Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Хмарна система MathLearning на базі MathPartner

Текстова частина до дипломної роботи
за спеціальністю „Програмна інженерія”

Керівник дипломної роботи
д.ф.-м.н., проф. Малашонок Г.І.

_____

(підпис)

“____” _____ 2023 р.

Виконав студент

_____ _

(прізвище та ініціали)

“____” _____ 2023 р.

Київ 2023

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

к.ф-м.н.

_____ С. С. Гороховський

13 листопада 2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на дипломну роботу

студенту 2-го курсу ІПЗ

Саху Роману Ігоровичу

на тему:

Хмарна система MathLearning на базі MathPartner

Зміст Текстової частини до магістерської роботи:

1. Вступ

2. Застосування технологій і практик хмарної розробки в навчальному процесі

    2.1.Мікросервісна архітектура для навчальних ресурсів

    2.2.Застосування підходу Blue/Green deployment для інфраструктури

    2.3.Впровадження процесу неперервної інтеграції/поставки для розробки

    2.4.Приклад імплементації open source проекту Mathpar Learning

3. Перспективи застосування Mathpar Learning для шкільної освіти в Україні.

    3.1.Використання Mathpartner як обчислювального модуля

    3.1.1. Створення навчальних ресурсів з використанням синтаксису LaTex

        A. Імпортування стандартизованих національних підручників.

        B. Створення підручника викладачем.

    3.2. Розробка самостійних і контрольних робіт, із прикладами вирішення та підказками.

3.3. Використання Mathpartner, як робочого зошита для проведення складних обчислень.

3.4. Використання можливостей Mathpartner, для перевірки правильності відповіді учня або демонстрації правильного рішення.

3.2.Створення бази задач та завдань для використання в освітньому процесі

3.2.3.

A. Створення бази навчальних планів дисциплін для різних рівнів складності.

B. Створення учителем бази планів проходження навчальної дисципліни для навчальної групи.

3.2.4. Створення і підтримання бази щоденників учнів із збереженням актуальної інформації.

3.2.5. Перспективи організації моніторингу успішності учнів в межах України на основі бази "щоденників".

3.3.Застосування хмарних обрахунків в процес опанування математики, фізики, хімії

4. Висновки

4.1. Розміщення Інформації про проект і вихідного коду в загальному доступі.

4.2.Аналіз подібних навчальних платформ.

4.3. Перспективи розвитку проекту.

5. Список літератури

6. Додатки

Дата видачі 26 жовтня 2022 р. Керівник

_____

Завдання отримав _____

Календарний план виконання дипломної роботи

Тема: Хмарна система управління освітнім процесом

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	26.10.2022	
2.	Огляд технічної літератури за темою роботи.	15.11.2022	
3.	Підготовка технічного завдання та технічної документації для веб застосунку.	10.12.2022	
3.	Розробка мікросервісної архітектури для застосунку.	12.12.2022	
4.	Проектування CI/CD процесу для застосунку.	20.01. 2022	
5.	Реалізація сервісів застосунку.	13.03.2023	
6.	Реалізація CI/CD процесу для застосунку.	02.04. 2023	
7.	Створення інструментів для проведення автоматичного тестування.	15.04. 2023	

8.	Створення документації для вихідного коду застосунку.	24.04. 2023	
8.	Аналіз отриманих результатів.	01.05. 2023	
10.	Корегування роботи за результатами попереднього захисту.	15.05. 2023	
11.	Остаточне оформлення пояснювальної роботи та слайдів.	06.06. 2023	
12.	Захист магістерської роботи (проекту)	14.06. 2023	

Студент _____

Керівник _____ “_____”_____

Зміст

Анотація

Використання мікросервісної архітектури та хмарних технологій, зокрема обчислень, має значні переваги для освітніх проектів на довгий термін. У ході даної роботи було досліджено основні переваги цих підходів і розроблено освітній проект під назвою Mathpar Learning, що використовує зазначені технології.

Проект Mathpar Learning передбачає розробку електронних навчальних матеріалів з математики, фізики та хімії. Під час створення проекту були розглянуті різні варіанти подання навчального матеріалу у цифровому форматі. Для цього була використана існуюча бібліотека електронних навчальних матеріалів.

Окрім того, в проекті було реалізовано інтеграцію хмарних обчислень, зокрема системи Mathpartner, що використовується як зовнішня залежність проекту. Це дозволяє студентам та викладачам зручно використовувати хмарні обчислення для виконання складних математичних розрахунків та моделювання на основі реальних даних.

Робота над проектом Mathpar Learning показала, що використання мікросервісної архітектури та хмарних технологій сприяє покращенню освітнього процесу, розширює можливості доступу до навчальних матеріалів та забезпечує ефективне використання обчислювальних ресурсів.

Вступ

Завдяки стрімкому розвитку технологій у сучасному світі, освітній процес стає вимогливішим, оскільки важко встигнути застосовувати нові досягнення. Однак розвиток хмарних технологій може значно спростити і здешевити розробку та підтримку освітніх ресурсів, якщо їх правильно використовувати. Використання мікросервісної архітектури дозволяє розбити складні навчальні програми на окремі сервіси, що спрощує загальну підтримку та розробку програми. Використання хмарних провайдерів полегшує процес забезпечення інфраструктури за допомогою підходу "Інфраструктура як код" (IaaC) і зменшує витрати. Впровадження хмарних обчислень в освітній процес також має багато переваг, починаючи з інтерактивних лекційних матеріалів і закінчуючи гнучкими пісочницями для експериментів та випробування власних рішень поза шкільними матеріалами.

