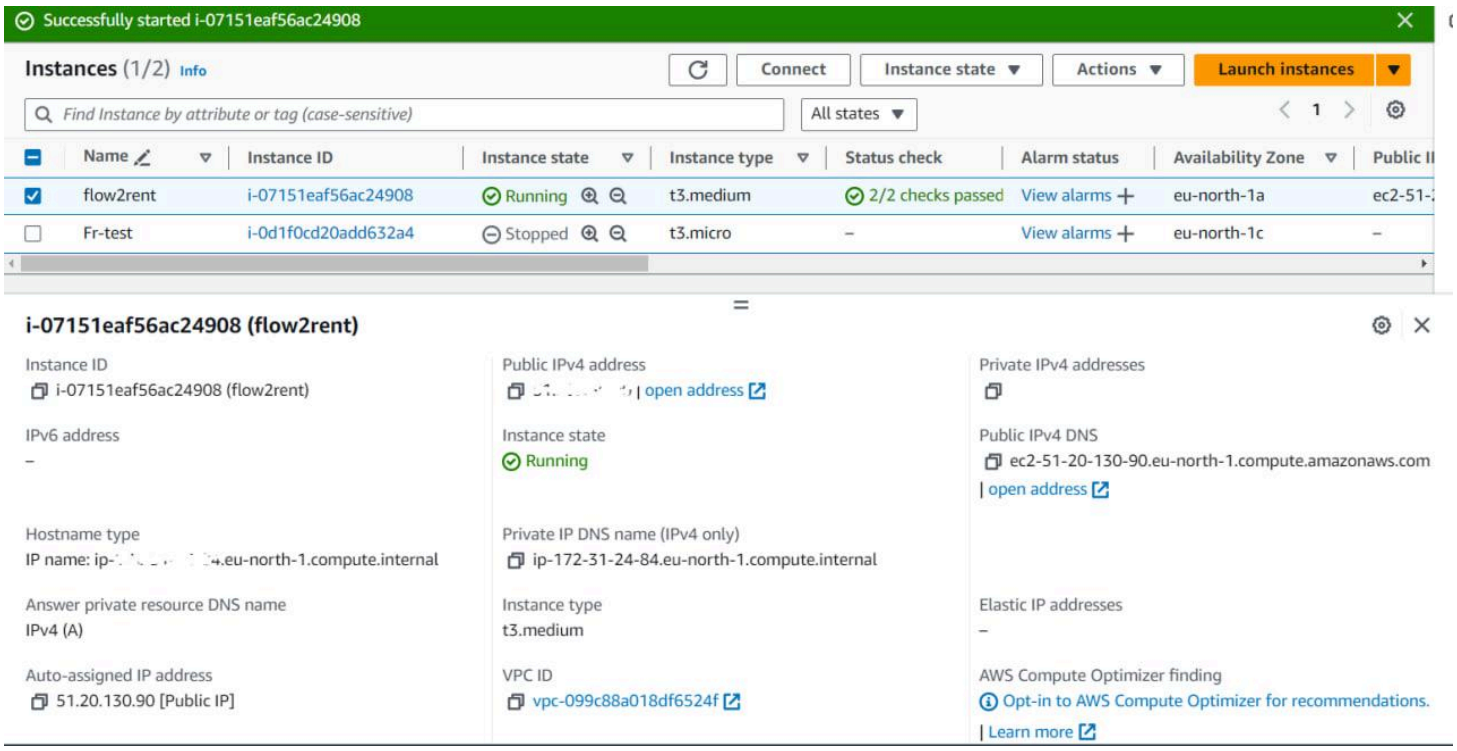
Використані мікросервіси

AWS та Oracle Cloud – системи хмарних сервісів, які дозволяють піднімати сервера в хмарному середовищі

