Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Створення компонент хмарного сервісу для вивчення STEM дисциплін

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення»- 12**1**

Керівник курсової роботи

докт. фіз-мат наук,

Малашонок Г.І.

_____

(підпис)

“____” _____ 2021 р.

Виконав студент ІПЗ-МП1:

Баранов Д.О.

“____” _____ 2021 р.

Київ 2021

Зміст

Вступ
Постановка задачі

Аналіз компонент хмарних застосуваннь та створення компоненнт хмарного застосування для вивчення STEM дисциплін.

STEM - Science (Наука), Technology (Технології), Engineering (Інженерія) та Mathematics (Математика).[1] Це нова методика отримання освіти коли навчання будується не навколо вчителя а навколо учня.

Учень отримує більше автономності ніж в звичайному навчанні. Крім того на навчання не впливають стосунки між вчителем і студентом. Учень більше вчиться самостійно вирішуючи проблеми.

Вимоги:

1. Створення компоненту акаунта
2. Створення компоненту курсу
3. Створення компоненту школи
4. Створення компоненту журнал
5. Створення компоненту GateWay

Розділ 1. Компонент хмарна архітектура

Причини використання

Традиційна розробка не встигає за стрімким ростом робочого навантаження. Від монолітної архітектури починають відмовлятись всі великі компанії перейшовши на мікросервісну архітектуру та контейнери.

Сучасний підхід – всі необхідні для роботи сервісу компоненти не розгортаються в віртуальному середовищі, а є частиною мікро сервісної архітектури.

Технології контейнерезації побудовані на вільном ПЗ. Найбільш популярний формат – Docker

Основні принципи контейнеризації [3]

Для ефективної роботи необхідно не просто створити образ контейнера і запустити його. Потрібно дотримуватися деяких стандартів.

1. 1 контейнер – 1 сервіс

   Контейнер повинен виконувити лише одну функції, не потрібно додавати в нього всі сутності від яких залежить додаток. Завдяки цьому принципу можно дуже добре маштабувати додаток.

2. Незмінність образа

   Всі зміни повинні вноситься на стадії розгортання образа. При використанні цього принципу можна гарантувати, не втрату даних при видаленні контейнеру.

3. Утилізація контейнерів

   Любий контейнер може в любий момент бути знищеним і замінений на інший без зупинки обслуговування. При виконанні цього

принципу означає що вихід контейнеру з строя не повинно бути новиною і ротація контейнерів повинна бити звичайною.

4. Звітність

   Контейнер повинен мати точки перевірка його готовності.

5. Управляємість

   Додаток в контейнеры повинен мати можливість взаємодіяти з контролюючим його процесом.

6. Самодостатність

   Образ повинен мати всі необхідні залежності для роботи – бібліотеками конфігами і прочим.

7. Лімітування

   Можливість лімітувати CPU та RAM.

Переваги [2]

1. Оптимізація, швидкість і маштабність.

   На один фізичний сервіс завдяки контейнерам вдається помістити в 2-3 раза більше додатків ніж при використанні віртуальної машини. Єдина платформа не потребує інтеграції додаткових компонентів, не потребує резервуації додаткових ресурсів. Оскільки всі компоненти запаковані в контейнери.

   Швидкість розготання одного контейнера від пари секунд до декількох хвилин, але це набагато швидше ніж розгортання звичайної віртуальної машини

Як наслідок, обслуговування великого числа звернень до системи на тому же об’єму «заліза».

2. Високий захист від відказів.

В кластері контейнери розташовані тонким слоєм, і коли якась нода вилітає то розрахунки переходять до наступної.

Кожен контейнер містть в собі повну среду. Додаток, бібліотеки, зависимі об’єкти, файли. За рахунок цього немає різниці на якій системі розгортати контейнер

3. Невеликий розмір

Кожен контейнер займає декілька десятків мегабайт, коли повноцінна система може займати величезні розміри.

4. Автономність

Кожен контейнер можна запустити окремо, наприклад для тестів.

Архітектура Docker

Docker - це засіб для запуску контейнерних додатків. Контейнерне додаток включає в себе додаток і файлову систему, складові середу, в якій воно виконується.