Головною метою цієї роботи є створення комплексу доступних матеріалів та ресурсів, що спростять впровадження вищезгаданих технологій у освітній процес. Для досягнення цієї мети необхідно провести аналіз особливостей архітектури хмарних технологій, систематизувати та використати базові підходи до розробки мікросервісних додатків. Також важливим кроком є розробка плану інтеграції хмарних обчислень у навчальну систему, включаючи опис способу роботи з системою та її гнучкість у підтримці та використанні. Не менш важливим завданням є надання рекомендацій та практичних прикладів використання сучасних підходів до реалізації неперервної інтеграції та неперервної доставки (CI/CD).

Для проведення аналізу було використано різноманітні джерела, включаючи ключові роботи та вебінари, які проводилися командами провідних хмарних провайдерів, таких як Amazon Web Services (AWS) і Microsoft Azure.

Ці ресурси були вибрані через їхню популярність і вплив на сучасну хмарну технологію. Крім того, було проаналізовано навчальні матеріали та посібники, які надає сервіс Mathpar, який використовується як провайдер хмарних обчислень.

Після завершення роботи були розроблені матеріали з рекомендаціями для кожного з описаних завдань. Крім того, було створено програмний продукт з відкритим кодом, який втілює ці рекомендації і є прикладом готової до використання навчальної системи. Кожен компонент системи має детальну технічну документацію і відповідає автоматизованим тестам. Демо-проект також був розгорнутий, включаючи імпортовану базу матеріалів та кілька шаблонів навчальних планів, щоб надати можливість тестування основних функцій системи.

Результати роботи можуть слугувати фундаментом для подальшого розвитку даного проекту і розробки національної платформи онлайн освіти у сфері STEM. Цей проект може бути потенційно цікавим великій IT компанії, яка має напрямок роботи Corporate Social Responsibility.

Глава 1. Застосування технологій і практик хмарної розробки в навчальному процесі

1.1. Мікросервісна архітектура для навчальних ресурсів

Мікросервісна архітектура - це підхід до розробки програмного забезпечення, при якому додаток розбивається на невеликі і незалежні компоненти, відомі як мікросервіси. Кожен мікросервіс представляє собою окремо працюючу функціональність з власним контекстом та інтерфейсом програмування додатків (API).

Основна ідея мікросервісної архітектури полягає в тому, щоб розділити складний додаток на більш прості і легко керовані компоненти. Кожен мікросервіс може бути розроблений, масштабований, випускатись та керуватись незалежно від інших мікросервісів. Кожен сервіс може бути написаний мовою програмування та використовувати технології, що найкраще підходять для його конкретного завдання.

Мікросервісна архітектура дозволяє забезпечити гнучкість, масштабованість та швидкість розробки програмного забезпечення. Кожен мікросервіс може бути розгорнутий окремо, розвиватись незалежно та використовувати різні технології і засоби зв'язку, такі як RESTful API або повідомлення.

Однак, мікросервісна архітектура також може створювати складність у керуванні дистрибуцією, моніторингу та керуванні версіями сервісів. Також потрібна надійна мережева комунікація між мікросервісами для забезпечення їх взаємодії.

Узагальнюючи, мікросервісна архітектура пропонує підхід до розробки додатків, в якому вони розбиваються на малий набір незалежних

При проектуванні мікросервісних застосунків існує кілька основних підходів, які можуть бути використані. Ось декілька з них:

1. Розбиття на мікросервіси на основі функціональності: В цьому підході застосунок розбивається на мікросервіси відповідно до функціональних областей. Кожен мікросервіс відповідає за конкретну функцію або бізнес-процес. Наприклад, можна мати окремі мікросервіси для аутентифікації користувачів, обробки платежів, керування замовленнями тощо. Цей підхід дозволяє розподілити відповідальність між різними командами розробників і забезпечити легку масштабованість окремих функцій.

2. Розбиття на мікросервіси на основі домену: У цьому підході застосунок розбивається на мікросервіси відповідно до окремих доменів або сфер бізнесу. Кожен мікросервіс фокусується на конкретній сфері бізнесу і виконує всі необхідні функції для цього домену. Наприклад, можна мати мікросервіси для керування клієнтами, інвентарем, відправленнями тощо. Цей підхід дозволяє краще організувати роботу команд, які відповідають за конкретні домени.

3. Розбиття на мікросервіси на основі данних: В цьому підході застосунок розбивається на мікросервіси відповідно до виділення окремих даних або сервісів у своїй власній базі даних. Кожен мікросервіс має свою власну базу даних і виконує операції

Поруч з перевагами мікросервісної архітектури є і її обмеження, які потрібно враховувати при розробці. Одним із таких прикладів є забезпечення зворотної сумісності і версійності.

Версіонування мікросервісів - це процес управління версіями та їхнім життєвим циклом в мікросервісній архітектурі. Оскільки мікросервіси є незалежними компонентами, кожен мікросервіс може мати свою власну версію, яка може розвиватися незалежно від інших мікросервісів у системі.

Основні аспекти версіонування мікросервісів:

1. Версійність API: Кожен мікросервіс може мати своє API, і зміни в API можуть призвести до несумісності між версіями. Для керування версіями API можна використовувати нумерування версій, наприклад, через URL-шлях або заголовок запиту. Також можуть використовуватися підхід "підтримка попередньої версії" (backward compatibility), де нові версії додають нові функції, не змінюючи старі.