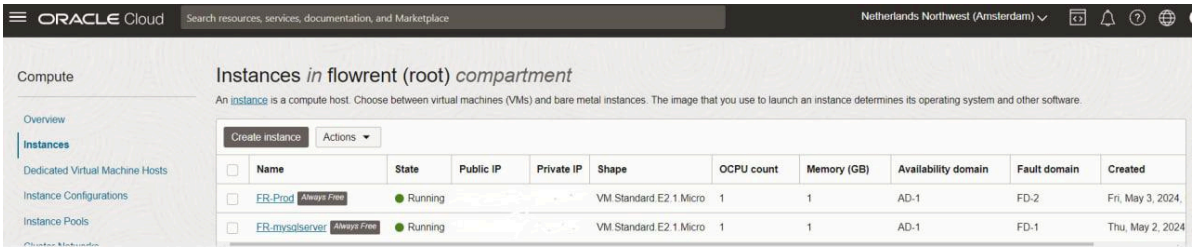
Задачі хмарних серверів

1. Запуск серверів

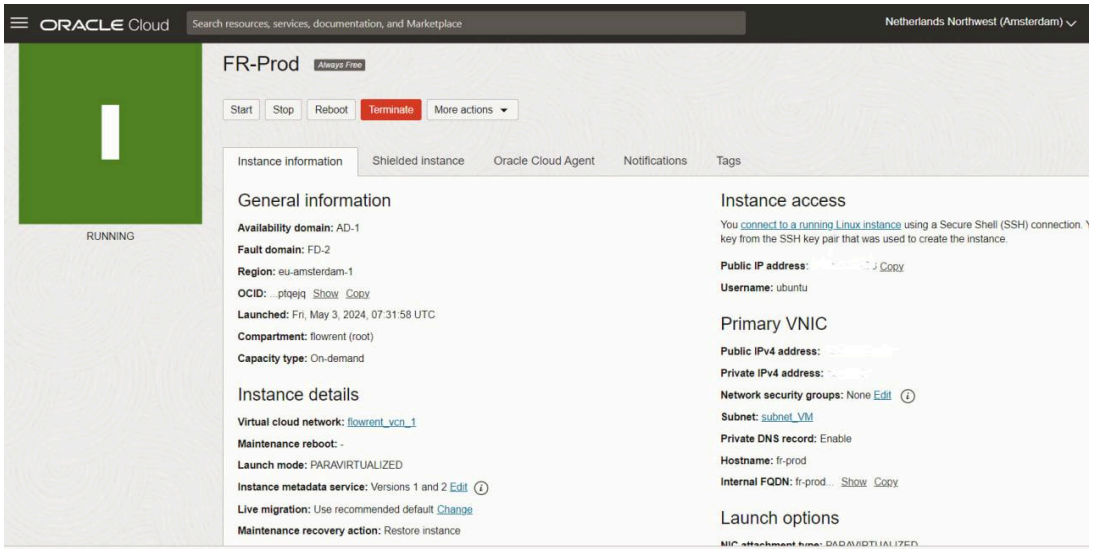
2. Цілодобова доступність серверів



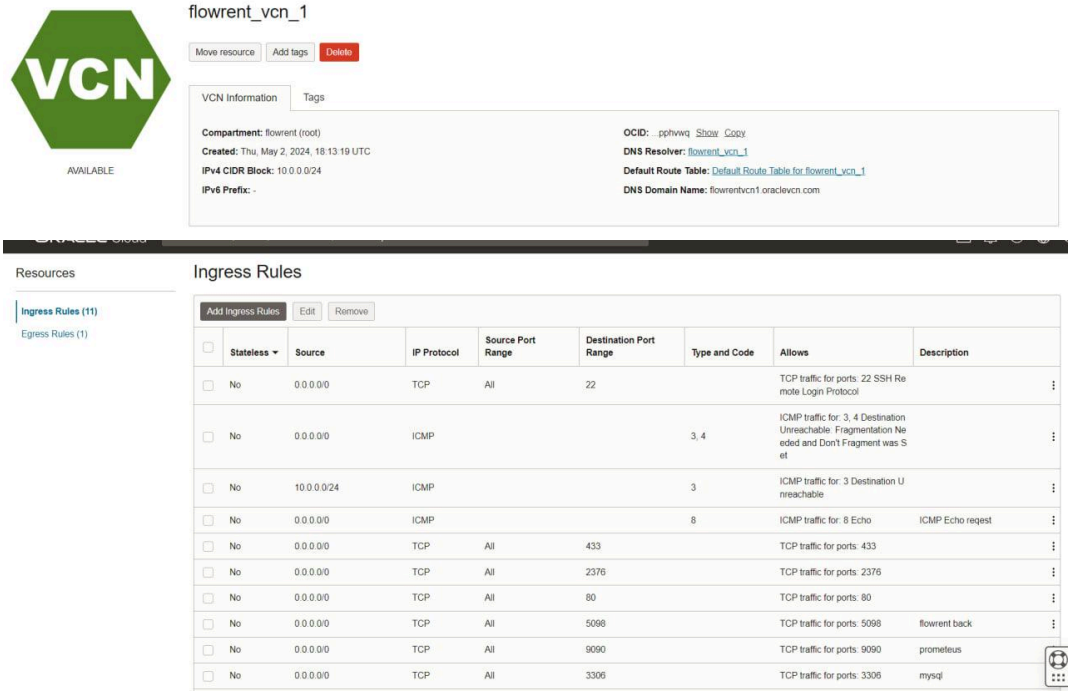
Малюнок 1. Інтерфейс керування серверами AWS



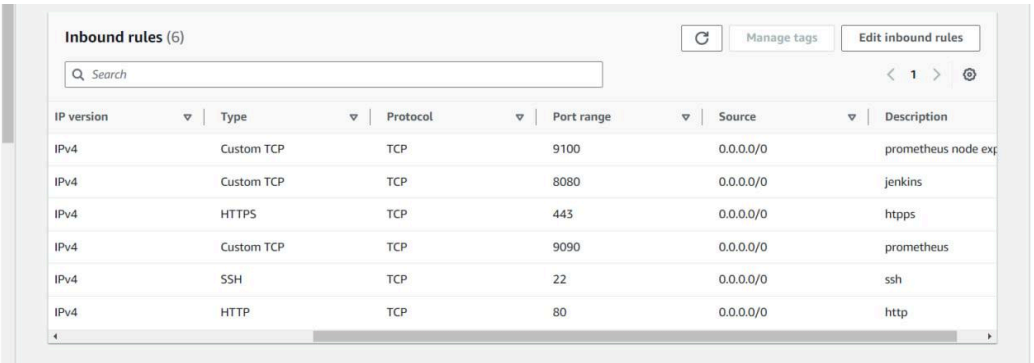
Малюнок 2. Інтерфейс керування серверами Oracle Cloud



Малюнок 3. Інтерфейс керування сервера через Oracle Cloud



Малюнок 4 та 5. Налаштування портів для доступу в Oracle Cloud



Малюнок 6. Налаштування портів для доступу в AWS

Jenkins – ПЗ для налаштування конвеєра та дотримання системи CI/CD

Задачі Jenkins:

