МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

Автоматизація розгортання сервісу за допомогою Vagrant

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник курсової роботи

к.т.н., ст. викладач

Черкасов Д. І.

_____

(підпис)

“____” _____ 2022 р.

Виконав студент 1 курсу

Іщенко І.О.

“____” _____ 2022 р.

Тема:Автоматизація розгортання сервісу за допомогою Vagrant

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	01.11.2021	
2.	Огляд технічної літератури за темою роботи	15.12.2021	
3.	Виконання аналізу сучасних рішень	27.04.2022	
4.	Розробка застосунку	14.05.2022	
5.	Написання пояснювальної записки	08.06.2022	

Студент _____

Керівник _____

“____”_____

Керівник курсової роботи

1. Вступ

Кожен програмний продукт у світі має пройти певний життєвий цикл для того, щоб в робочому вигляді вчасно потрапити до кінцевого користувача. Одним з найбільш важливих етапів є

розгортання застосунку на тестовому або робочому середовищі. Для того, щоб цей процес відбувався швидко без перебоїв та помилок, він має відбуватись автоматично. Автоматизація розгортання застосунку найбільш актуальна для тих організацій, які мають постійно випускати нові версії та покращувати продукт.

Автоматизація має певні переваги для бізнесу, наприклад:

- будь-який член команди може запустити розгортання застосунку;
- більш швидкий та ефективний процес розгортання, порівняно з ручним способом;
- збільшення продуктивності команди компанії;
- Зменшення кількості помилок;
- можливість збільшити кількість випусків продукту;
- своєчасне інформування про стан збірки.

Існує доволі багато інструментів для автоматизації розгортання та налаштування середовища. Один із таких інструментів буде розглянуто у цій роботі. Vagrant – система для налаштування та керування віртуальними машинами, що створені за допомогою гіпервізора. Vagrant надає робочі середовища, що легко настроюються, відтворюються та переносяться між системами. Середовища керуються одним процесом, щоб допомогти досягти максимальної продуктивності та гнучкості у налаштуванні та розгортанні застосунку. Це доволі корисний інструмент для розробників програмного забезпечення, адже він ізолює залежності та їх конфігурацію в одноразовому середовищі. Все, що необхідно - це створити та налаштувати єдиний Vagrantfile, який містить у собі усю конфігурацію системи, та запустити інструмент командою “vagrant up”. Після цього система буде налаштована та готова для роботи. Якщо потреби команди не обмежуються однією версією конфігурації системи, в такому випадку в одному файлі можливо описати декілька сценаріїв налаштування для двох та більше віртуальних машин.

Завдяки автоматизованій розгортці середовища, можна легко конфігурувати систему, що збільшує продуктивність та якість роботи команди, адже тестування та розробка буде виконуватись в середовищі з однаковими умовами. Також налаштування автоматизації розгортання системи, допоможе новим розробникам, які будуть вносити оновлення та зможуть легко застосовувати нові зміни. Адже не потрібно буде вручну створювати необхідну інфраструктуру, а лише використати вже налаштоване середовище.

Дана тема є доволі актуальною на сьогодні, адже все ще не всі підприємства користуються усіма привілеями автоматизованого розгортання через власні обмеження. Мета роботи полягає в демонстрації переваг автоматизації процесів розгортання для тестування роботи програмного забезпечення за допомогою системи Vagrant.

Планується розробити інформаційний блог для використання всередині підприємства та обмінну знаннями між командами. Цю систему можна буде розгортати на хмарних сервісах для використання в продакшені, та у налаштованому віртуальному середовищі для тестування.

Дана робота має ціль розкрити та продемонструвати наступне:

- переваги та недоліки Vagrant;

- розробка системи для обміну знань всередині компанії;
- розгортання сервісу на віртуальній машині;
- додаткова конфігурація системи за допомогою сторонніх інструментів;
- переваги автоматизованого розгортання системи порівняно з ручним способом;

2. Огляд існуючих рішень

Автоматизація розгортання застосунку допомагає набагато швидше та частіше випускати нові версії програмного забезпечення, адже сучасні інструменти майже повністю виключають ручне втручання. Сучасні системи автоматизації розгортання доволі прості в налаштуванні, можуть відтворювати робочі середовища та конфігурацію. Зазвичай інструменти автоматизації мають певні сфери застосування, та мають різний функціонал. Інструменти автоматизації можна порівняти за певними критеріями :

- Підтримка одного та більше серверів для розгортання та підтримки;
- Підтримка гіпервізорів;
- Інтеграція з хмарними платформами;
- Наявність плагінів для інтеграції з іншими сервісами;
- Простота налаштування;
- Платний/безкоштовний продукт;

2.1 Ansible

Ansible – це доволі простий та зрозумілий інструмент автоматизації інфраструктури, який виконує підготовку робочих середовищ, дозволяє керувати конфігурацією, дозволяє налаштувати розгортання застосунку та побудувати оркестрацію сервісів. Ansible використовує доволі просту мову для написання скриптів – YAML в форматі Ansible Playbook. Ansible Playbook дозволяє розгортати віртуальні машини з образів ISO та використовувати шаблони автоматизації. Для захисту конфіденційних даних, необхідно створити файл Ansible Vault, який буде зберігати необхідні паролі в шифрованому вигляді. Playbook, який використовує цей файл, розшифровує його під час виконання з використанням встановленого паролю.