2. Релізи та деплоймент: Кожна версія мікросервісу може мати свій життєвий цикл релізу та деплойменту. Нові версії можуть бути розгорнуті паралельно зі старими версіями, або старі версії можуть бути відключені після успішного впровадження нових. Рішення про розгортання нових версій може залежати від бізнес-потреб, технічних обмежень та стратегій розробника.

3. Керування залежностями: У мікросервісній архітектурі можуть існувати залежності між різними мікросервісами. При внесенні змін в один мікросервіс можуть виникати вимоги до змін в інших мікросервісах. Управління залежностями і вирішення конфліктів версій є важливими аспектами версіонування мікросервісів.

Версіонування мікросервісів є складним завданням, оскільки вимагає уваги до деталей та хорошої організації. Правильне керування версіями може забезпечити стабільність та сумісність між різними мікросервісами, дозволяючи ефективно розвивати та масштабувати систему.

У мікросервісній архітектурі кожен сервіс може мати свою власну окрему інфраструктуру. Це означає, що кожен сервіс може мати власні ресурси, середовище виконання та інструменти для розгортання і управління ним.

Основні аспекти окремої інфраструктури для кожного мікросервісу:

1. Ресурси: Кожен мікросервіс може мати власну набор ресурсів, таких як сервери, обчислювальні ресурси, бази даних тощо. Це дозволяє кожному сервісу масштабуватися незалежно від інших і використовувати тільки необхідні ресурси.

2. Середовище виконання: Кожен мікросервіс може використовувати своє власне середовище виконання, таке як платформа або мова програмування. Це дозволяє командам розробників використовувати ті технології, які найкраще підходять для їхніх потреб і вимог.

3. Інструменти розгортання і управління: Кожен мікросервіс може мати свої власні інструменти для розгортання, моніторингу, логування та керування. Наприклад, для автоматизації розгортання можуть використовуватися інструменти, такі як Docker або Kubernetes. Для моніторингу можуть використовуватися спеціалізовані системи моніторингу, такі як Prometheus або ELK Stack.

4. Комунікація та взаємодія: Мікросервіси можуть спілкуватися між собою за допомогою протоколів комунікації, таких як HTTP, REST або механізми передачі повідомлень, такі як RabbitMQ або Apache Kafka. Кожен мікросервіс може мати свою власну систему комунікації, адаптовану до його потреб.

Ця окрема інфраструктура для кожного мікросервісу дозволяє командам розробників працювати незалежно один від одного, масштабувати окремі

компоненти системи та підтримувати гнучкість і готовність до змін. Однак, вона також вимагає додаткового управління та координації для забезпечення взаємодії між сервісами та загальної ефективності системи.

1.2 Застосування підходу Blue/Green deployment для інфраструктури

Blue/Green deployment - це підхід до впровадження програмного забезпечення, що дозволяє розгортати нову версію програмного забезпечення на окремому середовищі (Green), перевіряти його працездатність і потім переключати трафік на нову версію, забезпечуючи безперервну роботу системи. Існуюча версія програмного забезпечення продовжує працювати на окремому середовищі (Blue), щоб забезпечити можливість швидкого повернення до попередньої версії, якщо виявляться проблеми.

Інфраструктура мікросервісів може бути розгорнута в декількох екземплярах, де кожен мікросервіс працює в окремому контейнері або віртуальній машині. Blue/Green deployment може бути застосований до інфраструктури мікросервісів, щоб забезпечити безперервну доставку та розгортання нових версій мікросервісів без впливу на роботу системи.

Основні кроки використання підходу Blue/Green deployment для інфраструктури мікросервісів:

1. Підготовка середовищ Green: Створення нових контейнерів або віртуальних машин для розгортання нової версії мікросервісів. Це може включати створення нових образів контейнерів або віртуальних машин, налаштування необхідних залежностей та конфігурацію середовища.

2. Тестування середовища Green: Виконання необхідних тестів на середовищі Green, щоб переконатися в його працездатності. Це може включати функціональне тестування, інтеграційне тестування та навантажувальне тестування.

3. Переключення трафіку: Після успішного пройдення тестів, трафік починає перенаправлятись з існуючого середовища Blue на нове середовище Green. Це може бути зроблено за допомогою балансувальника навантаження або маршрутизатора, які направляють запити на нове середовище.

4. Моніторинг: Слідкування за роботою нового середовища Green, аналіз метрик та логів, щоб впевнитися, що воно працює належним чином. Якщо виявляться проблеми, можна швидко повернутися до попередньої версії, переключивши трафік на середовище Blue.

5. Видалення старого середовища: Після переконливого успіху нового середовища Green, можна видалити старе середовище Blue. Це може включати зупинення контейнерів або віртуальних машин, які працюють в старому середовищі.

Підхід Blue/Green deployment дозволяє знизити ризики при впровадженні нових версій мікросервісів, забезпечує швидке відновлення в разі проблем та дозволяє забезпечити безперервну роботу системи. Він є популярним в інфраструктурі мікросервісів, де швидкість впровадження та надійність є важливими факторами.

B/G (Blue/Green) підхід для розгортання сервісів є одним із популярних методів, які дозволяють впроваджувати зміни в продуктивне середовище з

мінімальними впливами на користувачів. Під час розробки сервісу, наступні обмеження B/G підходу були враховані:

1. Відсутність станів сервісу: B/G підхід передбачає наявність двох окремих середовищ для розгортання - "синього" (blue) та "зеленого" (green). Сервіси не повинні зберігати стан між цими середовищами, оскільки кожне середовище вважається окремим ізольованим інстансом.