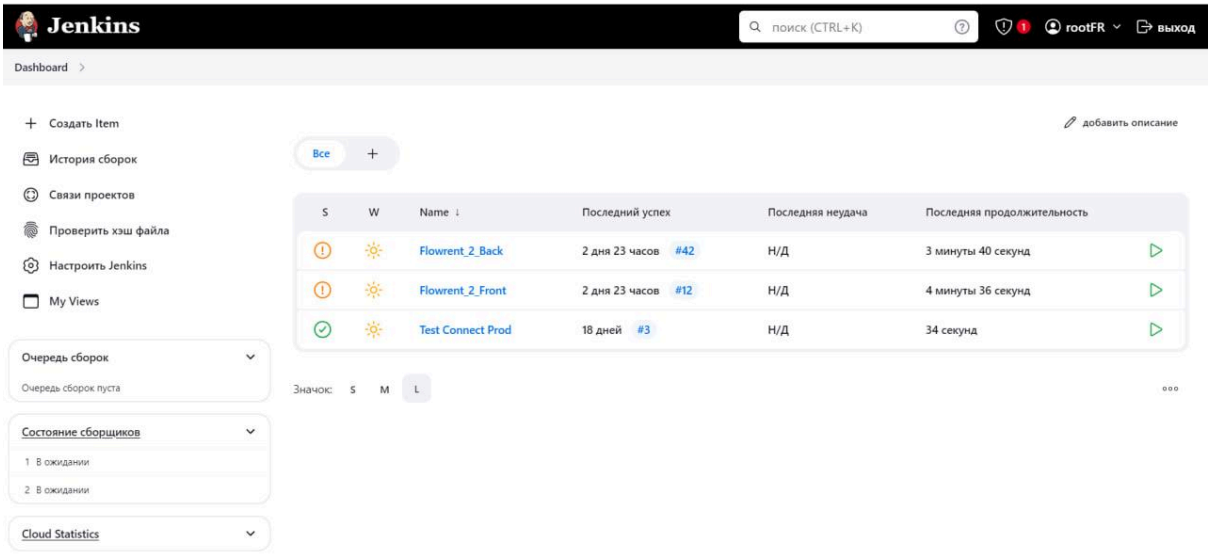
1. Збирання коду – запускається і виконується код, написаний відділом програмістів;

2. Тестування коду – перевірка коду на відсутність помилок;

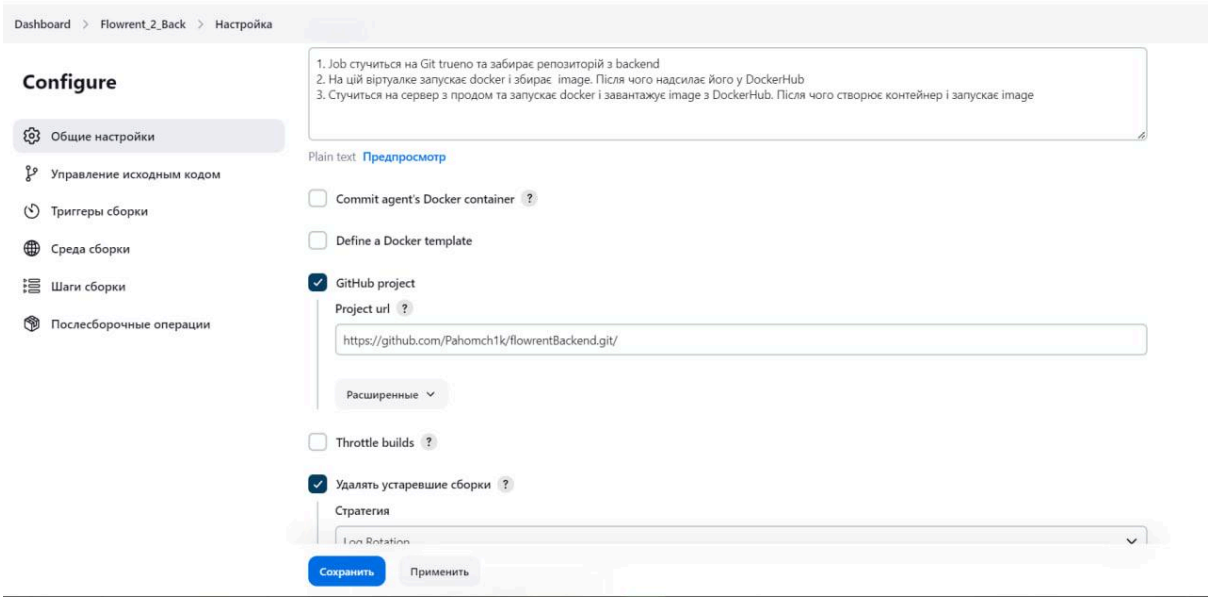
3. Збирання DockerFile – завантаження всіх потрібних елементів для роботи платформи та пакування в один контейнер

4. Відправка на головний сервер - виконується розгортання проекту готового для роботи клієнта;

5. Автоматична перевірка оновлень коду - при появі відповідних йде автоматичне перезбираня контейнеру та оновлення білда на сервері з продуктом.



