МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

«До захисту допущений»

Зав. Кафедри комп'ютерних
наук та прикладної математики

д.т.н., професор Мартинюк П.М.

«____» _____ 20_ р.

КВАЛІФІКАЦІЙНА РОБОТА

РОЗРОБКА PIPELINE CI/CD З ВИКОРИСТАННЯМ ANSIBLE ТА AWS

Виконав: Харченко Володимир В'ячеславович

Студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

група КН-41

_____

підпис

Керівник: к.т.н. доцент Ярощак Сергій Вікторович

_____

підпис

Рівне-2023

Національний університет водного господарства та природокористування

(повне найменування вищого навчального закладу)

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

Кафедра **комп'ютерних наук та прикладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань **12 Інформаційні технології**

(шифр і назва)

Спеціальність **112 Комп'ютерні науки**

(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри

_____

«____»_____2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____

(прізвище, ім'я, по батькові)

1. Тема роботи *"Розробка pipeline ci/cd з використанням ansible та aws"*
керівник роботи *Ярощак Сергій Вікторович, к.е.н., доцент каф. пр. мат.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)
затверджені наказом по університету від "*19"квітня 2023* року С №

2. Термін подання роботи студентом *01.06.23*

3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *розробити веб-сайт для спортивного залу та налаштувати ci/cd pipeline. Здійснити програмну реалізацію відповідного продукту та протестувати його.*

4. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Відеозапис; мультимедійна презентація.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*доцент Ярощак С.В.*	*7.11.22*	*7.11.22*
Розділ 2	*доцент Ярощак С.В.*	*10.12.22*	*10.12.22*
Розділ 3	*доцент Ярощак С.В.*	*17.03.23*	*17.03.23*

6. Дата видачі завдання *01.10.23*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Вивчення літератури за обраною тематикою*	*07.11.22-10.12.22*	*виконав*
2	*Формулювання постановки задачі*	*11.12.22-25.12.22*	*виконав*
3	*Розробка інформаційної моделі задачі*	*10.01.23-14.01.23*	*виконав*
4	*Розробка алгоритму проектування задачі*	*15.01.23-06.02.23*	*виконав*
5	*Здійснення програмної реалізації*	*07.02.23-05.04.23*	*виконав*
6	*Проведення тестів системи*	*06.04.23-23.04.23*	*виконав*
7	*Загальні висновки до роботи*	*05.05.23-14.05.23*	*виконав*
8	*Вивчення питань з охорони праці*	*15.05.23-27.05.23*	*виконав*
9	*Підготовка звіту кваліфікаційної роботи*	*28.05.23-15.06.23*	*виконав*
10	*Підготовка мультимедійної презентації*	*16.06.23-18.06.23*	*виконав*
11	*Підготовка до виступу*	*19.06.23-21.06.23*	*виконав*

Студент _____

_____

(підпис) (прізвище та ініціали)

Керівник роботи _____

_____

ЗМІСТ

РЕФЕРАТ 6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1 ОГЛЯД CI/CD ТЕХНОЛОГІЙ 9
1.1 Роль devops та використання CI/CD в сучасній розробці 9
1.2 Найпопулярніші технології CI/CD pipeline 10
1.3 Огляд хмарних технологій 12
1.4 Інструменти контейнеризації та оркестрування 15
РОЗДІЛ 2 ЗАСОБИ РЕАЛІЗАЦІЇ ТА РОЗРОБКИ 23
2.1 Архітектура веб-сайту 23
2.2 Використання Spring Framework для створення серверної частини веб-сайту. 24
2.3 Використання React.js для створення клієнтської частини веб-сайту. 27
2. 4 Використання PostgreSQL в якості бази даних. 30
2. 5 Створення серверів для CI/CD на AWS EC2 31
РОЗДІЛ 3 ІНТЕРФЕЙС КОРИСТУВАЧА 40
3.1 Огляд інтерфейсу користувача 40
ВИСНОВКИ 45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 46

РЕФЕРАТ

Кваліфікаційна робота: 45 с., 32 малюнки, 8 джерел.

Мета роботи:

- Розробити веб-сайт спортивного залу, в якому буде описана інформація про зал.
- Реалізувати можливість покупки та продовження абонементу.
- Забезпечити базу з інформацією про правильне виконання вправ.
- Реалізувати безперервну роботу веб-сайту на хмарних серверах з можливістю додавати нові функції з миттєвою інтеграцією без зупинки сервера.

Методи розробки: клієнт-серверні технології, хмарні сервіси, інструменти безперервної інтеграції, інструменти огкестрації.

Результатом роботи є розроблений веб-сайт, яких складається з клієнтської та серверної частини, та запущений на хмарному сервері з налаштованою безперервною інтеграцією

Ключові слова: веб-сайт, сервер, контейнеризація, оркестрація

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DRY	– Do not Repeat Yourself
MVC	– Model-View-Controller
ORM	– Object-Relational Mapping
REST API	– Representational State Transfer
JSON	– JavaScript Object Notation
CI/CD	– continuous integration and continuous delivery
AWS	– Amazon Web Services

ВСТУП

В наш час, коли люди ведуть малорухливий спосіб життя, існує ймовірність отримати проблеми зі здоров’ям в ранньому віці. Щоб уникнути даної участі хорошим рішенням буде почати відвідувати спортивний зал. Також для того щоб спростити процес вибору та взаємодії з залом потрібен веб-сайт який надає цю можливість. Також плюсом власників залу буде те, що багато людей шукають спортзал або фітнес-клуб через Інтернет. Сайт буде першим контактом, тому важливо створити позитивне перше враження. Можна візуально показати унікальність, представити фото та відео занять, а також розмістити відгуки задоволених клієнтів. Сайт дозволить детально описати всі послуги, які доступні.

Крім того, щоб веб-сайт був зручним, необхідно розробити систему, яка буде дозволяти клієнтам швидкий і постійний доступ, також необхідно зробити можливість додавання нового функціоналу. Ефективним рішенням цієї проблеми буде налаштування CI/CD pipeline. Це дозволить забезпечити постійну роботу сервера та покращити безпеку. Крім того покращить мобільність веб-сайту, так як CI/CD pipeline надає можливість один раз налаштувавши роботу інфраструктури запускати її на різних серверах. Також розробникам відкриється можливість для масштабування навантаження на веб-сайт або інтегрування корисних для бізнесу інструментів таких, як системи моніторингу даних.

Також потрібно відповідально обрати сервери на який буде працювати система. Хорошим вибором буде орендувати сервера Amazon. Так як Amazon надає готові інструменти для роботи та бере всю важку роботу на себе. Крім того користувачі платять лише за використаний трафік, а не фіксовану суму, що дозволить заощадити кошти.

У результаті виконання роботи прогнозується розробка веб-сайту, який буде складатись із серверної і клієнтської частини також його інтеграція на хмарні сервери.

РОЗДІЛ 1 ОГЛЯД CI/CD ТЕХНОЛОГІЙ

1.1 Роль devops та використання CI/CD в сучасній розробці

Сучасний світ програмування постійно змінюється, а розробники прагнуть знайти нові шляхи для поліпшення ефективності своєї роботи. Одним з найважливіших досягнень в цьому напрямку є технологія Continuous Integration/Continuous Deployment, відома під скороченням CI/CD, якою займаються devops інженери.

CI/CD - це відповідь на виклики, з якими зіштовхуються розробники. Ця технологія дозволяє забезпечити безперервну інтеграцію та автоматичне розгортання програмного забезпечення. Що це означає? Якщо подумати про традиційну модель розробки програмного забезпечення то з’ясується, що розробники працюють над своїми функціями окремо, потім об'єднують свій код і сподіваються, що він не викличе конфліктів або помилок. Це призводить до затримок, невпевненості та незадоволення як з боку розробників, так і з боку користувачів. Але технологія CI/CD змінює цю ситуацію. Вона забезпечує неперервну інтеграцію та означає, що розробники регулярно об'єднують свій код в спільному репозиторії, де він автоматично перевіряється на конфлікти та помилки компіляції. Це дозволяє виявляти проблеми значно швидше і вирішувати їх на ранніх етапах розробки. Крім того, автоматизоване тестування допомагає виявити функціональні та помилки в безпеці, забезпечуючи високу якість продукту.

Але CI/CD - це не просто про інтеграцію. Вона також надає можливість безперервного розгортання і означає, що коли код пройшов всі перевірки та тести, він автоматично готується для розгортання в продакшн середовище. Це значить, що розробники можуть швидко та безпечно випускати нові функції та виправлення, надаючи користувачам свіжі та покращені версії програмного забезпечення.

CI/CD - це справжня революція в розробці програмного забезпечення. Вона дозволяє розробникам прискорити процес розробки, зменшити ризик

виникнення помилок та поліпшити якість продукту. Крім того, вона створює фундамент для швидкого реагування на зміни на ринку та потреби користувачів.

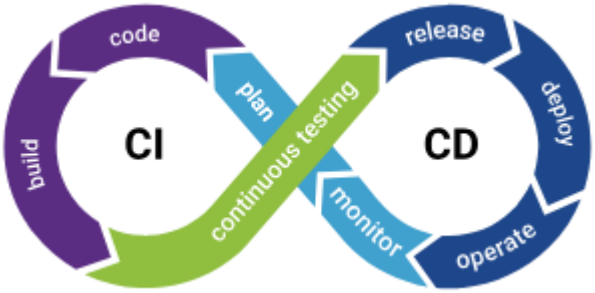


Рис. 1.1. Принцип роботи CI/CD

1.2 Найпопулярніші технології CI/CD pipeline

На сьогоднішній день існує багато популярних технологій для реалізації CI/CD процесу. Ось кілька з найбільш популярних та широко використовуваних:

а) Jenkins

б) GitLab CI/CD

в) Travis CI

г) CircleCI

Нижче детальніше розглянуті деякі з них.

У світі розробки програмного забезпечення, коли слова "швидкість", "ефективність" та "надійність" стали необхідними компаньйонами, з'явилося велике число інструментів, що обіцяють полегшити життя розробників. Але серед них особливе місце займає Jenkins - справжній володар CI/CD світу. Тому що Jenkins - це відкрита система автоматизації розгортання, яка завоювала серця розробників по всьому світу. Вона відкриває двері в безмежний світ безперервної інтеграції та автоматичного розгортання. За допомогою Jenkins розробники можуть забезпечити автоматичну перевірку, збірку, тестування та розгортання свого коду на регулярній основі. Одна з найбільших переваг Jenkins полягає в його гнучкості та розширюваності. Він надає безліч плагінів, які можуть бути легко встановлені та налаштовані для

відповідності потребам розробників. Це дозволяє використовувати Jenkins в різних типах проектів та з різними мовами програмування. Jenkins пропонує інтуїтивний інтерфейс користувача, який спрощує налаштування та керування процесом CI/CD. Розробники можуть легко налаштовувати послідовність дій, встановлювати умови тестування та автоматизувати процес розгортання. Jenkins також надає детальні звіти та повідомлення про стан проекту, що допомагає розробникам відстежувати прогрес і виявляти проблеми. Незалежно від розміру проекту чи команди розробників, Jenkins може стати надійним партнером для автоматизації.

Ще однією із основних CI/CD технологій є GitLab CI/CD. Одна з найвизначніших особливостей GitLab CI/CD - це його повна інтеграція з GitLab, що створює неперевершене середовище для керування версіями, спільної роботи та автоматизації розробки. Це означає, що розробники можуть проводити всі етапи розробки - від контролю версій коду до проходження тестів та розгортання - у єдиній платформі. Завдяки інтеграції GitLab CI/CD з GitLab, розробники можуть легко оглядати та аналізувати зміни в коді, стежити за прогресом розробки та спілкуватись з колегами без необхідності переходу між різними інструментами. Це покращує комунікацію та співпрацю між розробниками, що є важливим аспектом успішної розробки. Іншою вагомою особливістю GitLab CI/CD є його простота налаштування та використання. Він надає зрозумілий та легкий у використанні синтаксис для конфігурації CI/CD процесу, що дозволяє розробникам швидко налаштувати його під свої потреби. Більш того, GitLab CI/CD підтримує багато різних можливостей, таких як паралельне виконання тестів, контейнеризація та інтеграція з іншими сервісами, що дозволяє максимально розширити його функціональність.