2.1.1 Підтримка одного та більше серверів

Ansible має функціонал з підтримки та конфігурування кількох серверів. Він дозволяє легко клонувати віртуальні машини з шаблону, що дозволяє в короткий строк налаштовувати декілька віртуальних робочих середовищ. Можна використовувати як шаблони для розгортання віртуальних машин і далі вже налаштовувати інфраструктуру вже з готових шаблонів, або створювати їх з нуля.

2.1.2 Підтримка гіпервізорів

Так, Ansible має підтримку роботи з гіпервізорами. Для прикладу можна оглянути роботу з VMware. Ansible допомагає автоматизувати інфраструктуру VMware та прискорити процес від етапу розробки до виходу застосунку в виробниче середовище. Для спрощення роботи можна

використовувати Ansible Playbook і шаблони завдань для опису роботи з середовищем, а також створювати і розгортати застосунки одним запуском на виконання даного плейбуку.

2.1.3 Інтеграція з хмарними платформами

З Ansible можливо інтегруватись з хмарними платформами так само легко, як і розгортати локальну інфраструктуру. Той самий плейбук, який використовувався для розгортання на локальних або фізичних ресурсах, можна використати і для розгортання в хмарному або гібридному середовищі, адже плейбук зберігає налаштовану конфігурацію, яка має застосуватись на цільовій машині. Ansible допомагає зменшити необхідність навчати, як саме потрібно працювати з певним провайдером хмарних послуг, адже кожне розгортання буде чітко використовувати вже налаштовані політики та налаштування.

2.1.4 Інтеграція з іншими сервісами

Роботу з Ansible можливо розширити використовуючи власні, або вже створені модулі та плагіни, необхідні для роботи. Сервіс викристовує архітектуру під'єнування модулів, для того щоб забезпечити максимально гнучкий та розширюваний набор функцій. Ansible має інтеграції з хмарними провайдерами, такими як: AWS, GCP, Azure, тощо. Також Ansible має інтеграцію з сервісами Atlassian, дозволяє зв’язувати різні етапи життєвого циклу розробки продукту від розробки програмного коду до розгортання виробничого середовища.

2.1.5 Простота налаштування

Ansible це інструмент автоматизації, який встановлюється на головному вузлі, та віддалено керує машинами та другими пристроями за замовченням з використанням протоколу SSH. Загалом Ansible використовує для налаштування сценаріїв мову YAML, яка є доволі простою у використанні та знайомою для багатьох IT фахівців сьогодні. З використанням даної мови серіалізації даних скрипти автоматизації більше схожі на звичайний текст англійською мовою, який може бути зрозумілий навіть для звичайних користувачів.

2.1.6 Вартість використання сервісу

Ansible має 2 плани – standard та premium. Вони відрізняються часом підтримки. Для стандартної версії це 8\5, а для преміум версії це цілодобова підтримка. Також можна отримати пробну версію на 60 днів.

2.2 vRealize Automation

vRealize Automation – це сервіс для керування інфраструктурою для налаштування та автоматизації застосунків на основі контейнерів. Є частиною набору продуктів vRealize Suite для керування гібридними хмарними середовищами. Цей сервіс створений для автоматизації одного та більше хмарних середовищ за допомогою безпечного та самостійного забезпечення та допомагає спеціалістам налаштовувати ресурси, які необхідні для роботи. Серед функцій можна виділити наступні :

- Користувачі можуть налаштовувати багато хмарні середовища з централізованим керуванням ресурсів;
- Можна налаштувати попередження вразливостей та політики відповідності критеріям системи;

- Надає можливість автоматизації інфраструктури Kubernetes, а також підтримує сторонні сервіси.

2.2.1 Підтримка одного та більше серверів

Так, vRealize Automation підтримує роботу з кількома цільовими машинами. Інструмент має можливість керувати та конфігурувати декілька машин. За допомогою інструменту vRealize Orchestrator є можливість налаштувати сценарії, які будуть взаємодіяти з необхідною кількістю серверів.

2.2.2 Підтримка гіпервізорів

Оскільки vRealize Automation є частиною пакету vRealize Suite від VMware, можна зазначити, що інструмент має підтримку гіпервізорів. Для роботи з гіпервізором, необхідно запустити сервіс на цільовій машині, яка підтримує віртуалізацію.

2.2.3 Інтеграція з хмарними платформами

Так, vRealize Automation має можливість працювати з хмарними платформами. З 8 версії vRA сервіс має підтримку : AWS EC2, AWS GovCloud, Azure, Azure Government.

2.2.4 Інтеграція з іншими сервісами

vRealize Automation підтримує роботу з Ansible, Puppet та Terraform. Також можливо розгортати кластери vSphere з Tanzu Kubernetes Grid в режимі самообслуговування, або через хмарні шаблони VMware.

2.2.5 Простота налаштування

В складі пакету vRealize Suite використовуються інструменти для повного керування життєвим циклом застосунку та вирішення інших завдань в процесі розробки.

2.2.6 Вартість використання сервісу

Нажаль, сервіс не має безкоштовної версії, проте має безкоштовну пробну версію на 45 днів.