Малюнок 7. Інтерфейс керування Jenkins



Малюнок 8. Налаштування pipeline

Docker – ПЗ яке дозволяє реалізовувати метод контейнеризації.

Задачі Docker:

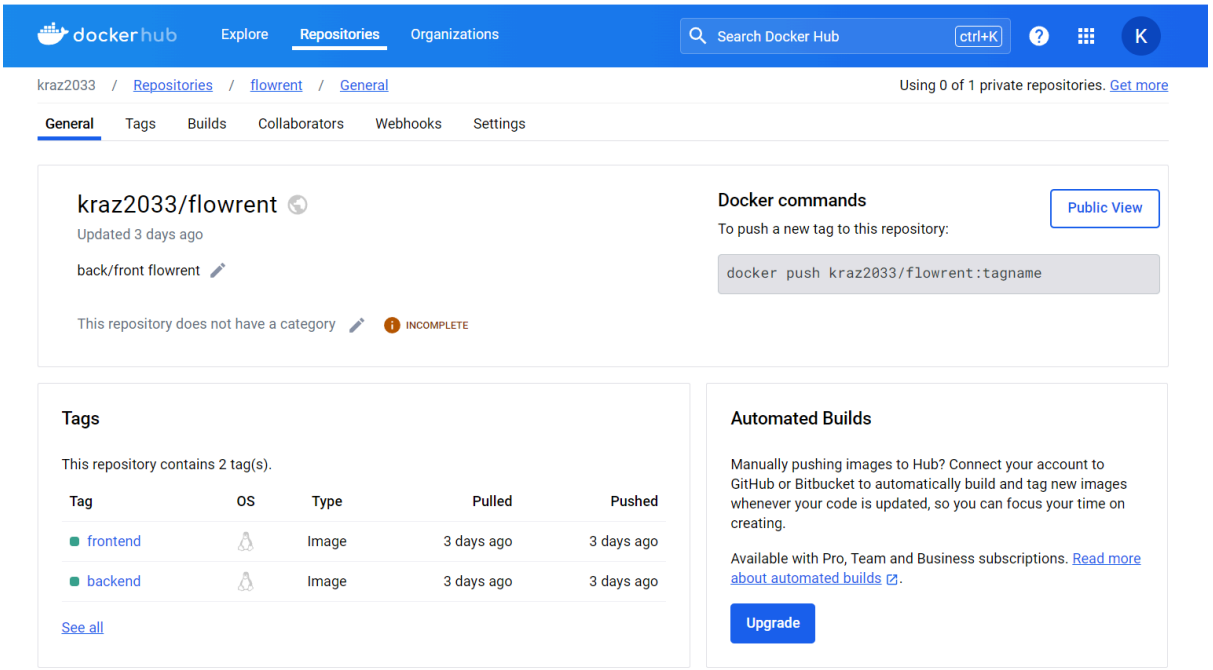
1. Збирання контейнера по зазначеному списку інструкцій (Docker File);

2. Взаємодія з хмарним сховищем DockerHub. Завантаження на хмарне сховище та вивантаження з нього нових контейнерів;

3. Завантаження, адміністрування та оновлення контейнерів при появі новішого білда;

4. Запуск контейнерів з зазначеними параметрами;

MySql – СКДБ (Система контролю баз даних). Потрібна для створення, зберігання та адміністрування БД.



Малюнок 9. Інтерфейс DockerHub з завантаженими image проекту

Задачі MySql:

1. Запис нових даних про клієнтів платформи та про нові готелі, хостели, квартири, тощо;

2. Зберігання вже внесених даних попередніми користувачами;

3. Створення потрібних нових таблиць під нові задачі та нові розділи сайту;

4. Надання та фільтрування інформації за запитом від користувачів або модераторів.

```
ubuntu@Fr-mysqlserver:~/.ssh$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.37-0ubuntu0.22.04.3 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| flowrent           |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql>
```