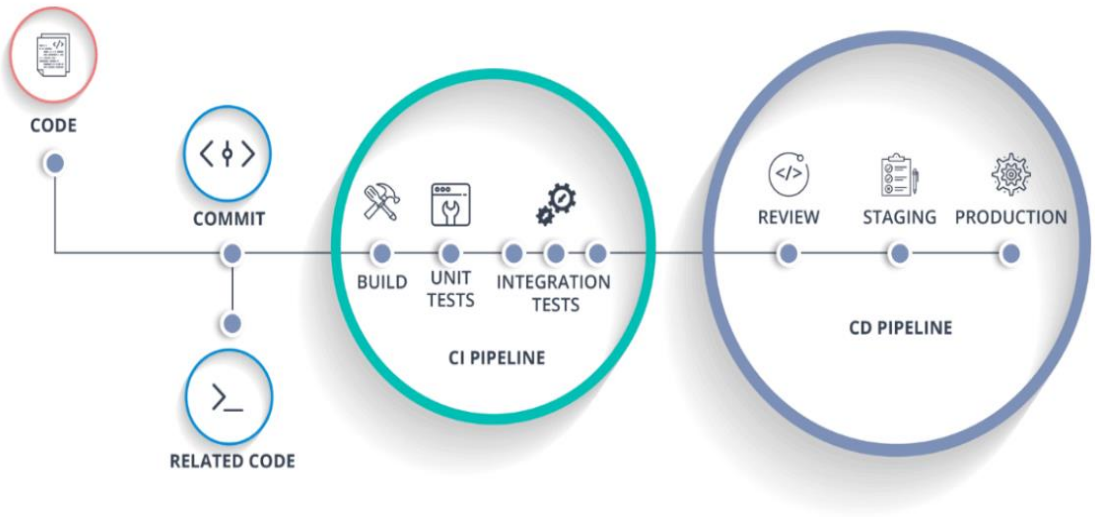


Рис. 1. 2. Детальна схема CI/CD pipeline

1.3 Огляд хмарних технологій

Хмарні технології в розробці програмного забезпечення - це незаперечна перевага для сучасних розробників. Вони змінюють підхід до створення, тестування та розгортання додатків, роблячи цей процес більш ефективним і доступним. Розглянемо найпопулярніші хмарні технології в розробці, які зараз перебувають у фокусі уваги.

Перша і найвідоміша з них - Amazon Web Services (AWS). AWS надає широкий спектр хмарних послуг, включаючи віртуальні машини, зберігання даних, бази даних та інструменти для масштабування додатків. Завдяки AWS розробники можуть швидко створювати інфраструктуру для своїх проектів, масштабувати ресурси за необхідності та ефективно керувати розгорнутими додатками.

Однією з найбільш привабливих особливостей AWS є його гнучкість. Платформа пропонує різноманітні послуги, які можна комбінувати та налаштовувати відповідно до потреб кожного користувача. Від віртуальних машин і зберігання даних до штучного інтелекту і Інтернету речей, AWS надає цілу гаму сервісів, які допомагають розробникам реалізувати свої ідеї і проекти. AWS заслуговує на визнання і своєю потужною інфраструктурою.

Компанія має розподілену мережу дата-центрів по всьому світу, які забезпечують високу доступність і швидкість обробки даних. Це означає, що користувачі AWS можуть розгортати свої додатки та послуги на серверах, розташованих у різних куточках світу, забезпечуючи надійність та швидкість роботи.

Однією з ключових переваг AWS є його масштабованість. AWS дозволяє розробникам легко масштабувати свої проекти, збільшуючи або зменшуючи ресурси в залежності від потреб. Це необхідно для компаній, що стикаються зі зростанням обсягів даних та навантаженням. Завдяки AWS користувачі можуть ефективно використовувати ресурси, що забезпечує оптимальну продуктивність і економію коштів. Компанія постійно розширює свій сервісний набір та пропонує нові технології, які стають ключовими факторами в індустрії. Наприклад, AWS Machine Learning дозволяє розробникам використовувати потужні алгоритми машинного навчання без значних зусиль. Крім того, AWS Elastic Beanstalk спрощує процес розгортання додатків, дозволяючи розробникам зосередитися на кодуванні, а не на інфраструктурних питаннях.

Рис. 1. 3. Найпопулярніші AWS сервіси

Важливо звернути увагу на один із AWS сервісів, який використовується в створенні CI/CD Pipeline, а саме Amazon Elastic Compute Cloud (EC2). Це один з найбільш популярних сервісів, наданих Amazon Web Services (AWS). EC2 відкриває широкі можливості для розробників, дозволяючи їм легко створювати і керувати віртуальними серверами в хмарному середовищі. EC2 надає гнучкість і масштабованість, які необхідні для розробки та розгортання додатків. Розробники можуть швидко створювати і налаштовувати віртуальні машини залежно від своїх потреб. EC2 пропонує широкий вибір типів інстансів, розрахованих на різні сценарії використання, включаючи машини загального призначення, оптимізовані під обчислення, пам'ять або зберігання. EC2 дозволяє розробникам вибирати певні параметри віртуальних машин, такі як розмір процесора, обсяг оперативної пам'яті, потужність сховища тощо. Це дає можливість налаштувати інфраструктуру для відповідності конкретним потребам проекту. Крім того, розробники можуть масштабувати свої ресурси вгору або вниз залежно від навантаження, забезпечуючи ефективне використання ресурсів і економію коштів. EC2 також забезпечує високий рівень надійності і безпеки. AWS забезпечує резервне копіювання та реплікацію даних, а також захист від відмов, що забезпечує незавершену роботу навіть у випадку помилки апаратного забезпечення чи інших непередбачуваних ситуацій.

Другою популярною хмарною платформою є Microsoft Azure. Azure пропонує широкий спектр сервісів, включаючи обчислення, зберігання, бази даних. Розробники можуть використовувати Azure для створення сучасних додатків, використовуючи масштабовану та безпечну інфраструктуру Microsoft. Однією з найважливіших особливостей Azure є її інтеграція з екосистемою Microsoft. Багато організацій уже використовують продукти та технології Microsoft, такі як Windows Server, SQL Server та Active Directory. Azure надає глибоку інтеграцію з цими продуктами, що дозволяє легко

розширювати існуючі інфраструктури та переносити робочі навантаження у хмарне середовище.

Третьою хмарною технологією, що заслуговує на увагу, є Google Cloud Platform (GCP). GCP надає інструменти та сервіси для розробки, тестування та розгортання додатків. Завдяки надійній та масштабованій інфраструктурі Google, розробники можуть легко створювати розподілені системи, використовуючи потужні сервіси обчислення та зберігання даних.

Крім цього, не можна забувати про платформи розробки хмарних додатків, такі як Heroku, Firebase та IBM Cloud. Ці платформи надають розробникам готове середовище для створення, тестування та розгортання додатків без необхідності налаштування інфраструктури.

Завдяки хмарним технологіям в розробці, розробники отримують багато переваг. Вони можуть прискорити розгортання додатків, зменшити витрати на інфраструктуру та мати можливість масштабувати свої проекти залежно від потреб. Хмарні технології стають невід'ємною частиною розробки програмного забезпечення, відкриваючи нові можливості та сприяючи прискоренню інновацій.

1.4 Інструменти контейнеризації та оркестрування

У сфері DevOps існує безліч інструментів контейнеризації та оркестрування, які допомагають забезпечити швидку розробку, доставку та масштабування додатків. Контейнеризація в розробці має декілька важливих переваг, які допомагають розробникам ефективно працювати та спрощують процеси розробки, тестування та розгортання додатків. Одна з них це переносимість. Це контейнери забезпечують переносиме середовище для додатків. Вони включають у себе всі необхідні залежності та налаштування, що робить їх переносними між різними операційними системами та середовищами. Розробники можуть розробляти та тестувати додатки на своєму робочому середовищі, а потім легко перенести їх до інших середовищ,

таких як тестові або виробничі сервери, зберігаючи при цьому стабільність та відтворюваність.

Також контейнеризація забезпечує ізольованість, що забезпечує високий рівень ізоляції між додатками та їх середовищами. Кожен контейнер має власне віртуальне середовище, що дозволяє уникнути конфліктів залежностей та можливих проблем, які можуть виникнути при використанні спільного сервера або віртуальної машини. Це дозволяє розробникам створювати інкапсульовані додатки, які працюють незалежно один від одного.

Третьою перевагою є швидкість розгортання, що дозволяє швидко розгортати додатки та їх залежності. Контейнери можуть бути запущені та зупинені швидко, що прискорює процес розгортання та розвитку додатків. Розробники можуть швидко перевірити зміни, внесені до додатків, без необхідності повного перекомпілювання або перезавантаження.

Масштабованість, що гарантується контейнеризацією дозволяютє легко масштабувати додатки, як горизонтально (запуск декількох копій контейнера) так і вертикально (зміна розміру контейнера). Це дозволяє розробникам забезпечити високу доступність та продуктивність своїх додатків, адаптуючи їх до зростаючих навантажень.

Оркестрація в розробці є важливим процесом, який допомагає управляти та координувати різні компоненти, сервіси та ресурси в розподіленому середовищі і полегшує розгортання та масштабування. Оркестрація забезпечує автоматичне розподілення та керування ресурсами, що дозволяє ефективно використовувати наявні обчислювальні потужності та забезпечувати високу доступність додатків.

Також оркестрація дозволяє розподілити навантаження між різними екземплярами додатків або сервісів. Вона автоматично вирішує, на якому сервері або контейнері краще запустити додаток, забезпечуючи оптимальне

використання ресурсів та забезпечуючи стабільність та продуктивність системи.

Оркестрація допомагає виявляти та відновлювати помилки автоматично. Якщо один з компонентів додатку несправний або зупинився, оркестратор може автоматично перезапустити його або замінити на новий екземпляр, забезпечуючи неперервну роботу системи.

Оркестраційні інструменти надають можливості для моніторингу та логування додатків та їх компонентів. Це дозволяє розробникам відстежувати стан додатків, виявляти проблеми та аналізувати вироблені дані для покращення продуктивності та надійності системи.

Декілька найпопулярніших інструментів в цій області включають Docker, Kubernetes та Ansible.

Docker - це відкрите програмне забезпечення, яке надає можливість контейнеризації додатків. Він є одним з найпопулярніших інструментів у світі контейнеризації та є основою багатьох інших інструментів контейнеризації й оркестрування.