2.3 Puppet

Puppet – це інструмент з відкритим вихідним кодом, який допомагає керувати та автоматизувати налаштування серверів. Під час використання сервісу необхідно визначити бажаний стан систем в інфраструктурі, якою необхідно керувати. Для використання необхідно написати код на предметно орієнтованій мові Puppet Code, який можна використовувати з широким спектром пристроїв та операційних систем. Для роботи використовуються Puppet primary server та Puppet agent. Основний сервер – це місце де зберігається основний код, а агент – це інструмент який переводить код в команди для подальшого виконання в налаштованих системах. Це називається запуском Puppet.

2.3.1 Підтримка одного та більше серверів

Так, Puppet підтримує роботу на декількох серверх одночасно. Можливо автоматизувати установку Puppet Master та Puppet Agent на віртуальних машинах. Якщо Puppet встановлено на віртуальну машину, то з’являється можливість повністю керувати її роботою за необхідністю. Puppet server є кросс-платформенним інструментом, це означає що використання Puppet можливо на машині з будь-якою операційною системою.

2.3.2 Підтримка гіпервізорів

Puppet підтримує роботу з сучасними гіпервізорами. Для роботи з віртуальною машиною на якій встановлено Puppet можна доєднатись з використанням протоколу SSH, та почати роботу.

2.3.3 Інтеграція з хмарними платформами

Puppet Enterprise забезпечує спільну роботу усіх ключових технологій автоматизації, які можуть знадобитись в процесі інтеграції з хмарною платформою. Для налаштування на хмарній платформі, достатньо розгорнути віртуальну машину, та встановити на неї Puppet.

2.3.4 Інтеграція з іншими сервісами

Puppet підтримує декілька видів плагінів, які використовуються модулями. Можливо використовувати як вже існуючі плагіни, так і створювати власні. На початку кожного сеансу сервер Puppet підвантажує усі підключені модулі, доступні в середовищі. Puppet досить добре працює в парі з Vagrant, та дозволяє отримувати додаткові функції для конфігурування та автоматизації розгортання середовища.

2.3.5 Простота налаштування

Puppet доволі просто встановлюється на цільову машину, та дозволяє підключати додаткові модулі з першого моменту запуску. Після того, як Puppet з модулями було встановлено, непотрібно заново конфігурувати інфраструктуру після кожного запуску, адже Puppet завантажує необхідні компоненти самостійно.

2.3.6 Вартість використання сервісу

Puppet є безкоштовним інструментом, але для великих систем може знадобитись версія Puppet Enterprise. З Puppet Enterprise можна отримати доволі зручний користувацький інтерфейс, який буде відображати аналітику та корисну інформацію з усіх середовищ. Ціна варіюється від 112 до 199 доларів за робочий вузол.

2.4 Vagrant

Vagrant це доволі простий у використанні інструмент для роботи з віртуальними середовищами. Vagrant надає доволі простий у використанні консольний клієнт для керування та автоматизації розгортання середовища. Vagrant є кросс-платформенним, тож може запускатись на будь-якій операційній системі, що значно полегшує роботу, та дозволяє використання сервісу на будь-якій машині. Для налаштування розгортання віртуального середовища використовується доволі проста мова YAML, якою описуються всі необхідні налаштування всередині файлу Vagrantfile. Оскільки усі налаштування описані в одному файлі, це спрощує клонування та перенесення робочих середовищ з однаковими налаштуваннями між командами. Vagrant дозволяє доволі просто описувати та підключати роботу багатьох компонентів системи таких як : сервер бази даних, користувацький інтерфейс, веб-сервер, балансувальник навантаження, тощо. Інструмент дозволяє зменшити кількість помилок ручного налаштування кожної частини системи окремо, адже опис інфраструктури знаходиться в одному файлі.

2.4.1 Підтримка одного та більше серверів

Vagrant може визначати та контролювати декілька гостьових машин для кожного конфігураційного файлу. Ці машини здатні працювати в зв’язці. Це може бути використано для розподілу декількох

серверів, моделюванню розподіленої системи або для тестуванню роботи системи в аварійних ситуаціях. Запуск складних середовищ з багатьох машин часто може привести до непередбачуваних наслідків, що може викликати подальші помилки. Цю проблему Vagrant вирішує тим, що сервіс дозволяє моделювати усі середовища в контексті одного середовища Vagrant.

2.4.2 Підтримка гіпервізорів

Vagrant має доволі гарну підтримку роботи в парі з Virtual Box та Hyper-V. Також сервіс може працювати з іншими типами віртуальних машин, що досягається за допомогою окремих провайдерів Vagrant. Оскільки для роботи можуть бути необхідні додаткові функції, які більше підходять для необхідних варіантів використання. Для початку роботи з окремим провайдером, необхідно попередньо його інсталювати через систему плагінів Vagrant.

2.4.3 Інтеграція з хмарними платформами

Наразі Vagrant має наявний функціонал cloud-init, який дозволяє ініціювати розгортку середовища в хмарі. Він підтримується основними провайдерами хмарних послуг, приватними хмарами та встановленню на власному залізі. При завантаженні cloud-init ідентифікує хмару в якій працює та завантажує необхідні метадані для роботи в хмарі для налаштування системи. Це може включати налаштування мережі, сервісів зберігання даних, налаштування ключів SSH, тощо. Наразі цей функціонал знаходиться на етапі тестування, тож можуть бути певні перебої в роботі системи.