2. Консистентність відкладених завдань: У B/G підході потрібно забезпечити, щоб будь-які відкладені або асинхронні завдання, що виконуються в "синьому" середовищі, також були налаштовані для виконання в "зеленому" середовищі. Це гарантує консистентність операцій та уникнення втрати даних під час перемикання між середовищами.

3. Контроль фонових запланованих завдань: При перемиканні між середовищами, необхідно забезпечити, щоб фонові або заплановані завдання (наприклад, резервне копіювання, обробка черги повідомлень тощо) були вірно настроєні та виконувалися безперебійно в обох середовищах.

4. Версіювання будь-яких змін в видимих назовні частинах функціоналу: B/G підхід дозволяє впроваджувати новий функціонал шляхом включення "зеленого" середовища у виробничий потік даних. Зміни повинні бути версіоновані та управлятися таким чином, щоб забезпечити сумісність між версіями та можливість переключення між ними.

5. Імплементація логіки відновлення бази даних до попереднього стану (roll-back скрипти): Якщо впровадження нової версії сервісу призводить до проблем або помилок, необхідно мати механізм відновлення бази даних до попереднього стану. Roll-back скрипти дозволяють повернутися до

попередньої стабільної версії, якщо щось пішло не так під час розгортання нової версії.

Ці обмеження допомагають забезпечити безперебійність та надійність при впровадженні змін в продуктивне середовище за допомогою B/G підходу.

1.3 Впровадження процесу неперервної інтеграції/поставки для розробки

CI/CD (Continuous Integration/Continuous Deployment) - це підхід до розробки програмного забезпечення, який дозволяє автоматизувати процеси інтеграції, тестування та розгортання програмних змін. Використання CI/CD дозволяє розробникам швидко і безпечно впроваджувати нові функції та виправлення помилок в продукцію.

Основні етапи CI/CD процесу для розробки включають наступні кроки:

1. Керування версіями: Використовуйте систему керування версіями, таку як Git, для збереження коду і ведення історії змін.

2. Інтеграція: Регулярно об'єднюйте код розробників у спільну гілку (наприклад, "develop" або "main") за допомогою механізму інтеграції, наприклад, Git merge або Git rebase. Це дозволяє виявляти конфлікти злиття раніше і забезпечує стабільну базову гілку для подальшої роботи.

3. Автоматична збірка: Створюйте автоматичну систему збирання (build system), яка збирає програмний код використовуючи засоби, такі як Maven або Gradle. Це дозволяє перевірити код на синтаксичні помилки та залежності, що допомагає уникнути проблем під час розгортання.

4. Автоматичні тести: Розробляйте автоматичні тести (unit tests, integration tests, end-to-end tests) для перевірки функціональності та якості програмного забезпечення. Ці тести повинні запускатися автоматично при кожній зміні коду і допомагати виявляти помилки на ранніх етапах розробки.

5. Контейнеризація: Використовуйте технології контейнеризації, такі як Docker, для створення відокремлених середовищ розгортання. Це дозволяє забезпечити консистентність середовища між розробкою і продуктивним середовищем.

6. Автоматичне розгортання: Налаштуйте автоматичне розгортання (deployment) програмного коду в середовище продукції після успішного проходження всіх тестів. Це може бути здійснено за допомогою інструментів, таких як Jenkins, GitLab CI/CD, або інші.

7. Моніторинг: Встановіть систему моніторингу, яка дозволяє відстежувати стан програмного забезпечення в продукції. Це допоможе виявляти проблеми та відслідковувати ефективність системи.

Ці етапи можуть бути додатково розширені або адаптовані відповідно до потреб вашого проекту або команди розробників. Використання інструментів автоматизації і контейнеризації допомагає значно прискорити процес розробки та зменшити ризики при впровадженні змін у продукцію.

Постійна інтеграція (CI – Continuous integration) полягає в практиці регулярного об'єднання (інтеграції) коду розробників у спільну гілку репозиторію, щоб виявити інтеграційні помилки і конфлікти якомога раніше. Основна мета CI - забезпечити швидку зворотну зв'язок щодо стану коду та виявлення проблем, що допомагає забезпечити стабільну та надійну основу для розробки програмного забезпечення.

Основні принципи постійної інтеграції включають:

1. Часті коміти: Розробники регулярно комітять свій код у спільний репозиторій. Кожен коміт викликає процес автоматичної збірки та тестування.

2. Автоматична збірка: При кожному коміті запускається автоматична процедура збірки, яка компілює та будує програмний код у виконуваний формат. Це дозволяє перевірити синтаксичні помилки та залежності.

3. Автоматичні тести: Разом з процесом збірки запускаються автоматичні тести (наприклад, юніт-тести), які перевіряють функціональність та коректність роботи програмного коду. Це допомагає виявляти помилки рано в процесі розробки.

4. Швидкий зворотний зв'язок: Розробники отримують швидкий зворотний зв'язок про стан коду після кожного коміту. Якщо виникають проблеми або помилки, розробники можуть вчасно їх виправити.

5. Автоматична ретроспектива: Після кожного успішного коміту може запускатись процес автоматичної ретроспективи, який аналізує статистику тестів, покриття коду тощо. Це допомагає виявляти тенденції та покращувати якість розробки.

6. Управління конфліктами: При об'єднанні різних гілок коду CI може виявляти конфлікти та проблеми злиття. Це дозволяє вчасно їх вирішувати та забезпечує стабільну роботу спільного кодової бази.