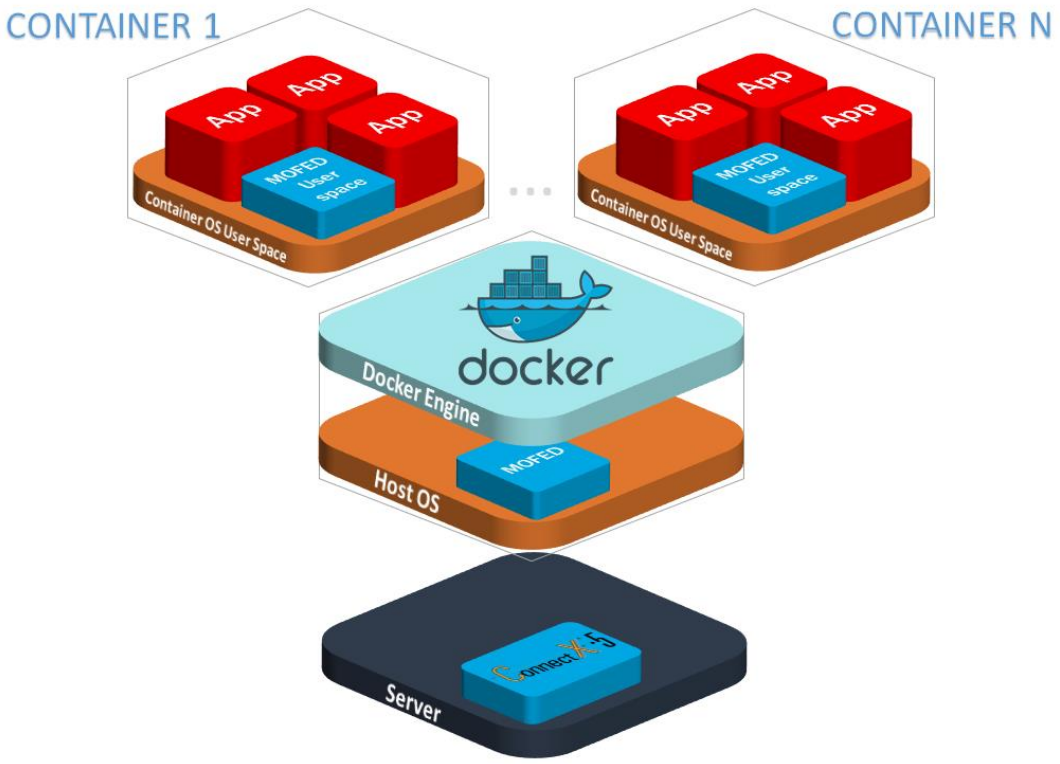


Рис. 1. 4. Принцип роботи docker

Основна ідея Docker полягає в упакуванні додатків та їх залежностей в стандартизовані контейнери. Контейнер є ізольованим середовищем, в якому можна запускати додатки безпосередньо на операційній системі хоста, незалежно від конфігурації цієї операційної системи. Це дозволяє розробникам розподіляти додатки разом з усіма необхідними залежностями, такими як бібліотеки, фреймворки та інструменти, і переносити їх між різними середовищами без проблем.

Також потрібно звернути увагу на такий сервіс як Docker Hub. Docker Hub - це публічний реєстр образів контейнерів, що надає сервіси для зберігання, пошуку та розповсюдження контейнерів Docker. Це найбільший та найпопулярніший реєстр контейнерів, що забезпечує широкий вибір готових образів, які можна використовувати для розгортання додатків у контейнерах.

Docker Hub надає можливість завантажувати та зберігати образи контейнерів у хмарі. Розробники можуть завантажувати свої власні образи

або використовувати публічно доступні образи, створені спільнотою Docker. Це дозволяє ефективно управляти та контролювати версії образів, що використовуються в розробці та розгортанні.

Також Docker Hub надає потужний пошуковий механізм, який дозволяє знайти багато готових образів для різних сервісів та програм. Розробники можуть шукати образи за ключовими словами, категоріями або власниками, що спрощує процес пошуку та вибору відповідного образу для своїх потреб. Docker Hub дозволяє розробникам створювати як публічні, так і приватні репозиторії для зберігання своїх образів. Публічні репозиторії доступні для загального користування, тоді як приватні репозиторії можуть бути обмежені доступом лише до обраних користувачів або команди розробників.

Docker Hub дозволяє налаштовувати автоматичне розгортання образів контейнерів з репозиторію під час процесу CI/CD. Це дає змогу автоматично розгортати нові версії додатків у контейнерах після проходження тестування та перевірки коду. Docker Hub підтримує широку спільноту розробників, яка активно ділиться готовими образами, документацією, порадами та підтримкою. Розробники можуть спілкуватися, обговорювати проблеми та знаходити відповіді на свої запитання у спільноті Docker.

Ansible відіграє важливу роль у спрощенні та автоматизації процесів розгортання, конфігурації та управління інфраструктурою. Завдяки своїм особливостям та перевагам, Ansible стає незамінним інструментом для команд розробників та операторів систем. Однією з найбільших переваг Ansible є його декларативний підхід. Замість описування послідовності кроків для досягнення певного стану системи, Ansible використовує конфігураційні файли, відомі як плейбуки, в яких описується бажаний стан системи. Ansible автоматично конвертує поточний стан системи до зазначеного стану, що дозволяє зосередитись на результаті, а не на кроках досягнення його. Це забезпечує простоту та легкість у використанні Ansible.

Також однією із особливостей Ansible є його агент-менеджерська архітектура. Ansible не потребує встановлення окремих агентів на вузлах, які підлягають керуванню. Він працює за протоколом SSH і використовує вже наявні засоби інфраструктури для взаємодії з вузлами. Це спрощує процес встановлення та конфігурації та дозволяє легко масштабувати інфраструктуру. Ansible також надає широкий спектр модулів та плагінів, які дозволяють автоматизувати багато типових задач. Він підтримує багато провайдерів хмарних послуг, контейнеризацію, мережеву автоматизацію та багато інших сценаріїв. Це робить Ansible універсальним інструментом, який можна використовувати для автоматизації різних аспектів розробки та управління інфраструктурою.

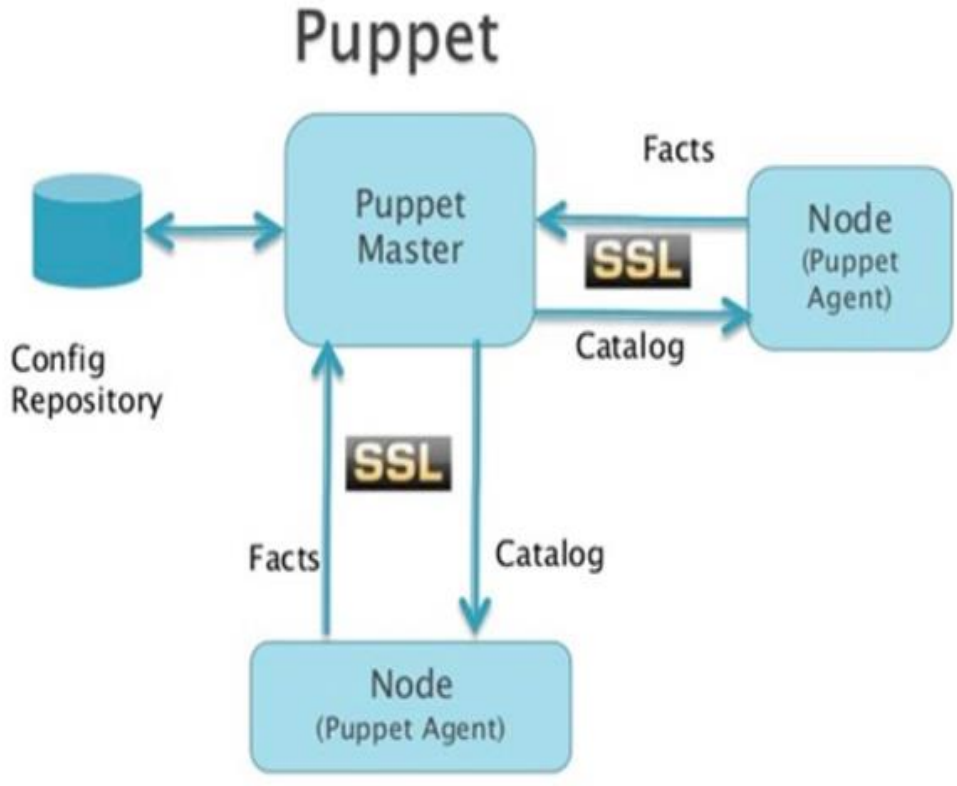


Рис. 1. 5. Схема роботи Ansible

Kubernetes, часто називаний "K8s", є одним з найпопулярніших інструментів оркестрування контейнерів у світі розробки та розгортання програмного забезпечення. Цей відкритий інструмент, розроблений компанією Google, вирішує складні завдання керування та масштабування

контейнеризованих додатків, роблячи процеси розробки та управління інфраструктурою більш ефективними.

Одна з найбільших переваг Kubernetes полягає в його здатності автоматизувати та управляти розгорнутими контейнерами. Замість ручного керування кожним контейнером окремо, Kubernetes надає зручний інтерфейс та набір інструментів для декларативного опису бажаного стану системи. Він автоматично відстежує стан контейнерів та забезпечує, щоб вони завжди працювали за встановленими правилами, незалежно від кількості контейнерів або вузлів. Крім того, Kubernetes надає потужні механізми масштабування. Він дозволяє горизонтально масштабувати додатки шляхом автоматичного додавання або видалення контейнерів залежно від навантаження. Це дозволяє забезпечити високу доступність додатків та ефективне використання ресурсів інфраструктури.

Ще однією важливою особливістю Kubernetes є його здатність до самовідновлення. Він постійно моніторить стан контейнерів та вузлів і, у разі виявлення проблеми, автоматично запускає нові контейнери або перенаправляє трафік, щоб забезпечити неперервну роботу додатків. Також Kubernetes є портативним та розширюваним інструментом. Він може працювати з будь-яким провайдером хмарних послуг або локальною інфраструктурою, що дозволяє розгортати додатки на різних середовищах без зміни коду. Крім того, Kubernetes має широкий набір розширень та плагінів, що дозволяють налаштувати його під конкретні потреби проекту.

Ці інструменти контейнеризації та оркестрування грають ключову роль у розробці та управлінні додатками в середовищі DevOps. Вони допомагають забезпечити швидкість, надійність та масштабованість в процесах розробки, тестування та впровадження додатків. Вибір конкретного інструменту залежить від ваших потреб та вимог вашого проекту, але в кінцевому підсумку вони допоможуть вам ефективно працювати в середовищі DevOps.

РОЗДІЛ 2 ЗАСОБИ РЕАЛІЗАЦІЇ ТА РОЗРОБКИ

2.1 Архітектура веб-сайту

Архітектура веб-сайту складається з серверної частини реалізованої на Spring Boot, клієнтської частини реалізованої на ReactJs та бази даних PostgreSQL.

Серверна частина використовує MVC архітектуру. Тобто в якості Model виступають так звані “Entity” що описують сутність бази даних.

```
@Entity
public class Tariff {
    no usages
    @Id
    @GeneratedValue
    private Integer id;
    no usages
    private String title;
    no usages
    private String description;
    no usages
    private BigDecimal price;
    no usages
    private Period duration;
}
```

Рис. 2. 1. Приклад Entity в Spring Boot

Spring Boot взаємодіє з ORM PostgreSQL через фреймворки hibernate та Spring Data JPA.

Controller маніпулює даними з Model та передає їх у потрібній розробнику формі до View у форматі JSON.


```
{
    "role": "USER",
    "access_token": "eyJhbGciOiJIUzI1NiJ9.
        eyJzdWIiOiJraGFyY2hlbmtvX2FrMTlAbnV3bS5lZHUudWEiLCJpYXQiOjE2ODc3MTgyMTEsImV
        4cCI6MTY4NzgwNDYxMX0.ngJRAOH_UGYGA9HAx1OQPFhg7qTwiGFtgtzYrXnQi0s",
    "refresh_token": "eyJhbGciOiJIUzI1NiJ9.
        eyJzdWIiOiJraGFyY2hlbmtvX2FrMTlAbnV3bS5lZHUudWEiLCJpYXQiOjE2ODc3MTgyMTEsImV
        4cCI6MTY4ODMyMzAxMX0.FFsmVe88pL4TVmfbL3Yvqc6XbMA-fD-SjWzHg61lyWk"
}
```

Рис. 2. 2. Приклад отриманих даних при авторизації

На стороні клієнта дані з сервера доставляються бібліотекою axios, що дозволяє виконувати HTTP запити після чого відображаються користувачеві.

Зв'язавши Spring Boot і React, побудувана повноцінна архітектуру REST.