Це приблизна схема стандартної віртуалізації

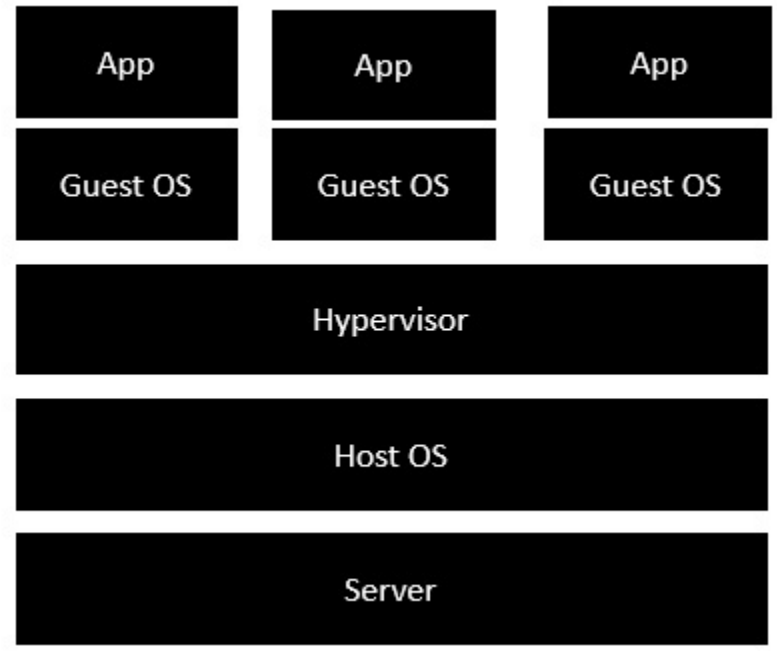


Рис. 1 Архітектура стандартної віртуалізації

На цій схемі після гіпервізора ми повинні установити декілька гостевих ОС, на яких будуть розгорнуті додатки

На наступній схемі та сама архітектура за допомогою Докера.

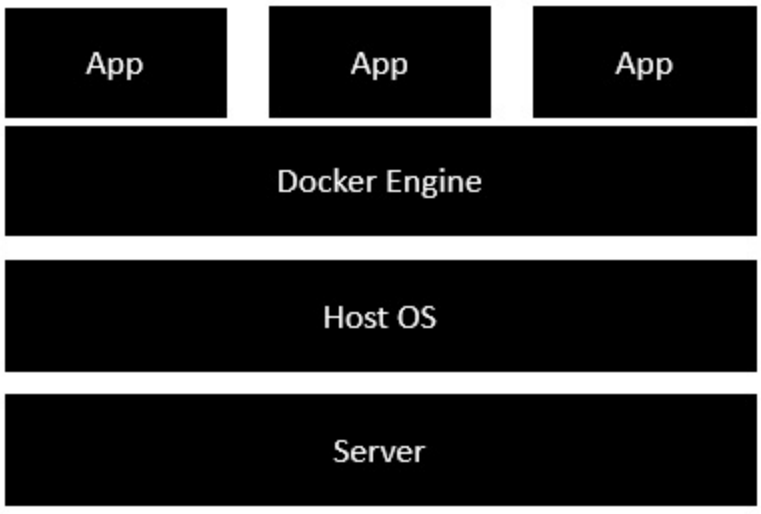


Рис. 2 Архітектура віртуалізації нового покоління

На цій схемі після HOST ОС, який являється базовою машиною, далі розгортається Docker, для запуску ОС, які раніше були віртальними машинами, в цьому випадку використовуються контейнери Докер.

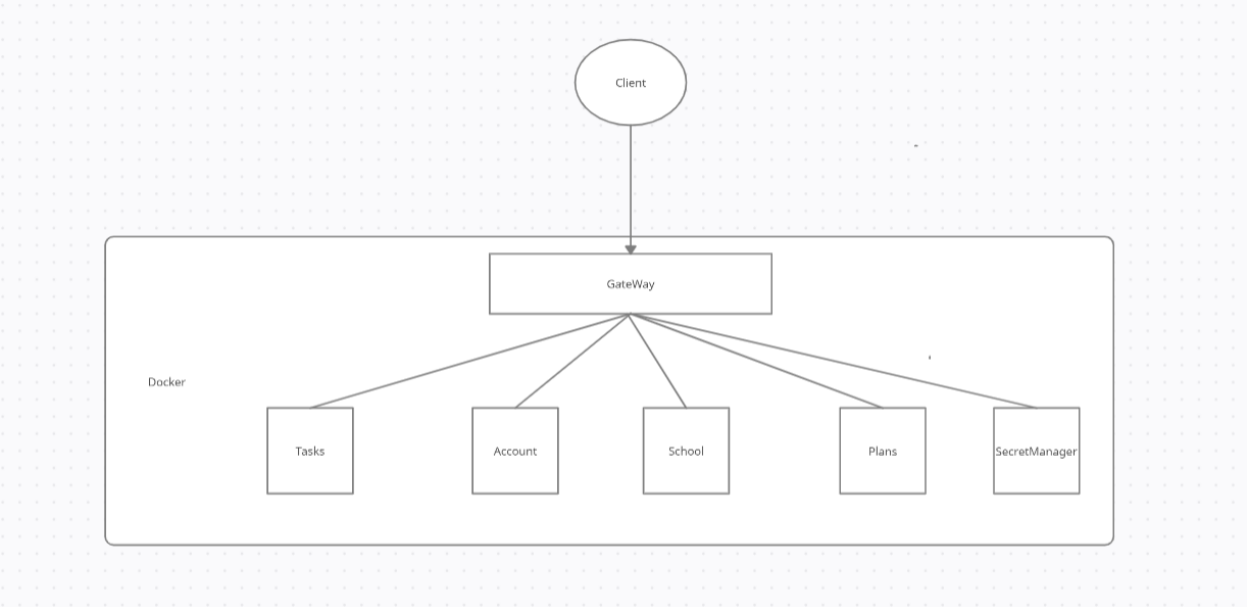
Docker-образ – це read-only шаблон. Образ може містити операційну систему наприклад Ubuntu. Образи використовуються для створення контейнерів.