2.4.4 Інтеграція з іншими сервісами

Vagrant має дуже широкий набір плагінів які можуть сильно розширити та полегшити роботу по налаштуванню інфраструктури. Система надає можливість використовувати вже готові плагіни, які називаються «коробками», або створювати та підключати власні. Переглянути усі наявні плагіни можна за допомогою сервісу Vagrant Cloud, який включає в себе як вже готові базові рішення для розгортання інфраструктури, так і додаткові інструменти для автоматизації, конфігурування та роботи з середовищем.

2.4.5 Простота налаштування

Vagrant для опису конфігурації середовищ використовує доволі просту мову YAML, яка дозволяє описати усі необхідні налаштування системи в одному файлі Vagrantfile. Це дозволяє без зайвих зусиль описати взаємодію усіх компонентів інфраструктури в одному місці та уникнути помилок конфігурування системи окремо. Після завершення опису системи, достатньо виконати команду vagrant up, яка встановить усі необхідні компоненти та запустить систему.

2.4.6 Вартість використання сервісу

Vagrant має 3 плани використання, один з яких є безкоштовним. Також існує два плани, які відкривають додатковий функціонал, такий як : створення персональних «коробок» з плагінами, та можливість ділитися плагінами між командами. Платні версії сервісу дозволяють користувачам отримати додаткові функції необхідні для роботи. Для особистої роботи існує план Personal, який коштує 5 доларів на місяць\особисту «коробку». Для роботи в компанії існує план Organization, який коштує 25 доларів на місяць\ «коробку».

2.5 Порівняльний аналіз

На сьогодні існує велика кількість сервісів, які допомагають автоматизувати розгортання застосунку, спростити конфігурування середовищ та налаштовувати автоматичні пайплайни CI\CD. Велика кількість сервісів є доволі складними у використанні та потребують доволі глибких навичок в налаштуванні інфраструктури та мережі. Vagrant використовує доволі простий формат для опису налаштування середовища, та доволі легко дозволяє створювати та відтворювати робочі системи, які можуть максимально імітувати реальні робочі ситуації. На відміну від усіх вище описаних інструментів, Vagrant має найбільш повну безкоштовну версію, та дозволяє використовувати більшу кількість функціоналу. Також Vagrant має велику кількість плагінів, які дозволяють розширювати можливості для налаштування та збільшення продуктивності у відтворенні робочих процесів та створенні віртуальних середовищ.

3. Структурна розробка власного рішення

Для розробки блогу обрано мову C# та фреймворк ASP.NET Core 6 для розробки найбільш продуктивного та кросс-платформенного рішення для побудови серверної частини. Для розробки частини користувацького інтерфейсу обрано фреймворк Razor Pages та Bootstrap, який дозволяє створювати візуальну частину з використанням мови C#. Для полегшення роботи застосунку з базою даних обрано ORM Entity Framework Core, яка дозволяє повністю конфігурувати базу даних з використанням підходу Code-First та доволі ефективно будувати запити. Для налаштування та конфігурування віртуального середовища в рамках даної роботи використано сервіс Vagrant та гіпервізор Virtual Box для максимально швидкої та комфортної роботи. Основними вимогами до застосунку є :

- Застосунок має працювати на створеному віртуальному середовищі та відповідати на запити з використанням протоколу HTTP;
- Дані мають зберігатись в реляційній базі даних;
- Має бути створений користувацький інтерфейс для тестування валідної роботи застосунку;

Для опису конфігурації системи створено Vagrantfile в якому описано кроки по створенню середовища, на яке буде розгортатись система для тестування та роботи. Для опису компонентів та налаштування віртуального середовища Vagrant використовує мову YAML, яка є доволі простою для розуміння та дозволяє швидко описувати усі необхідні компоненти системи. Для керування віртуальним середовищем обрано гіпервізор Virtual Box, який має доволі простий та зрозумілий інтерфейс, та дозволяє легко переглядати наявні робочі віртуальні середовища та керувати ними.

3.1 Опис компонентів застосунку

3.1.1 Компонент веб-сервісу

Даний компонент реалізує повноцінний веб-застосунок, який має користувацький інтерфейс та контролери, які використовують архітектуру REST API для керування сутностями, які зберігаються в базі даних. Фреймворк Razor Pages дозволяє створювати веб-сторінки та контролери для обробки запитів в суміжних класах для створення більш ефективного застосунку. Для стилізації та спрощення розробки користувацького компонента використано bootstrap, який надає готову таблицю стилів та дозволяє швидко розробити зручний інтерфейс

користувача. Компонент має сторінку з усіма наявними категоріями блогу, сторінку з перегляду постів за категорією, сторінку з усіма постами та сторінку з переглядом повного тексту поста.

3.1.1.1 Сторінка категорій

На цій сторінці користувач може переглянути повний список категорій блогу, ця сторінка відкривається при переході на головну сторінку застосунку. Також є можливість створювати нові категорії, які будуть необхідні для створення нових постів.

3.1.1.2 Сторінка постів за категорією

На цій сторінці користувач може переглянути усі пости за певною категорією, а також перейти на сторінку створення нового поста.