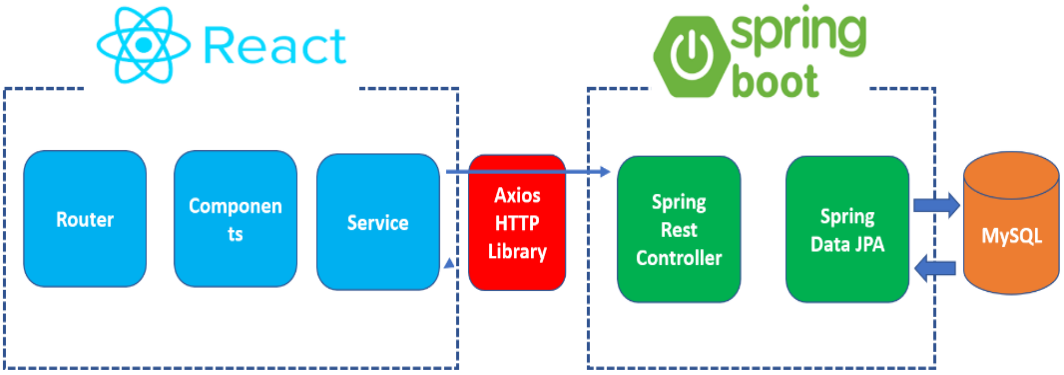


Рис. 2. 3. Схема архітектури веб-сайту

2.2 Використання Spring Framework для створення серверної частини веб-сайту.

Серверна частина додатку була створена з використання технології Spring Framework. Spring Framework відкриває широкі можливості для створення потужних, гнучких та ефективних програмних рішень, особливо в галузі розробки корпоративних додатків.

Spring Framework є простором розробки програмного забезпечення, який надає комплексний підхід до створення Java-додатків. Він базується на принципах інверсії управління (Inversion of Control, IoC) та відображення об'єктів (Dependency Injection, DI), що дозволяє полегшити розробку, покращити тестованість і забезпечити легку інтеграцію з іншими фреймворками.

Основні функції Spring Framework:

a) Inversion of Control (IoC)

b) Dependency Injection (DI)

c) Аспектно-орієнтоване програмування (Aspect-Oriented Programming, AOP)

d) Модульність

Компоненти Spring Framework використанні в програмі.

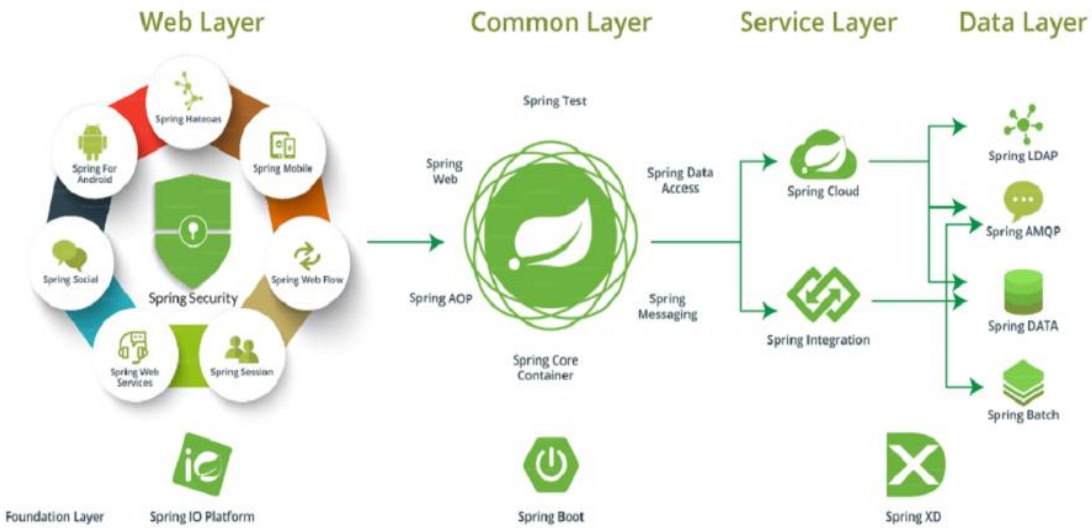


Рис. 2. 4. Фреймворки Spring

Spring Core: Це основний модуль Spring Framework, який забезпечує IoC-контейнер та DI. Він містить класи та інтерфейси, такі як ApplicationContext та BeanFactory, які дозволяють створювати та керувати компонентами додатка.

Spring MVC: Цей модуль надає функціональність для розробки веб-додатків за архітектурою MVC (Model-View-Controller). Він забезпечує обробку HTTP-запитів, керування потоком даних та відображення на веб-сторінки.

Рис. 2. 5. Список endpoints

Spring Data: Цей модуль дозволяє спростити роботу з реляційними та нереляційними базами даних. Він надає абстракції для взаємодії з різними джерелами даних і спрощує розробку доступу до даних.

Spring Security: Цей модуль забезпечує функціонал для реалізації механізмів аутентифікації та авторизації в додатках. Він дозволяє захистити ресурси додатка і забезпечити безпеку даних.

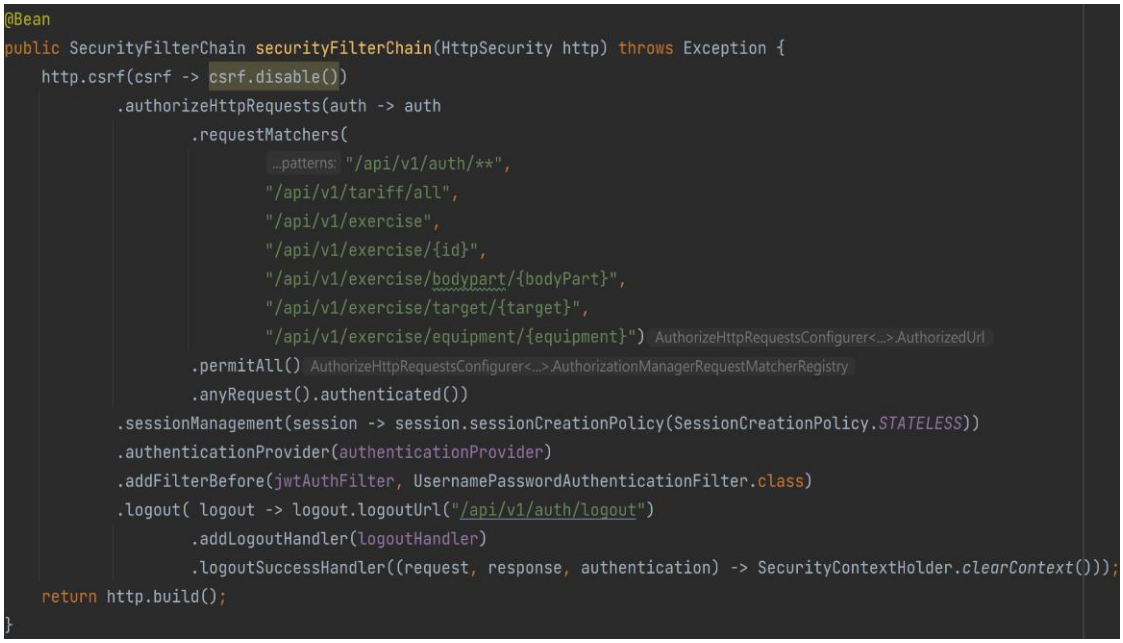


Рис. 2. 6. Метод конфігурації доступу

2.3 Використання React.js для створення клієнтської частини веб-сайту.

React.js є потужним інструментом, який дозволяє створювати модульні, ефективні та масштабовані користувацькі інтерфейси. React - це бібліотека JavaScript, розроблена компанією Facebook. Вона використовується для побудови користувацьких інтерфейсів з використанням компонентного підходу. Замість традиційного підходу до маніпулювання DOM-елементами безпосередньо, React використовує віртуальний DOM та ефективний механізм оновлення компонентів.

Основними концепціями React.js являються компоненти, віртуальний DOM, односторонній потік даних.

Компоненти є будівельними блоками користувацького інтерфейсу і можуть бути використані для побудови складних інтерфейсів. Кожен компонент має власний стан (state) і може бути відображений на екрані. Компоненти можуть бути вкладені один в одного, що сприяє створенню модульного коду.

Віртуальний DOM - це представлення реального DOM у вигляді JavaScript-об'єкту. React порівнює старе та нове представлення та здійснює

мінімальні зміни в реальному DOM для оновлення інтерфейсу, що робить його швидшим та ефективнішим.

У React.js використовується принцип одностороннього потоку даних. Це означає, що дані пересуваються від батьківських компонентів до дочірніх компонентів. Компоненти не можуть безпосередньо змінювати стан інших компонентів, що сприяє більш прогнозованому та легкому управлінню станом додатку.

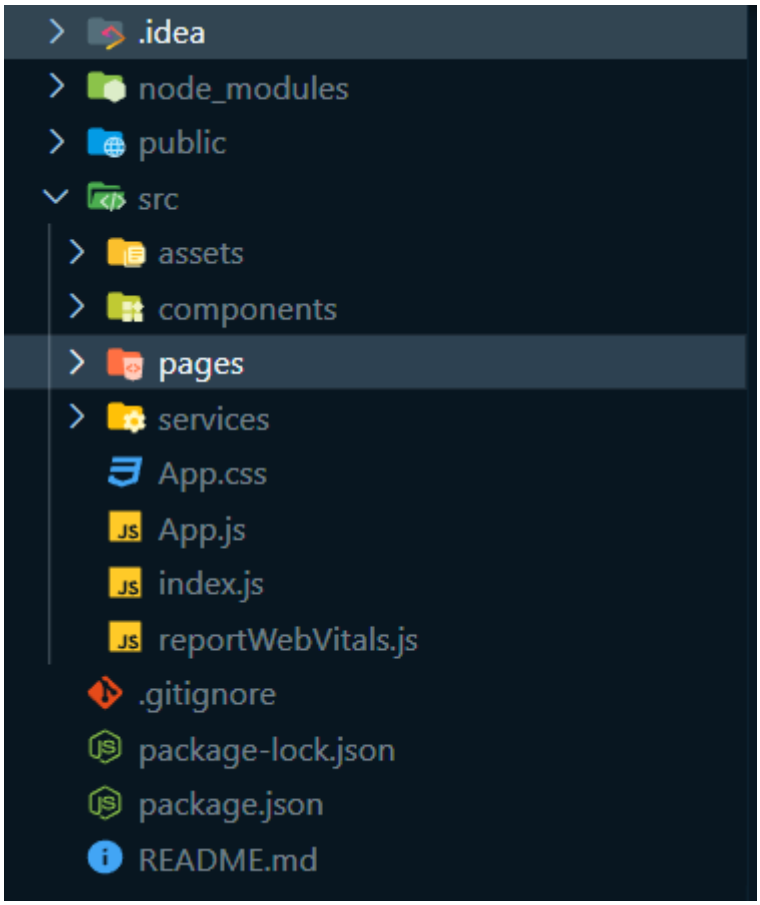


Рис. 2. 7. Коренева структура клієнтської частини

React.js має велику екосистему розширень та сторонніх бібліотек. Існують багато рішень для стилізації (наприклад, Material UI, який використовується в програмі) та роутингу (наприклад, React Router), що допомагають побудувати потужні та функціональні додатки.

```
"@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@fortawesome/free-solid-svg-icons": "
^6.4.0",
    "@fortawesome/react-fontawesome": "^0.2.0
",
    "@mui/icons-material": "^5.11.16",
    "@mui/material": "^5.13.4",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "fontawesome": "^5.6.3",
    "fortawesome": "^0.0.1-security",
    "jwt-decode": "^3.1.2",
    "localforage": "^1.10.0",
    "match-sorter": "^6.3.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-horizontal-scrolling-menu": "
^2.7.1",
    "react-loader-spinner": "^5.3.4",
    "react-router-dom": "^6.11.1",
    "react-scripts": "5.0.1",
    "sort-by": "^1.2.0",
    "web-vitals": "^2.1.4"
```