Використання постійної інтеграції сприяє покращенню командної співпраці, ранньому виявленню помилок та швидкому впровадженню змін.

1.4 Приклад імплементації open source проекту Mathpar Learning

Mathpar learning – це навчальний open-source проект, який реалізовує всі підходи описані раніше в роботі а також використовує зовнішній сервіс для виконання хмарних обрахунків про що буде описано далі.

Основні ідеї проекту:

Для вчителів:

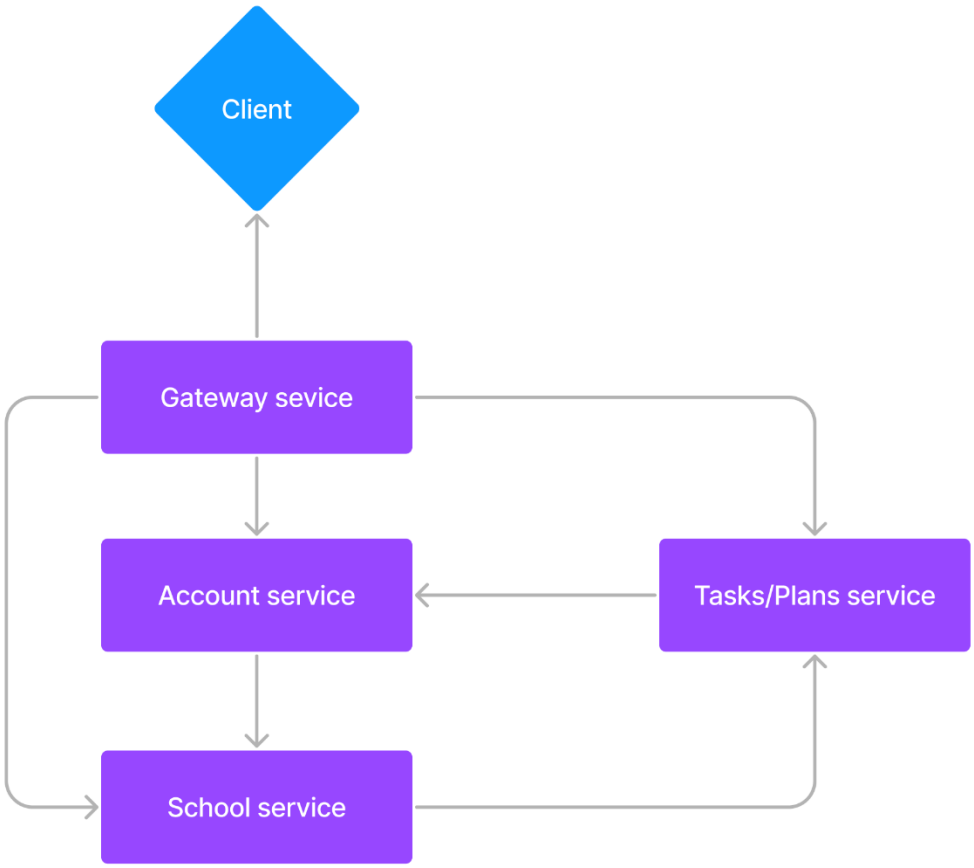
1. Електронний облік студентів: Сервіс може дозволяти вчителям зберігати та оновлювати інформацію про студентів, включаючи особисті дані, контактну інформацію, успішність тощо.
2. Навчальні групи та перерозподіл студентів: Вчителі можуть створювати групи студентів і перерозподіляти їх між ними в залежності від потреб навчання або інших факторів.
3. Навчальні плани: Сервіс може дозволяти вчителям створювати та налаштовувати навчальні плани для різних навчальних груп з урахуванням навчальних програм та вимог.
4. Дистанційні практичні та контрольні роботи: Сервіс може надавати засоби для проведення дистанційних практичних завдань та контрольних робіт, включаючи надання завдань, збір відповідей та оцінювання результатів.
5. Викладка успішності учнів: Сервіс може забезпечувати швидку та компактну інформацію про успішність учнів, таку як середній бал, оцінки за окремі завдання або предмети.

Для студентів:

1. Прив'язка до кількох навчальних закладів: Студенти можуть мати можливість бути зареєстрованими в декількох навчальних закладах та зберігати історію свого навчання при зміні шкіл, ВНЗів тощо.
2. Перегляд інтерактивних лекційних матеріалів: Сервіс може дозволяти студентам переглядати записані лекції, навчальні матеріали та ресурси в будь-який зручний для них момент часу.
3. Тренування в практичних завданнях: Студентам можуть надаватись практичні завдання, які вони можуть виконувати необмежену кількість разів для тренування та вдосконалення своїх навичок.
4. Екзаменаційні роботи у відведений час: Сервіс може забезпечувати можливість студентам проходити екзаменаційні роботи або тести відповідно до встановленого графіку або дедлайну.
5. Перегляд успіхів: Студенти можуть переглядати свої успіхи, оцінки за завдання та предмети, а також відстежувати свій прогрес у навчанні.

Застосунок Mathpar Learning може використовувати мікросервісну архітектуру для реалізації цих ідей, що означає розбиття функціональності на окремі сервіси, що взаємодіють між собою. Кожен сервіс може бути відповідальним за конкретну функцію, наприклад, облік студентів, управління групами, проведення тестів тощо.