Малюнок 10. Інтерфейс MySql

```
mysql> Show tables from flowrent;
+-----------------------+
| Tables_in_flowrent    |
+-----------------------+
| __EFMigrationsHistory |
| aspnetroleclaims      |
| aspnetroles           |
| aspnetuserclaims      |
| aspnetuserlogins      |
| aspnetuserroles       |
| aspnetusers           |
| aspnetusertokens      |
| countries             |
| image                 |
| reviews               |
| stays                 |
| tags                  |
| whishlistcategories   |
| whishlistitems        |
+-----------------------+
15 rows in set (0.00 sec)

mysql>
```

Малюнок 11. Структура таблиць БД проекту

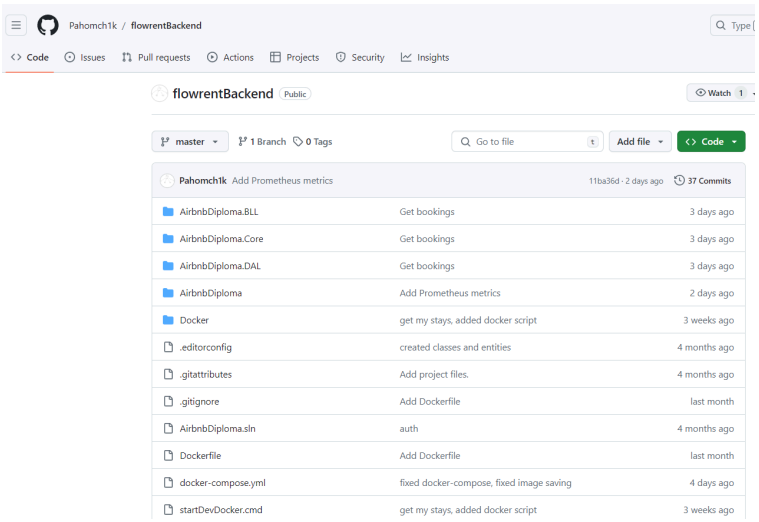
GitHub – платформа для хмарного зберігання та управління репозиторіями коду

Задачі GitHub:

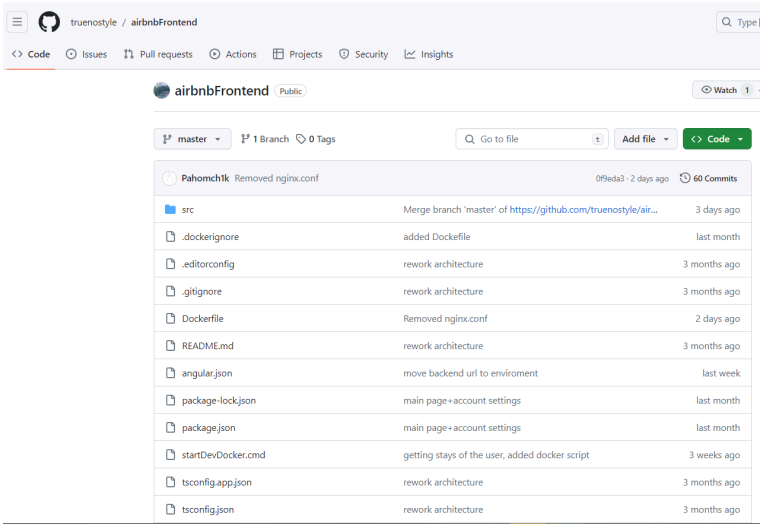
1. Зберігання репозиторіїв;

2. Доступ до сумісної роботи над одним репозиторієм;

3. Інтеграція з CI/CD.



Малюнок 12. Репозиторій BackEnd коду проекту



Малюнок 13. Репозиторій FrontEnd коду проекту

Prometheus – ПЗ для збору даних про систему або її складові.

Задачі Prometheus:

1. Відстеження стану серверів, контейнерів, баз даних та інших компонентів інфраструктури;

2. Збір метрик та інформації про продуктивність і стан додатків;

3. Аналіз даних за тривалий період для виявлення трендів і закономірностей, що можуть допомогти в оптимізації роботи системи.