Рис. 2. 8. Використанні бібліотеки в клієнтській частині

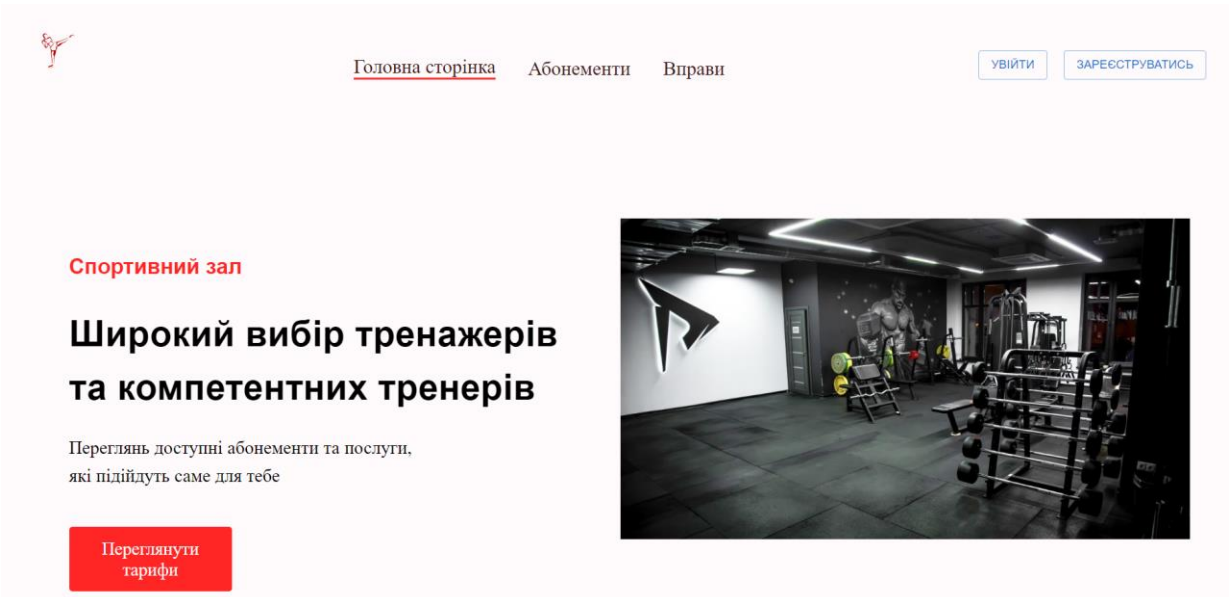


Рис. 2. 9. Фрагмент головної сторінки веб-сайту

2. 4 Використання PostgreSQL в якості бази даних.

PostgreSQL є однією з найпопулярніших відкритих СУБД та відома своєю надійністю, масштабованістю та розширюваністю.

PostgreSQL - це об'єктно-реляційна система управління базами даних, розроблена з урахуванням стандартів SQL та ACID (Atomicity, Consistency, Isolation, Durability). Вона пропонує багато передових функцій та можливостей, що робить її вибором для різних видів додатків, від невеликих веб-сайтів до великих корпоративних систем.

PostgreSQL підтримує розширення та сторонні модулі, що дозволяє розширювати його функціональність. Вона дозволяє створювати власні типи даних, функції, операції та індекси для відповідності з унікальними потребам.

Ще PostgreSQL відома своєю надійністю та міцністю. Вона гарантує виконання ACID-властивостей, що забезпечує консистентність, ізольованість та довіреність даних. PostgreSQL також має потужні механізми відновлення, що дозволяють відновлювати дані після випадку збою та підтримує широкий спектр функцій SQL, включаючи підзапити, вьюшки, тригери, хранимі процедури та багато іншого. Вона також надає підтримку складних операцій з об'єктами, такими як географічні дані, JSON, XML тощо.

PostgreSQL може масштабуватися від невеликих однокористувацьких систем до великих корпоративних розподілених середовищ. Вона використовує розподілені механізми для обробки завдань та підтримує реплікацію для забезпечення високої доступності та надійності.

Найчастіше її використовують у веб розробці для зберігання даних веб-додатків. Вона має підтримку для багатьох популярних веб-фреймворків та інструментів розробки. Також завдяки своїм продвинутим функціям та підтримці аналітичних операцій, PostgreSQL є популярним вибором для систем бізнес-інтелекту та аналітики даних.

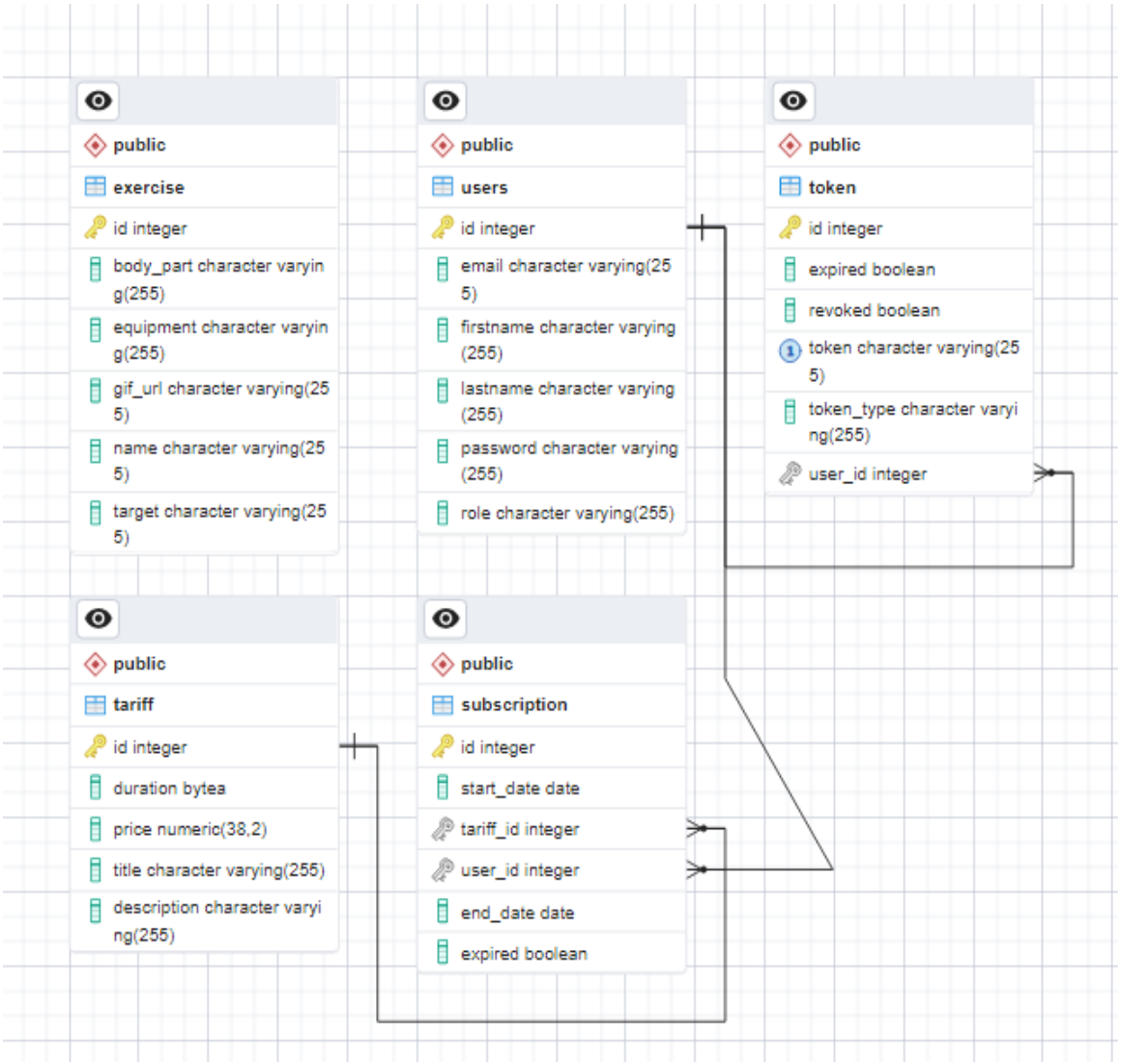


Рис. 2. 10. Схема бази даних веб-сайту

База даних веб-сайту зберігається на сайті nean.tech.

2. 5 Створення серверів для CI/CD на AWS EC2

EC2 сервери надають безліч можливостей, які роблять їх відмінним вибором для різноманітних сценаріїв використання. EC2 дозволяє вибирати конфігурацію сервера залежно від потреб. Можна встановлювати тип процесора, кількість ядер, обсяг пам'яті та сховища, мережеві параметри та інші характеристики. Це дозволяє налаштувати сервер точно під потрібні вимоги. EC2 надає можливість розподілити сервери на різні регіони та доступні зони. Це забезпечує високу доступність та надійність в разі виникнення неполадок або аварійних ситуацій. Також EC2 забезпечує різні

механізми для захисту ваших серверів та даних. Ви можете налаштувати правила безпеки, використовувати шифрування для зберігання даних та встановлювати контроль доступу до ресурсів.

Для налаштування CI/CD необхідно створити 4 хмарних сервера. Для серверної частини, для клієнтської частини, для Jenkins та для Ansible.

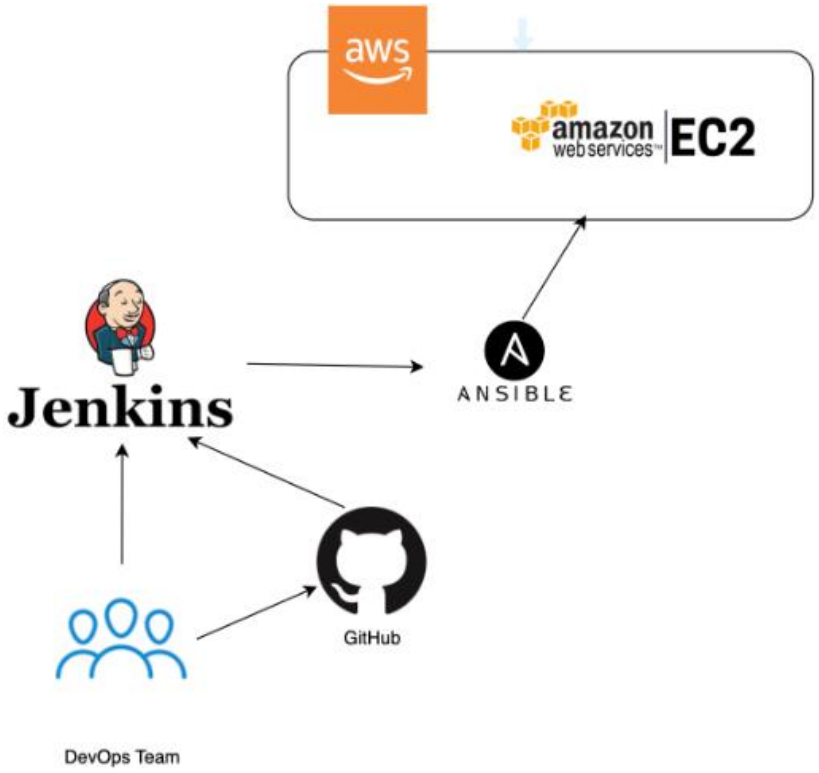


Рис. 2. 11. Архітектура ec2 серверів

Для кожного з обрана OS Amazon Linux та тип сервера t3.micro.