3.1.1.3 Сторінка усіх постів

На цій сторінці користувач може переглядати усі пости, які наявні в системі за усіма категоріями. Користувач може перейти по назві на сторінку огляду поста, який зацікавив, а також перейти на сторінку створення нового поста.

3.1.1.4 Сторінка огляду поста

На цій сторінці користувач може побачити повний текст посту, інформацію коли його було створено, а ім’я автора. Також на сторінці під постом можна побачити найсвіжіші створені пости.

3.1.1.5 Сторінка створення поста

На цій сторінці користувач може створити новий пост шляхом вказанням названня посту, тексту, ім’я автора, дату створення та назву категорії з випадаючого списку. Після цього на цей пост можна перейти зі сторінки категорії, для якої цей пост було створено, а також на сторінці з усіма постами.

3.1.2 Компонент роботи з базою даних

Для взаємодії з сервером бази даних використовується ORM Entity Framework Core, яка дозволяє ефективно виконувати запити з використанням мови C#. Моделі класів пов’язані з сутностями бази даних через Fluent API, це інструмент для налаштування взаємозв’язків між сутностями та їх властивостями. В застосунку існує клас ApplicationDbContext в якому визначені колекції з специфічним типом DbSet, які пов’язані з таблицями в базі даних, та дозволяють робити запити напряму до таблиць з використанням LINQ. LINQ – Language-Integrated Query це інструмент побудови запитів до будь-якої колекції з допомогою методів, які схожі на мову SQL. Сервером бази даних виступає Microsoft SQL Server, який дозволяє легко керувати даними та ефективно обробляти запити. Entity Framework Core отримує доступ до сервера бази даних через спеціальний рядок підключення, який зберігається в змінних середовища та визначає повний шлях до працюючого сервера.

3.1.3 Компонент налаштування робочого середовища

Для налаштування робочого середовища використовується Vagrantfile та Virtual Box. Для початку роботи необхідно обрати цільову операційну систему, яка буде розгортатися і знайти необхідний образ серед Vagrant Box.Після вибору образу віртуальної машини, необхідно виконати команду **vagrant init** <ім’я образу>. Після цього Vagrant згенерує Vagrantfile (Додаток 8) в цільовому каталозі

з необхідними налаштуваннями для запуску віртуальної машини. В ньому містяться налаштування віртуальної машини, назва Vagrant Box, порти які будуть відкриті для переадресації та інструкції для налаштування мережі. Після внесення будь-який правок у Vagrantfile необхідно перезапустити віртуальну машину командою vagrant reload.

3.2 Опис процесу розгортки

Для запуску віртуальної машини з налаштуваннями, які знаходяться в Vagrantfile, необхідно перейти в каталог, де він знаходиться і виконати команду в консолі vagrant up. Після цього віртуальна машина запуститься і з'явиться можливість під'єднатися до неї за протоколом ssh, виконавши команду vagrant ssh. Для роботи з фреймворком .NET необхідно встановити його на віртуальну машину виконавши певні команди (Додаток 9). Для перевірки, чи встановився SDK, необхідно виконати команду dotnet –version та звірити чи збігається необхідна версія фреймворка із тим, який встановлено на віртуальну машину. Наступним кроком, необхідно встановити веб-сервер, який дозволить працювати застосунку в мережі в якості зворотнього проксі-сервера, який приймає HTTP запити та переспрямовує його на Kestrel (веб-сервер який ввімкнений за дефолтом у фреймворку ASP.NET). Для даної роботи було обрано веб-сервер Apache через простоту в налаштуванні. Після встановлення веб-серверу до операційної системи необхідно ввімкнути модуль modproxy для роботи як зворотній проксі-сервер. Для розгортання застосунку на віртуальній машині, необхідно перенести усі бінарні файли та бібліотеки, які використовує застосунок на віртуальну машину. Vagrant надає змогу автоматично переносити файли з робочої папки на головній машині, на віртуальну. Це надає можливість автоматично розгорнути файли використовуючи вбудований функціонал Visual Studio, який називається File Share. Цей інструмент публікує усі файли застосунку в цільову папку з обраною конфігурацією (Release/Debug) та цільову архітектуру середовища (x86/x64/Portable). Після завершення розгортання усі файли будуть перенесені в цільову папку та будуть доступні на віртуальній машині. Перед початком тестування застосунку в робочому середовищі, потрібно доналаштувати веб-сервер. Спочатку необхідно створити конфігураційний файл, який буде збирати помилки при роботі застосунку та містити конфігурацію сервера, на який будуть надходити запити. Після створення конфігураційного файлу, необхідно додати файл з розширенням .service, який міститиме інформацію про робочий каталог та точку входу в застосунок. Після цього необхідно запустити новостворений сервіс та перезапустити сервер Apache для початку роботи застосунку. Коли усі налаштування завершені можна відкрити браузер та перевірити роботу застосунку за публічною ip адресою.

3.3 Налаштування інфраструктури

Для повноцінної роботи застосунку найкраще при запуску налаштувати усі необхідні сервіси. Для цього Vagrant через зарезервоване слово provision, на цьому етапі необхідно вказати провайдера сервісу забезпечення. Інструмент забезпечення Vagrant Shell дозволяє завантажувати та запускати скрипти на віртуальній машині. Ці скрипти можуть виконати описані вище кроки, як встановлення .NET, встановлення веб-серверу Apache та налаштування конфігурації застосунку.