Grafana – Пз для візуалізації метрик та сповіщень.

Задачі Grafana:

1. Візуалізація даних через дашборди для відображення метрик з різних джерел (в нашому випадку Prometheus);

2. Агрегація даних з різних серверів;

3. Відображення метрик в реальному часі;

4. Створення сповіщень на основі заданих умов звітування.

ПЕРСПЕКТИВИ

AWS та Oracle

Наразі система підготовлена та адаптована під подальше розширення.
Першим кроком буде перехід на більші потужності всіх серверів. Зміна їхніх розмірів дасть змогу швидше опрацьовувати запити та зберігати більші об’єми даних. Також це дасть змогу розширити платформу для нових функцій або нових розділів.
За рахунок того, що наразі всі сервера знаходяться вже у хмарних середовищах, збільшення розмірів робиться програмно.

Kubernetes

При розширенні системи бажаний перехід з Docker на Kubernetes. Це дозволить оптимізувати процеси при збільшенні кількості контейнерів. Також ми не втрачаємо попередні наробки по контейнерам, бо Kubernetes підтримує контейнери створені Docker.

MySql

В подальшому треба буде підключити візуалізацію СКБД. Скоріш за все, буде обрано PhpMyAdmin за простоту, відкритий код та зручність у користуванні.

Amazon S3 Bucket

При збільшенні кількості фото-/відеоматеріалів треба буде піднімати окремий сервіс для зберігання файлів. Гарним варіантом буде S3 Bucket від Amazon. Більшість сервісів вже адаптовані під роботу з ним, що допоможе інтеграції.

Jenkins

В майбутньому через збільшення задач та одночасно виконуваних процесів, треба буде розширювати його за рахунок додаткових серверів (Nodes).

ВИСНОВКИ

Розробка проекту з клонування Airbnb була багатогранним завданням, яке вимагає використання широкого спектра технологій і методологій. Дана робота відповідає за побудову та підтримку інфраструктури для забезпечення ефективної роботи всіх компонентів системи.

У процесі роботи були використані наступні платформи:

Jenkins - автоматизація процесів CI/CD, яка дозволила швидко інтегрувати зміни коду, забезпечуючи безперервну доставку нових версій продукту.

Docker - контейнеризація частин платформи для ізоляції під час виконання, що спростило розгортання проекту.

MySQL - БД для зберігання та обробки даних про користувачів та бронювання, яка забезпечила високу продуктивність.

Prometheus та Grafana - моніторинг та візуалізація стану системи в режимі реального часу.

AWS та Oracle Cloud - хмарні платформи для розміщення та управління інфраструктурою, що дало змогу мати постійний доступ до додатку.

Github - система контролю версій для відстеження змін та координації роботи.

Проект було розгорнуто на трьох серверах, що дозволило забезпечити балансування навантаження та резервування сервісів, а також підвищити стійкість системи.

Застосування практик DevOps дозволило досягти автоматизації процесів розгортання та керування, що значно зменшило час введення нових версій і підвищило якість кінцевого продукту.

Результат роботи показав, що використання комплексного підходу DevOps дозволяє створювати надійні та ефективні системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

№	№ п.п. Ресурс
1	GitHub Розробник: GitLab, Inc. https://docs.github.com/
2	Amazon Web Services Розробник: AWS, Inc. https://docs.aws.amazon.com/
3	Oracle Cloud Розробник: Oracle, Inc. https://docs.oracle.com/en/cloud
4	Docker Розробник: Docker, Inc. https://docs.docker.com/
5	Jenkins Розробник: Jenkins, Inc. https://www.jenkins.io/doc/
6	Grafana Розробник: Grafana, Inc. https://grafana.com/
7	Prometheus Розробник: Prometheus, Inc. https://prometheus.io/docs/introduction/overview/
8	MySql Розробник: MySQL HeatWave, Inc. https://dev.mysql.com/doc/
9	digitalocean Розробник: digitalocean, Inc. https://digitalocean.com/