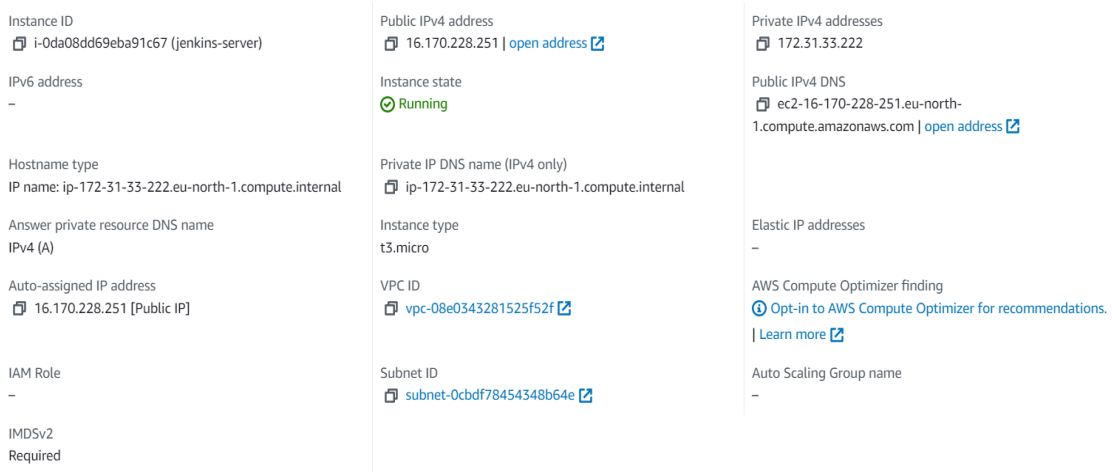
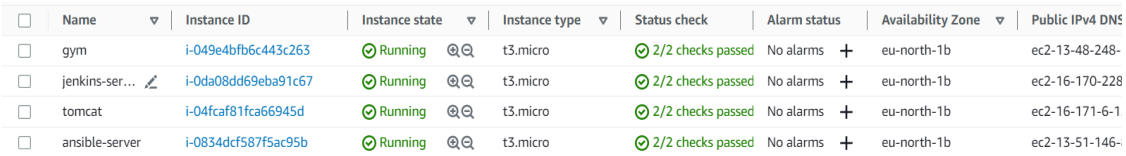


Рис. 2. 12. Інформація про Jenkins сервер



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
gym	i-049e4bfb6c443c263	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-13-48-248-
jenkins-ser...	i-0da08dd69eba91c67	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-16-170-228
tomcat	i-04fcaf81fca66945d	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-16-171-6-1
ansible-server	i-0834dcf587f5ac95b	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-13-51-146-

Рис. 2. 13. Dashboard із створеними серверами

В першу чергу був налаштований Jenkins сервер. Щоб мати до нього доступ необхідно підключитись до нього з допомогою терміналу на ec2 dashboard або локально з допомогою ssh ключа. В даному випадку здійснювалось підключення із локальної OS Ubuntu.

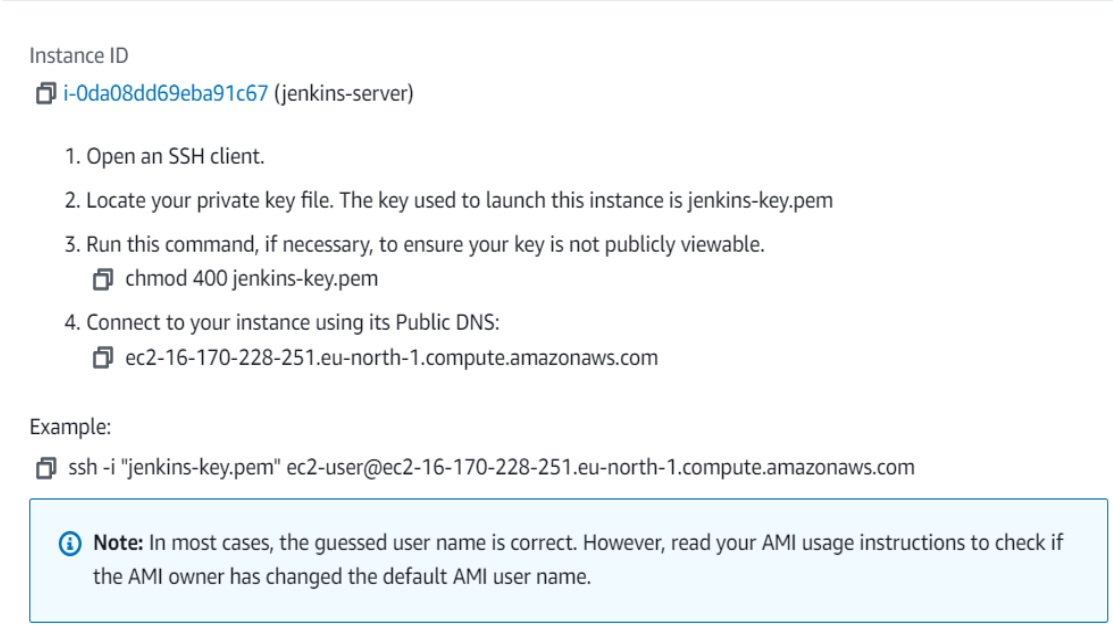


Рис. 2. 14. Інструкція підключення до ec2 сервера через ssh

Після підключення було виконане встановлення open-jdk-17 командою

sudo amazon-linux-extras install java-openjdk17 -y

так, як Jenkins написаний на мові програмування java. Команди встановлення та запуску Jenkins:

sudo yum install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start jenkins

Для перевірки коректного встановлення перейти на адресу сервера з портом 8080.

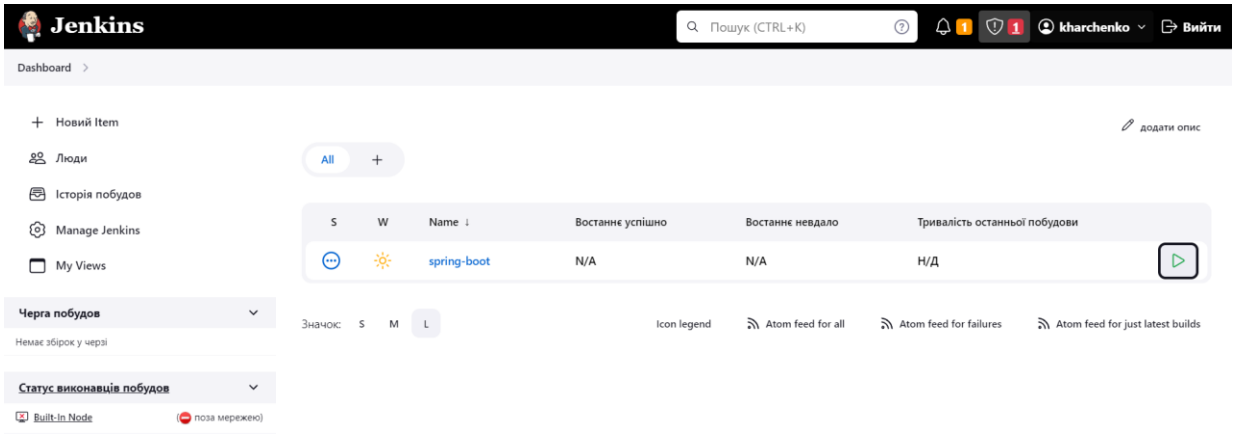


Рис. 2. 15. Головна сторінка Jenkins

Наступним кроком було налаштування tomcat сервера на якому буде працювати серверна частина веб-сайту. Після підключення було встановлено jdk та завантажено tomcat server з офіційного сайту. Щоб запустити сервер потрібно перейти по маршруту /opt/tomcat/bin та запустити файл startup. Для перевірки потрібно перейти на сервер з портом 8080.

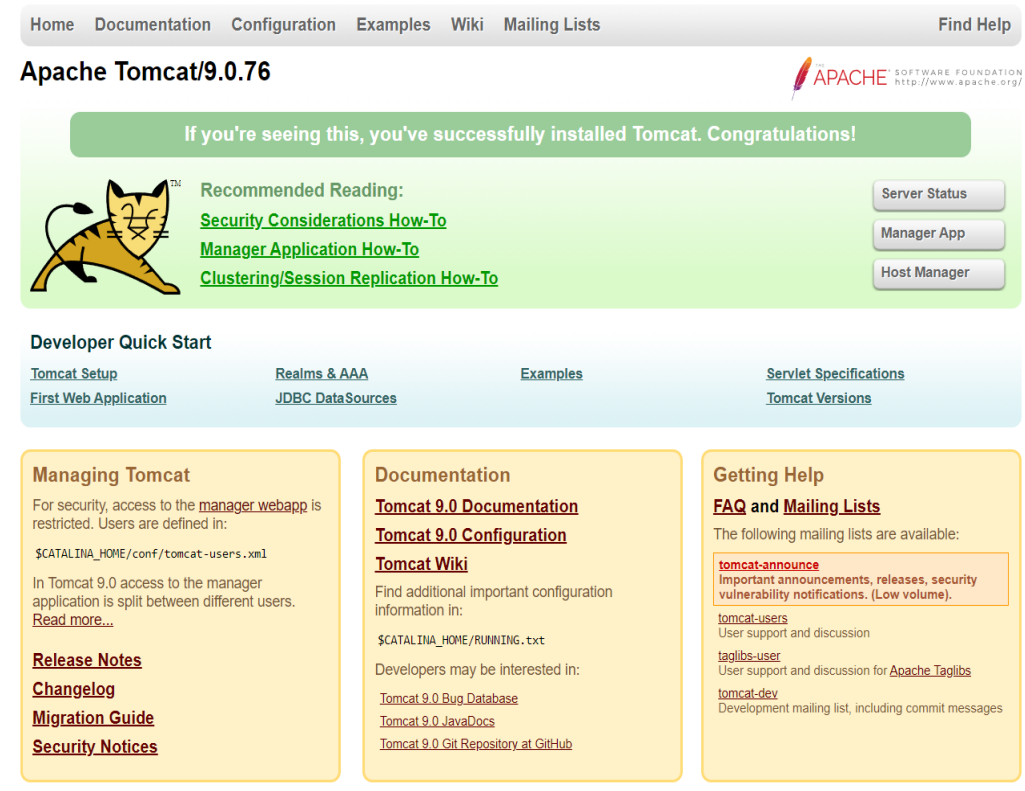


Рис. 2. 16. Головна сторінка tomcat сервера

Ansible - це інструмент управління конфігурацією та автоматизації, який дозволяє розгортати, налаштовувати та керувати великими масштабами серверів та інфраструктури. Він дозволяє розробникам та операторам здійснювати конфігурацію та управління серверами швидко, просто та ефективно. Основна мета використання Ansible - це автоматизувати рутинні задачі налаштування та управління, що зменшує ймовірність помилок та забезпечує консистентність у всій інфраструктурі та дозволяє зручно описувати та керувати конфігурацією серверів. За допомогою Ansible playbooks, можна визначити стан бажаних налаштувань серверів та автоматично застосовувати їх до великої кількості машин. Ansible спрощує розгортання додатків на серверах. Він дозволяє автоматично встановлювати залежності, налаштовувати параметри середовища.

Також Ansible дозволяє зберігати конфігурацію в системі контролю версій, що спрощує спільну роботу та відстеження змін конфігураційних файлів.

Після цього було налаштовано Ansible сервер. Після того як було підключено до сервера необхідно встановити pip installer та ansible виконавши команди:

```
sudo yum install pip
pip install ansible
```

Також було під’єднано через через ssh ключ до сервера з tomcat, щоб ansible playbooks виконувалися б автоматично. Потрібно створити плейбук, який буде доставляти оновлений jar файл серверної частини сайту з github в Jenkins pipeline і також плейбук, що отримує код веб-сайту з github.

```yaml
- name: Set up multiple authorized keys
  authorized_key:
    user: clouduser
    state: present
    key: '{{ item }}'
  with_file:
    - public_keys/some-user.pub

- name: Ensure kernel is at the latest version
  yum: name=kernel state=latest

- name: Install latest Java 8
  yum: name=java-1.8.0-openjdk.x86_64 state=latest

- name: Install Netcat
  yum: name=nmap-ncat state=latest
```

Рис. 2. 17. Код common-task.yaml

```yaml
- name: Create skeleton directory
  file: path=/opt/skeleton state=directory

- name: Download skeleton runnable jar
  get_url:
    url: https://github.com/KharchenkoV/gym-backend
    dest: /opt/skeleton/skeleton.jar
    backup: yes
    force: yes

- name: Ensure app is configured
  template:
    src: application.properties.j2
    dest: /opt/skeleton/application.properties

- name: Ensure logging is configured
  template:
    src: logback-spring.xml.j2
    dest: /opt/skeleton/logback-spring.xml

- name: Install skeleton systemd unit file
  template: src=skeleton.service.j2 dest=/etc/systemd/system/skeleton.service

- name: Start skeleton
  systemd: state=restarted name=skeleton daemon_reload=yes
```

Рис. 2. 18. Код deploy.yaml

Також створений плейбук для того щоб об’єднати минулі плейбуки.