4. Висновки

Результатом курсової роботи є розроблена тестова версія онлайн блогу для обміну знаннями в організаціях невеликого розміру, яка може розгортатись на віртуальну машину для тестування. Застосунок використовує Sql Server для роботи з базами даних та Vagrant для автоматичного налаштування робочої інфраструктури. За допомогою Vagrant можливо автоматично розгортати та

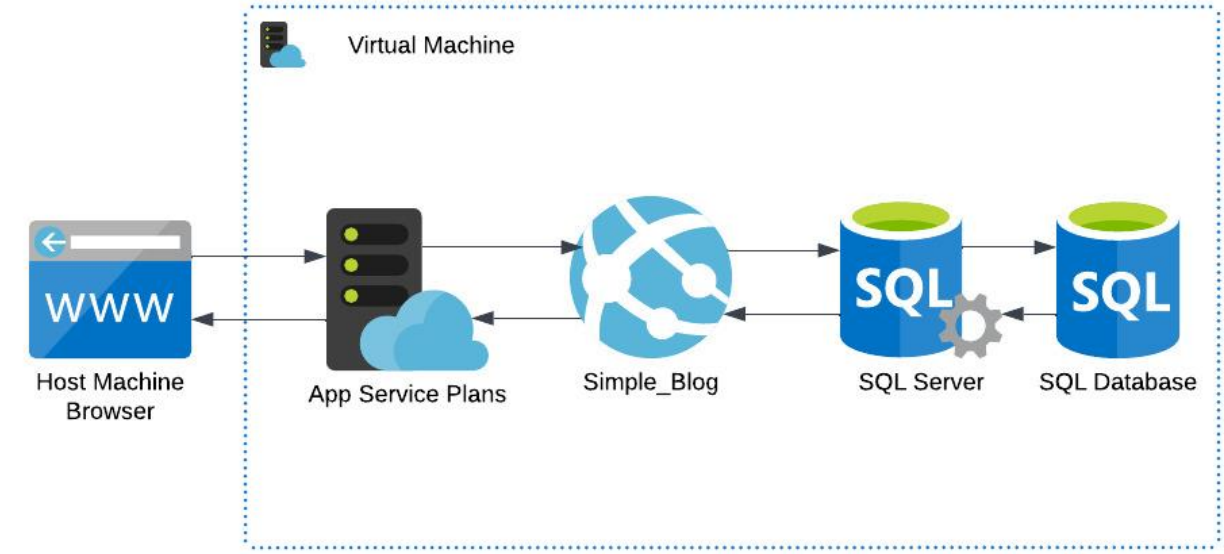
копіювати тестові середовища для максимальної імітації робочої інфраструктури. За допомогою логування можна відстежувати помилки, які стаються під час виконання. Було використано велику кількість технологій для розробки : ASP.NET Web App, Razor Pages, Entity Framework Core, Sql Server, Bootstrap, Dependency Injection. А також було використано інструменти та сервіси для автоматизації розгортання застосунку : Vagrant, Virtual Box, Visual Studio File Share, Shell, Apache. В рамках цієї роботи було розглянуто велику кількість сервісів, які можуть додатково покращити автоматизацію розгортання як на локальну віртуальну машину, так і в хмарне середовище, такі як : Chef, Puppet, Ansible. Робочий прототип онлайн блогу демонструє властивості сучасного підходу до розробки веб-застосунків а також тестування правильної роботи до потрапляння в робоче середовище. Застосунок складається з 2х робочих частин, кожна з яких розгортається на віртуальній машині та дозволяє в повній мірі перевірити роботу рішення.