Для досягнення мети було створено наступні сервіси:



Модуль "**Client**" відповідає за взаємодію користувачів з клієнтським додатком у рамках проекту Mathpar Learning, який використовує RESTful застосунок. Для забезпечення цієї взаємодії використовується фреймворк Vue.js. Модуль "Client" розробляється незалежно від серверної частини проекту і має свій власний репозиторій. Фреймворк Vue.js дозволяє зберегти всі розроблені компоненти у вигляді HTML, JavaScript і CSS файлів, які можуть бути використані в браузері.

Вихідний код даного модуля доступний за покликанням https://gitlab.com/mathparlearning/clientdev

Модуль "**Gateway Service**" відповідає за розподіл статичних ресурсів і клієнтського додатку. Його головною метою є централізація розташування ресурсних файлів для забезпечення їх окремого масштабування, захисту і шифрування, якщо це необхідно. Якщо архітектура клієнтського додатку зміниться на багатокомпонентну, модуль "Gateway Service" буде відповідати за розподіл кожного з компонентів. Цей модуль може бути реалізований за допомогою будь-якої мови програмування або навіть менеджера навантаження. Для спрощення і забезпечення консистентності в проекті Mathpar Learning використовується Java (Spring Framework) для реалізації цього модуля. Вихідний код даного модуля доступний за посиланням: https://gitlab.com/mathparlearning/gatewaydev

Account service - це компонент, який відповідає за обробку аккаунтів користувачів. Кожен користувач повинен створити свій власний аккаунт, який служитиме його ідентифікаційною карткою для різних дій. Ці активності можуть включати шкільні функції, такі як викладання вчителем, адміністрування школи або виконання завдань учнем, а також сервісні функції, такі як створення та редагування шаблонів завдань, баз підручників, розробка та зміна планів занять тощо. Крім того, сервіс аккаунтів відповідає за будь-які модифікації аккаунту, такі як відновлення пароля, зміна інформації та безпосереднє видалення аккаунту. При видаленні аккаунту його місце зберігається порожнім, щоб уникнути випадкового створення нового аккаунту залежностями в інших сервісах. Значить, фактично видалення аккаунту означає лише видалення приватної (PII) інформації.

School service - це компонент, який відповідає за логіку управління структурними елементами школи. В його функції входить: управління користувачами (шкільними профілями аккаунтів), авторизація користувачів (визначення ролей), управління класами студентів, управління групами студентів, прив'язка навчальних планів (які контролюються модулем навчальних планів) до груп, збереження прогресу навчання студентів. Шкільний сервіс не працює безпосередньо з аккаунтами, але створює профілі користувачів. Ці профілі встановлюють зв'язок "Багато до одного" між школами та аккаунтами відповідно. Крім того, профіль містить інформацію про роль користувача і може містити іншу специфічну інформацію, притаманну школі. Профілі обов'язково повинні мати прив'язку до школи і аккаунту. Для створення профілю директора необхідно увійти до особистого аккаунту і заповнити відповідну форму (будь-хто може створити школу і аккаунт стане директором школи). Після створення школи, її директор або заступник можуть створити профілі для інших користувачів. Директор, чий профіль створюється автоматично при створенні школи, може створювати заступників, вчителів та студентів. Заступник може створювати лише вчителів та студентів. Вчителі та студенти не мають можливості створювати шкільні профілі. Класи можуть бути створені та заповнені/змінені лише заступником. Групи, з своєї сторони, створюються та заповнюються вчителем. В групу можна одразу додати всіх студентів конкретного класу. Студенти можуть переглядати асоційовані з ними групи (визначені вчителем) та пов'язані з групою навчальні плани. Кожен студент може виконувати завдання та переглядати лекції, доступні відповідно до навчальних планів. Весь прогрес студента зберігається і прив'язується до його профілю. Вчителі також можуть переглядати прогрес кожного студента за навчальним планом у групах, до яких вони мають доступ.

Модуль **Tasks Service** в проекті Mathpar Learning відповідає за управління навчальними завданнями. Завдання в системі можуть бути двох типів: "Потребують відповіді" і "Не потребують відповіді". Перший тип включає будь-які задачі, які мають умову і питання для вирішення. Ці завдання використовуються переважно для практичних вправ та екзаменаційних блоків у шкільних програмах. Завдання, які не потребують відповіді, в основному представляють собою лекційний матеріал, створений для підтримки навчання. Вони все ще включені в категорію завдань, оскільки мають інтерактивну природу. Це означає, що кожна лекція написана з використанням розмітки для хмарних обчислень, що дозволяє студентам змінювати будь-яку частину лекції та перераховувати результат. Таким чином досягається вищий рівень зрозуміння, оскільки лекції перестають бути просто текстом і стають завданнями, які студенти можуть легко відтворити і повторити самостійно. Кожне завдання може бути створене з нуля або на основі іншого (батьківського) завдання. Ієрархія завдань завжди зберігається в базі даних, щоб мати можливість відтворити оригінал. Авторство будь-якого завдання прив'язане до облікового запису користувача, який створив завдання. Оскільки завдання можуть використовуватись не тільки в шкільних системах, прив'язка здійснюється до облікового запису, а не до шкільного профілю.

Також в цьому модулі реалізовано можливість створення навчальних планів. Навчальні плани складаються з окремих завдань і дати, доки дійсне завдання(дедлайн).