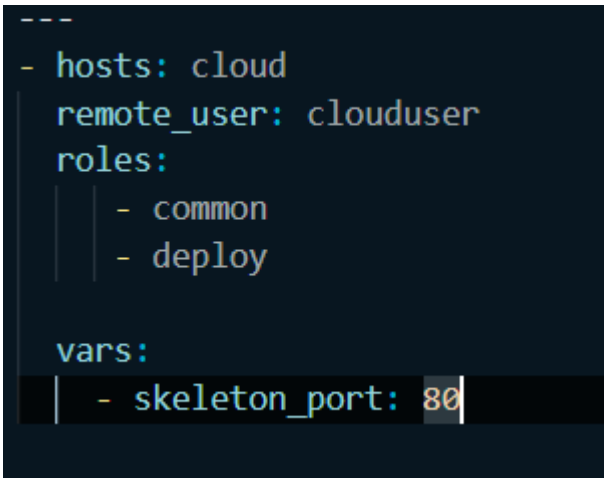


Рис. 2. 19. spring-playbook.yaml

Останнім етапом налаштування було налаштування pipeline на Jenkins. Jenkins дозволяє розробникам налаштовувати та виконувати CI/CD процеси з використанням різних інструментів та платформ. Jenkins дозволяє автоматично збирати, компілювати та тестувати програмний код з репозиторіїв у системах контролю версій, таких як Git. Він дозволяє налаштувати регулярні запуски збирання коду та надавати звіти про стан збирання також надає можливість налаштовувати та запускати автоматичні тести на основі визначених критеріїв. Це дозволяє виявляти помилки та проблеми швидше, забезпечуючи високу якість програмного забезпечення.

Jenkins дозволяє інтегрувати код, що розробляється різними розробниками, у спільну базову гілку. Він автоматично перевіряє конфлікти, виконує збірку та тести, щоб забезпечити цілісність коду та надає гнучкість для налаштування CI/CD процесів. Розробники можуть налаштовувати логіку виконання кроків, додавати спеціалізовані плагіни та розширення, створювати скрипти та налаштовувати управління залежностями.

Для налаштування потрібно перейти на сторінку керування Jenkins та створити pipeline, який збирає воєдино всю вище описану інфраструктуру та запустити.

```
pipeline {
    agent { label 'master' }
    tools {
        maven 'Maven 3.3.9'
        jdk 'jdk1.8.0'
    }
    stages {
        stage('Build') {
            parallel {
                stage('Build Backend'){
                    steps {
                        dir('backend'){
                            sh '
mvn clean install spotbugs:spotbugs checkstyle:checkstyle deploy'
                        }
                    }
                }
                stage('Build Frontend'){
                    steps {
                        dir('frontend'){
                            sh 'npm install --save'
                            sh 'npm install jest-cli --save'
                            sh 'mvn clean install'
                        }
                    }
                }
            }
        }
        stage('Deploy to test'){
            steps {
                dir('deployment'){
                    echo 'Deploying to test'
                    sh '
ansible-playbook -i dev-servers spring-playbook.yaml'
                }
            }
        }
        stage('API tests'){
            steps {
                echo 'Executing API tests'
            }
        }
        stage('Performance tests'){
            steps {
                echo 'Executing Performance tests'
            }
        }
    }
}
```

Рис. 2. 20. Jenkins pipeline скрипт

РОЗДІЛ 3 ІНТЕРФЕЙС КОРИСТУВАЧА

3.1 Огляд інтерфейсу користувача

Після того як користувач переходить на веб-сайт, йому доступна головна сторінка.

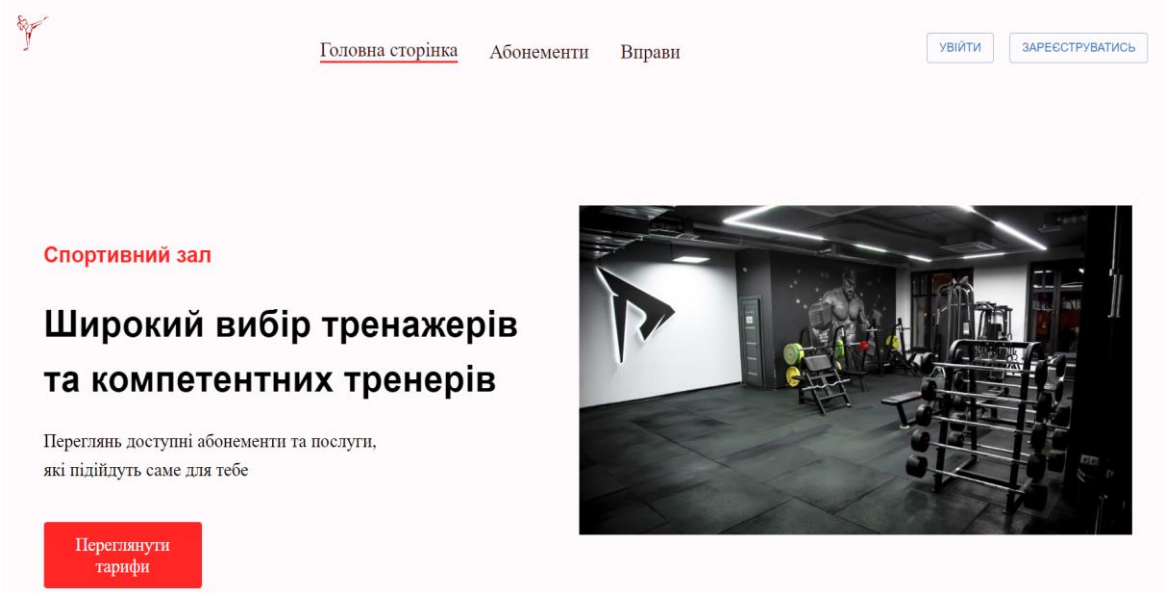


Рис. 3.1. Головна сторінка програми

Користувачеві буде доступна основна інформація про спортивний зал та меню навігації. Меню навігації складається з посиланнями на “Головну сторінку”, “Абонементи”, “Колекцію вправ” і також посилання на сторінки логіну та реєстрації.

Для того, щоб у користувача була можливість здійснювати покупки необхідно авторизуватися. Для того щоб новому користувачеві авторизуватися потрібно створити новий акаунт перейшовши на сторінку “Зареєструватись” .

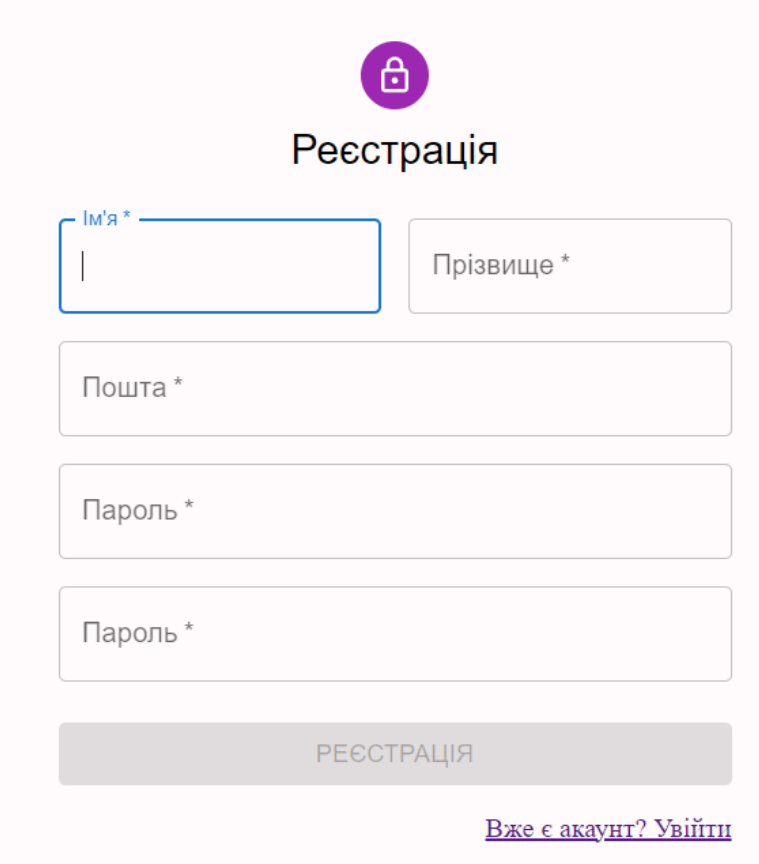


Рис. 3.2. Форма реєстрації

Перейшовши за посиланням користувачеві пропонується заповнити дані. Якщо дані були введені коректно користувач аутентифікується автоматично. Також якщо користувач вже має акаунт, він переходить по посиланню “Увійти” та автентифікується по логіну та паролю.

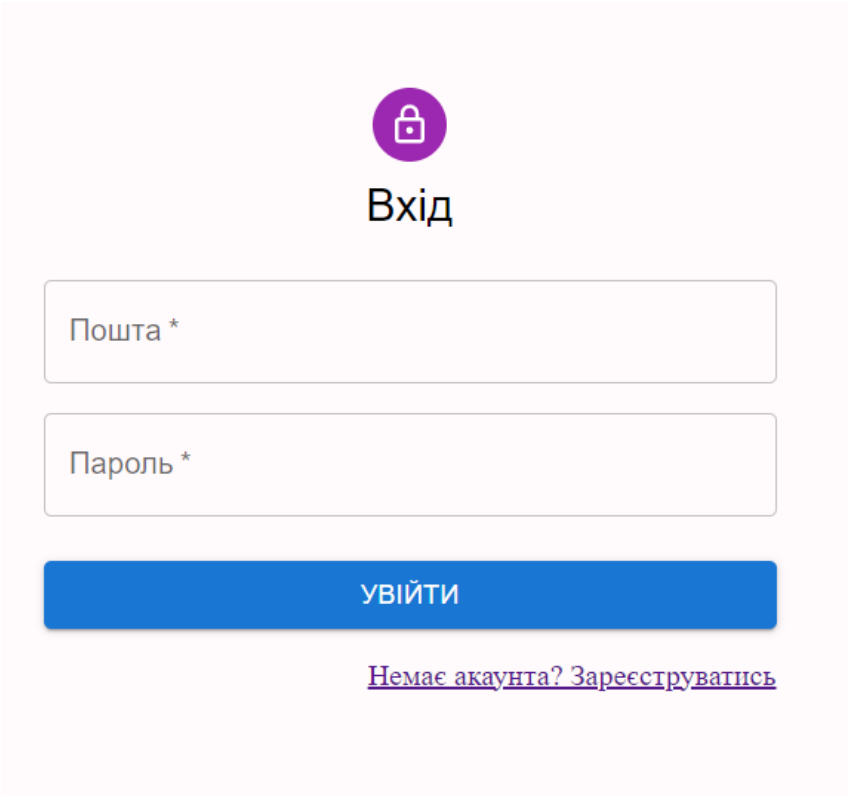


Рис. 3.3. Форма входу

Якщо користувач перейде по посиланю “Абонементи”, йому відобразиться інформація про доступні тарифи абонементів.