5. Джерела

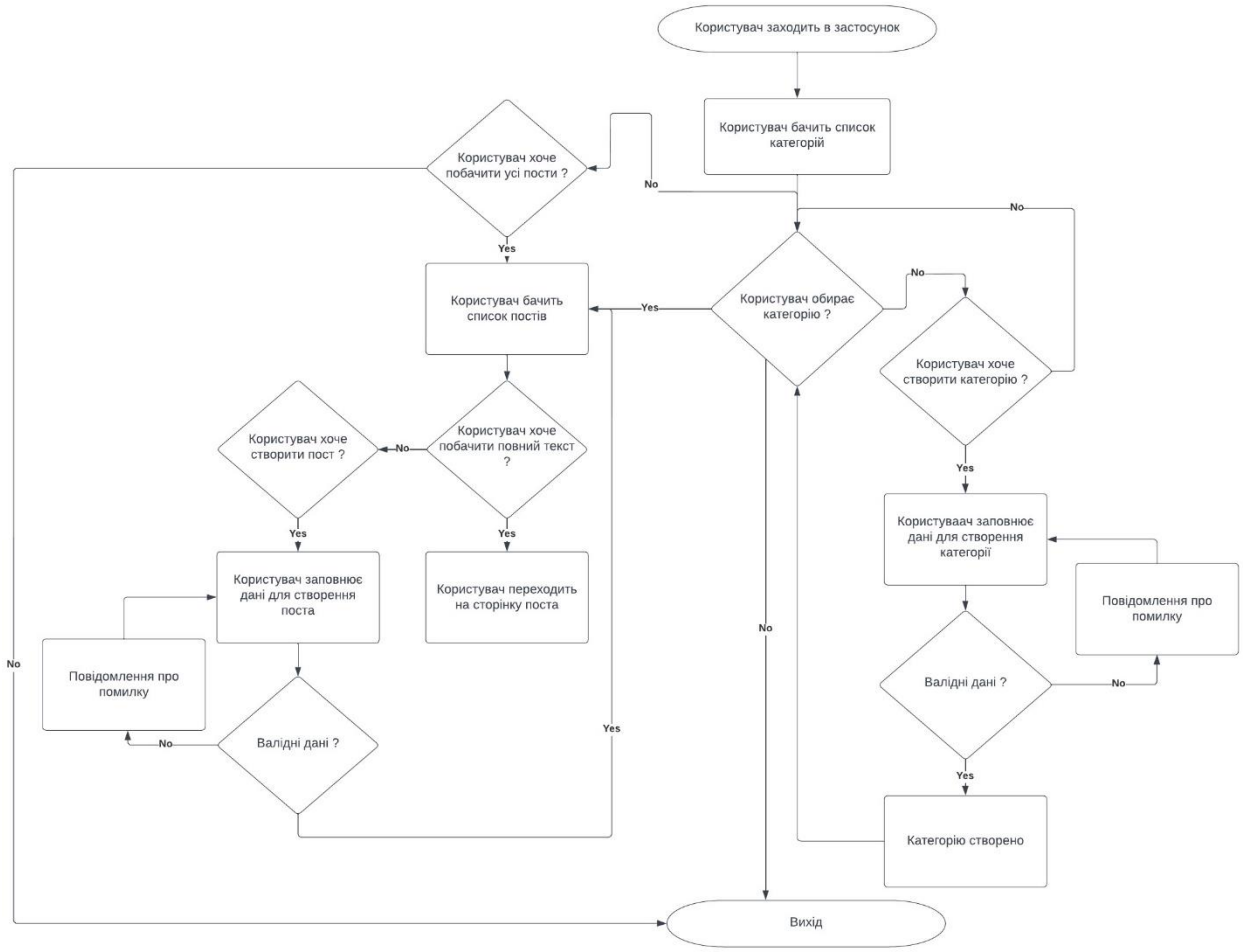
1. Vagrant Documentation - https://www.vagrantup.com/docs
2. Vagrant Information Page - https://www.vagrantup.com/intro
3. Ansible Automation Page - https://www.ansible.com/products/automation-platform?hsLang=en-us
4. Ansible Pricing - https://www.ansible.com/products/pricing
5. What is vRealizeAutomation? - https://www.parallels.com/blogs/ras/vrealize-automation/#:~:text=vRealize%20Automation%20is%20a%20DevOps,delivery%20of%20container%2Dbased%20applications.
6. Why Puppet? - https://puppet.com/why-puppet/
7. Fluent API - https://www.entityframeworktutorial.net/efcore/fluent-api-in-entity-framework-core.aspx
8. Vagrant Shell - https://www.vagrantup.com/docs/provisioning/shell