Реєстр – Docker-реэстр зберігає образи.

Контейнер – як говорилось вище, щось схоже на директорію. В контейнері зберігається все що потрібно для роботи додатку

Розділ 2. Специфікація сервісу для вивчення STEM дисциплін

Схема та короткий опис



Client – написаний за допомогою Vue фреймворка, забеспечує взаємодію з сервером.

GateWay – вузол який об’єднує всі компоненти.

Tasks – збереження завданнь в системі.

Account – сервіс підтримки облікових записів.

School – містить інформацію про навчальний заклад.

Plans – містить інформацію про курсы.

SecretManager – налаштування, обмін повідомленнями.

База даних - таблиці баз даних створюються та управляються за допомогою сценаріїв у ddl папці ресурсів. Класи сутності перевіряють таблиці лише для того, щоб бути оновленими з кодовою базою.

Компонент Account

Задачі та цілі компоненту - головна мета це забезпечити ведення облікових записів. Реєстрація та вхід до системи.

Сервер – за допомогою спрінг. Взаємодія з базою за допомогою JpaRepository та Lombok.

Детальний опис

Загальна структура. У компоненті 6 пакетів:

_configs - це пакет, відповідальний за будь-яку конфігурацію та початкові класи.

Містить – в собі конфігурацію для БД, відправки повідомлень. Також клас для запуску серверу.

controllers – класи з mapping для створення видалення та іншої взаємодії облікових записів

entities – містять в собі таблицю класи для створення таблиць Account, AuthenticationToken, ChangePasswordToken.

repositories - це пакет зі сховищами даних Spring для роботи з базою даних

services – містить класи для роботи з логічною частиною додатку. Наприклад перевірка доступності поштової скриньки.

utils - це пакет для будь-яких службових програм, таких як DTO, властивості, спеціальні винятки, константи.

Компонент School

Задачі та цілі компоненту – створення запису навчального закладу. Запис вчителя до групи. Запис учнів до групи. Підтримка.

Сервер – за допомогою спрінг. Взаємодія з базою за допомогою JpaRepository та Lombok.

Детальний опис

Загальна структура. У компоненті 6 пакетів:

_configs - це пакет, відповідальний за будь-яку конфігурацію та початкові класи.

Містить – в собі конфігурацію для БД, відправки повідомлень. Також клас для запуску серверу.

controllers - класи які реалізовують Get/Post Mapping сервісу.

- ClassController – створення класу, додавання студентів, а також видалення, редагування.
- GroupController – створення, видалення групи. Додавання студентів до групи. Присвоювання вчителя до групи.
- SchoolController – створення та видалення шкіл.

entities – класи для створення таблиць бази даних.

- School
- SchoolAddress
- SchoolClass
- SchoolGroup
- UserProfile

repositories - це пакет зі сховищами даних Spring для роботи з базою даних

services - класи що реалізовують всі логічні функції додатку.

utils – службові функції.

Компонент Tasks

Задачі та цілі компоненту – створення завданнь. Взаэмодія з ними.

Сервер – за допомогою спрінг. Взаємодія з базою за допомогою JpaRepository та Lombok.

Детальний опис

Загальна структура. У компоненті 6 пакетів:

_configs – конфігурація додатку

controllers – створення завдання і обробка відповідей.

entities – таблиця для збереження завданнь

repositories - це пакет зі сховищами даних Spring для роботи з базою даних

services пакет містить усі класи, що реалізують логічні компоненти модуля

utils – службові функції та константи.

Компонент SecretManager

Компонент забеспечує взаємодія з школаю, акаунтами та налаштування.

Детальний опис

Configuration – налаштування сервісу.

controllers – контролер для налаштування.

Services – логіка сервісу.

Висновок

Навчання за допомогую STEM освіти має багато переваг серед яких: інтегроване навчання за «темами», а не з предметів, підвищення впевненості до своїх сил, автономність та інші.

Раніше не було зрозуміло чи використання контейнерів необхідність для продакшена. Але індустрія не стоїть на місці та оцінила їх продуктивність. На сьогодні використання контейнерів – дешевий та швидкий спосіб задеплоїти додаток.

З кожним днем доля ринку контейнерів зростає в порівнянні з ВМ.

Перспективи такого підходу також доволі гарні:

1. Оптимізація
2. Швидкість
3. Маштабованість
4. Розмір
5. Захист від поломок

Що дозволяє створити продукт з стабільною роботою.

Список літератури:

1. https://hvylya.net/analytics/society/maybutnye-ukrayinskoyi-molodi-stem-osvita.html
2. https://www.iksmedia.ru/articles/5532413-Kontejnery-kak-novyj-oblachnyj-tren.html
3. https://tproger.ru/articles/containers-explained/
4. https://habr.com/ru/post/253877/