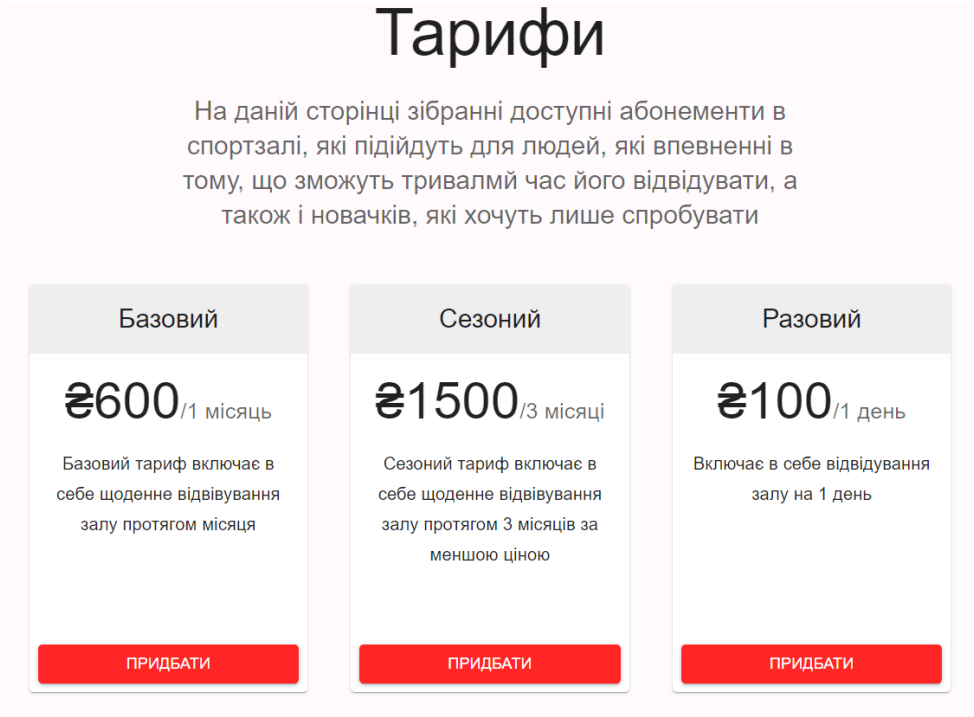


Рис. 3.4. Сторінка з тарифами абонементів

Користувач зможе оплатити тариф лише коли авторизований.

Також, якщо користувач перейде по посиланю “Вправи” відкриється сторінка в якій зібраний тека вправ на кожну м’язову групу, інформація про необхідне обладнання та приклад виконання.

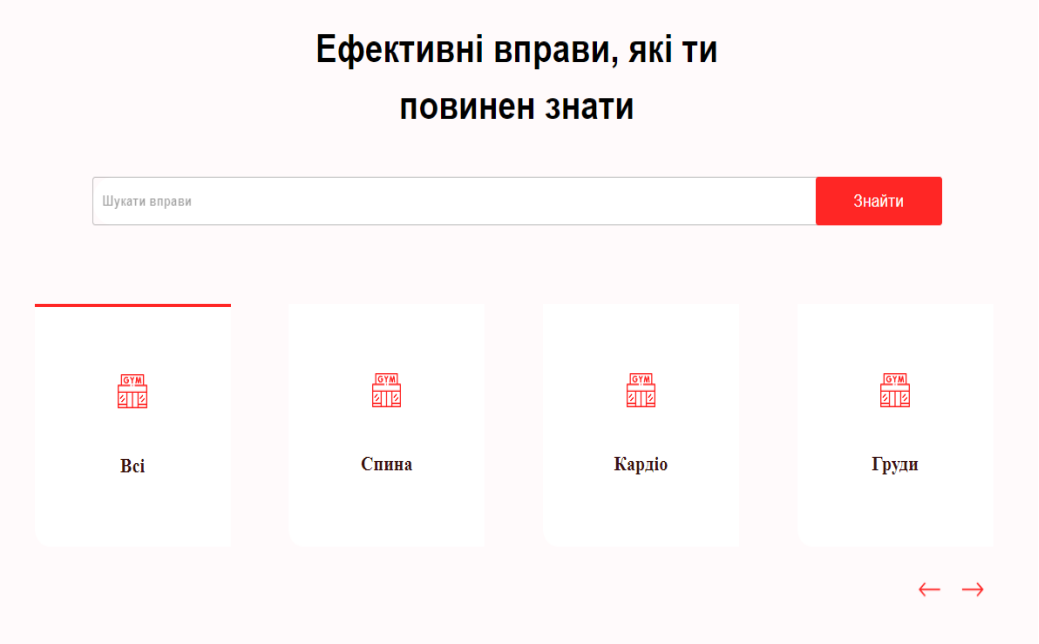


Рис. 3.5. Навігація по вправам

При переході на сторінку в першу чергу користувач бачить навігацію по вправам, а саме можливість пошуку по назві вправи і також по категоріям частини тіла.

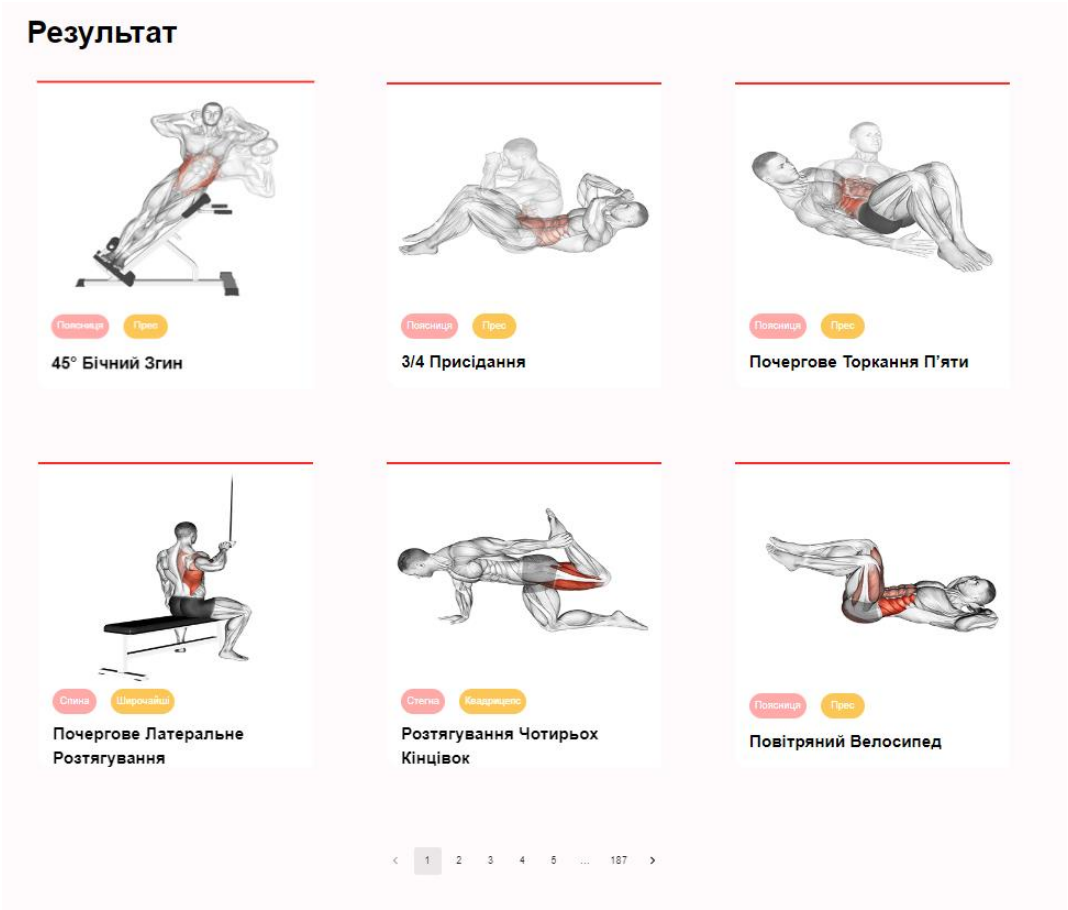


Рис. 3.6. Відображення вправ на сторінці

Також користувач має можливість переглянути до якого терміну діє абонемент на сторінці “Аккаунт”.

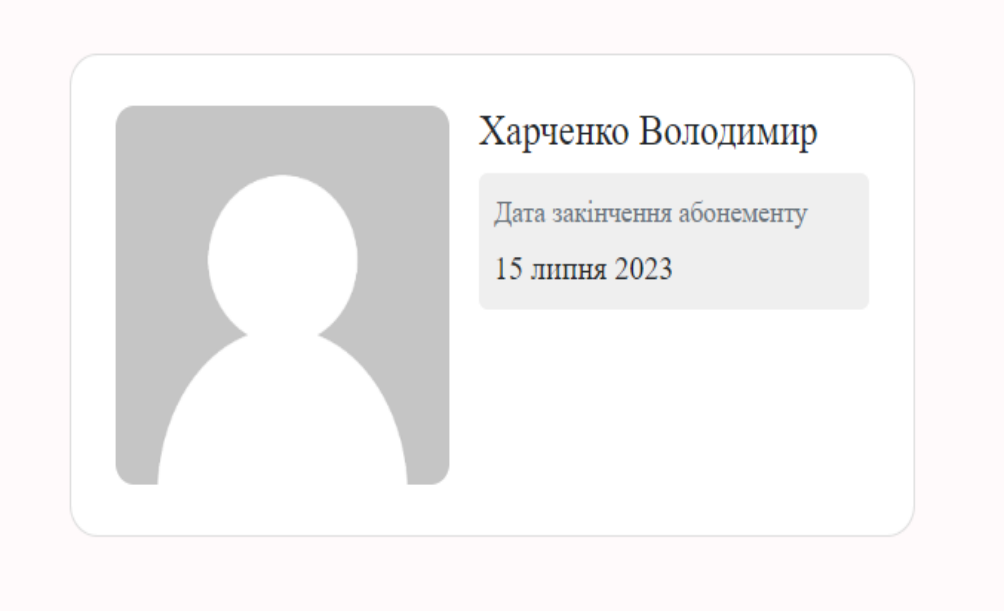


Рис. 3.7. Аккаунт користувача

ВИСНОВКИ

У кваліфікаційній роботі був розроблений та розгорнутий на хмарних серверах веб-сайт спортивного залу. Використано технології CI/CD для створення pipeline та надано можливість обміну даних між компонентами системи.

Налаштування автоматичної збірки та деплою було виконане з допомогою технологій Ansible та Jenkins. Та було реалізовано з допомогою AWS EC2. Взаємодія між серверами виконувалася через протокол SSH.

Серверна та клієнтська частини виконувалася на мовах програмування java і javascript відповідно. Вони побудовані з використанням архітектури REST, що дозволяє додати нові мікросервіси при їх необхідності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Continuous Integration: CircleCI vs Travis CI vs Jenkins vs Alternatives [Електронний ресурс] // Django Stars. – 2023. – Режим доступу до ресурсу: https://djangostars.com/blog/continuous-integration-circleci-vs-travisci-vs-jenkins/
2. Installing Jenkins [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://jenkins.io/doc/book/installing/
3. Сайт хмарного сервісу Amazon Web Services [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/devops/continuous-delivery/?nc1=h_ls
4. 8. What is Continuous Delivery? [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://aws.amazon.com/devops/continuous-delivery/?nc1=h_ls
5. What is DevOps? The Ultimate Guide to DevOps [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://resources.collab.net/devops-101/what-isdevops
6. Ansible Documentation. Basic Concepts [Електронний ресурс] – 2023 –Режим доступу: https://docs.ansible.com/ansible/latest/network/getting_started/basic_concepts.html
7. Spring Framework [Інтернет-ресурс] / Режим доступу: https://spring.io/
8. Documentation for ReactJS [Електронний ресурс] – Електронні дані. – Режим доступу : https://uk.reactjs.org/