6. Додатки

Додаток 1 – Структурна схема застосунку

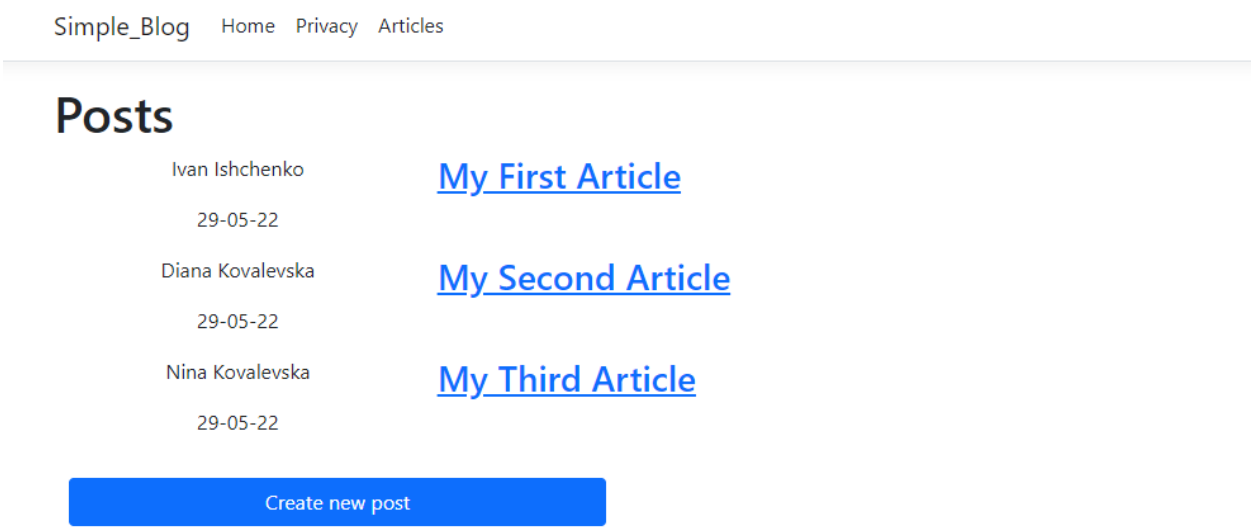


Додаток 2 – Блок-схема функціонування застосунку

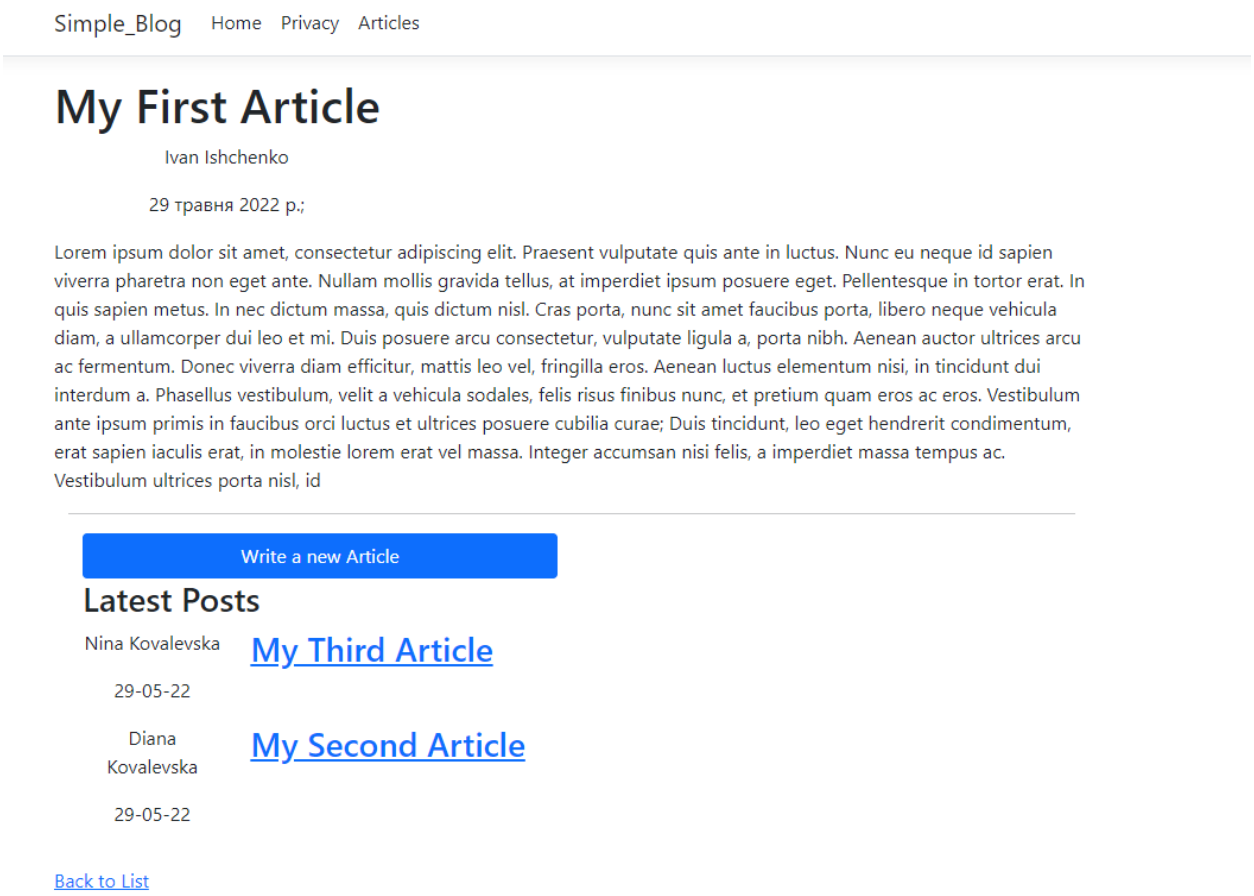


Додаток 3 – Сторінка перегляду категорій

Додаток 4 – Сторінка усіх постів



Додаток 5 – Сторінка огляду поста



Додаток 6 – Сторінка створення поста

Додаток 7 – автоматично згенерований Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/focal64"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
  config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine and only allow access
  # via 127.0.0.1 to disable public access
  config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network "private_network", ip: "192.168.33.10"

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
```

Додаток 8 – команди для встановлення .NET на віртуальну машину

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb

sudo dpkg -i packages-microsoft-prod.deb
sudo apt-get update
sudo apt-get install -y dotnet-sdk-6.0
```

Додаток 9 – робочий Vagrantfile

```
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/focal64"

  config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.network "public_network"

  config.vm.synced_folder "../data", "/vagrant_data"

  config.vm.provider "virtualbox" do |vb|
    vb.gui = true
    vb.memory = "1024"
  end

  config.vm.provision "shell", inline: <<-SHELL
    apt-get update

    wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-
prod.deb -O packages-microsoft-prod.deb

    sudo dpkg -i packages-microsoft-prod.deb

    sudo apt-get update
    sudo apt-get install -y apt-transport-https
    sudo apt-get update
    sudo apt-get install -y dotnet-sdk-6.0

    apt-get install -y apache2

    systemctl restart apache2
    a2enmod proxy proxy_http proxy_html
  SHELL

  config.vm.provision :shell, path: "vagrant_data/configure_application.sh"

  config.vm.provision :shell, path: "vagrant_data/configure_service.sh"

  config.vm.provision :shell, path: "vagrant_data/start.sh"
end
```