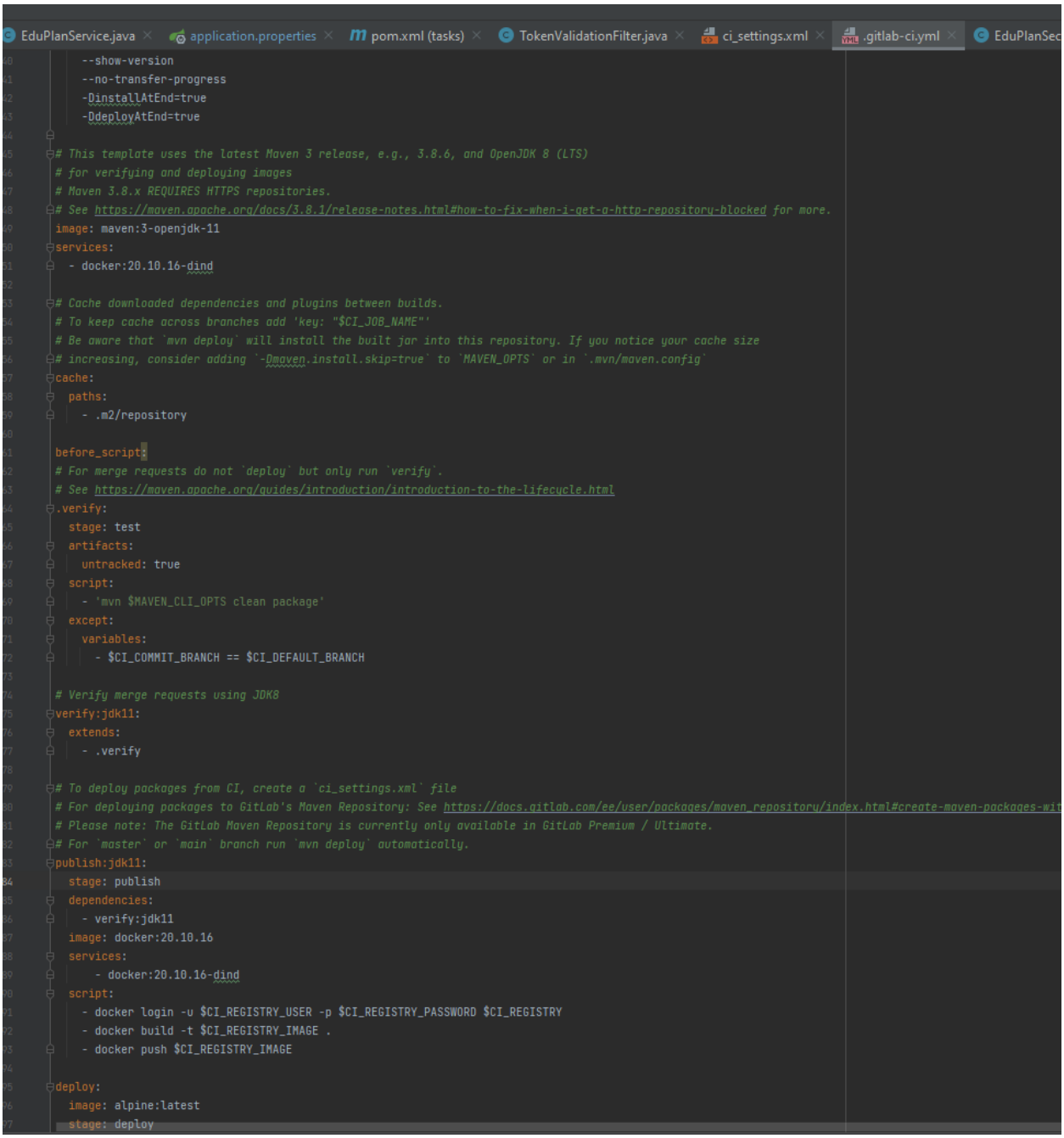
Окремим модулем, який відповідає за збереження та контроль вразливих параметрів бекенд сервісів, є **Secrets Manager**. Цей модуль призначений для безпечного зберігання таких важливих параметрів, як паролі, рядки

підключення до баз даних, доступи до поштових сервісів та інше. Використання Secrets Manager було мотивоване cloud-native підходом, оскільки більшість хмарних провайдерів надають можливість безпечного зберігання вразливих параметрів у хмарних сховищах. Це спрощує інтеграцію з такими сховищами, оскільки кожен бекенд сервіс може отримувати вразливі параметри з зовнішнього джерела, яким є secret manager модуль.

Secrets Manager модуль використовує файлову систему для зберігання параметрів. Ці параметри можуть бути закодовані або зберігатися у чистому вигляді (рекомендується перший варіант). Файли з параметрами можуть бути змінені в реальному часі, і не потребується перезапускати модуль для їх оновлення. Для полегшення управління, введене поняття namespace-ів. Кожен namespace є скупченням параметрів, які об'єднані за певною характеристикою. Наприклад, "account.namespace" представляє скупчення параметрів, що використовуються сервісом "account".

Застосування Spring Cloud Config є альтернативою Secrets Manager модулю. Spring Cloud Config також забезпечує збереження і управління конфігураційними параметрами в хмарному середовищі. Він надає можливість централізованого зберігання і керування конфігураціями, які можуть бути використані різними сервісами. Інтеграція Spring Cloud Config також полегшує заміну або розширення конфігураційного модуля залежно від потреб проекту.

Проект Mathpar Learning також реалізовує CI/CD підходи в своїй розробці. При кожній спробі зливання змін у головну (integration) гілку запускаються автоматичні GitLab CI pipelines. Декларативний опис пайплайну міститься у файлі .gitlab-ci.yml.



Пайплайн складається з 3-х етапів. Побудова пакунку, після цього відбувається збірка докер образу і йог опублікація у відповідний registry, після цього відбувається деплой безпосередньо на сервер, де і розгортається сервіс. GitLab CI дозволяє параметризувати пайплайни під потреби, приховати чутливі дані,

а також надає можливість моніторингу стану системи. Цей інструмент дозволяє уникнути використання Jenkins і спростити процес CI/CD.

Під час розробки системи Mathpar Learning було приділено велику увагу тестуванню. Для цього був розроблений власний тестовий фреймворк з використанням відкритого фреймворку Cucumber-java. Цей фреймворк дозволяє створювати потужну тестову базу, яка дозволяє перевіряти роботу різних частин функціоналу вже інтегрованої і розгорнутої системи на конкретному середовищі.

Залежно від рівня деталізації тестування використовуються різні типи тестів:

1. Поверхневе API тестування: це перевірка працездатності основних функцій системи, таких як створення облікових записів, шкіл, робота з завданнями і планами тощо. Цей тип тестування проводиться швидко, але перевіряє лише базовий набір функцій, щоб переконатися, що системою можна користуватися без помітних помилок. Використовуються відкриті API ендпоінти, які використовує клієнтський застосунок.

2. Детальне API тестування: цей тип тестування перевіряє працездатність всіх функцій системи. Воно включає в себе поверхневе тестування з більшим покриттям. Це дозволяє зробити висновок про те, чи можна використовувати всі функції системи. Використовуються відкриті API ендпоінти, які використовує клієнтський застосунок.

3. Регресійне тестування: цей тип тестування перевіряє не тільки працездатність всіх функцій системи, а й конкретні випадки, що раніше мали помилки або потенційні проблеми. Регресійне тестування дозволяє отримати

чіткий висновок про стан системи. Цей тип тестування має найвище покриття, але він займає більше часу через велику кількість коду, який треба перевірити.

4. UI тестування: цей тип тестування відкриває віртуальний браузер і перевіряє правильність роботи клієнтського застосунку. Це найближчий до клієнта тип тестування, оскільки перевіряє взаємодію з функціоналом, з яким взаємодіє клієнт.

Важливою особливістю є те, що кожен тест у будь-якому наборі тестів є абсолютно незалежним від інших тестів, порядку тестування і середовища. Ресурси, необхідні для проведення тесту, створюються самим тестом, а відповідальність за очищення будь-яких тестових даних, створених під час тестування, лежить на самому тесті. Це дозволяє використовувати будь-який набір тестів для тестування будь-якого середовища в будь-який момент часу. Після завершення тестів не потрібно виконувати жодних ручних дій.

Приклад тестового сценарію виконаного з використанням бібліотеки Cucumber і підходу BDD, широко застосованого у тестуванні веб додатків:

```
  Scenario: Group can be created and deleted successfully
    When I try to create group with name "Generic school group" headed by teacher "Teacher"
    Then Request finished with status 200
    And I save result with key "group1"
    When I try to get group "group1"
    Then Request finished with status 200
    And I verify the group has name "Generic school group" and assign to teacher "Teacher"
    When I try to delete group "group1"
    Then Request finished with status 200

#  TODO uncomment when error handling is implemented
#    When I try to get group "group1"
#    Then Request finished with status 404
```

Глава 2. Використання Mathpartner як обчислювального модуля

2.1. Створення навчального контенту з використанням синтаксису LaTeX

2.1.1. Створення, імпортування підручників на національному рівні.

Однією з найважливіших проблем, пов'язаних з дистанційною освітою та електронним навчанням, є спосіб подання матеріалу та завдань учням. У стандартній схемі освіти матеріали зазвичай надаються у паперовій формі, таких як книги, посібники, збірники тощо. В такому форматі автори можуть передати свої ідеї максимально доступним способом, оскільки надрукований статичний матеріал залишається незмінним і завжди має той вигляд, який надав його автор. Проте в електронних матеріалах відсутній цей фактор. Завдяки постійній зміні технологій, спосіб подання матеріалу також може змінюватись. Крім того, актуальність дистанційної освіти виросла в рази після епідемії Коронавірусу, а зараз і воєнного стану. Тисячі дітей вимушені були змінити своє місце проживання, а відповідно і навчання.

Ми могли б використовувати просто зображення підручників з формулами, але це є неефективно і менш гнучко при потребі внесення змін. Натомість існує спеціалізована «мова» верстки документів призначена саме для технічного і наукового спрямування.

LaTeX (вимовляється "лей-тех" або "ла-тех") - це мова та система верстки документів, яка широко використовується для створення наукових, технічних і математичних документів. Вона була розроблена Леслі Лампортом у 1980-х роках і стала популярною серед дослідників, вчених і академічних письменників.

LaTeX використовується для створення документів зі складною структурою, які містять математичні формули, таблиці, зображення, посилання, бібліографічні посилання тощо. Однією з найбільших переваг LaTeX є його здатність автоматично форматувати документи, що дозволяє авторам зосередитися на змісті, а не на зовнішньому вигляді.

У LaTeX документи створюються у вигляді текстових файлів з розширенням .tex, які потім компілюються у вихідний документ у форматі PDF, DVI або інших. LaTeX використовує систему макророзширень, що дозволяє створювати власні команди та оточення для спрощення форматування та структурування документів.

Багато видавництв, журналів і конференцій активно використовують LaTeX для підготовки своїх публікацій, оскільки він забезпечує професійний вигляд та високу якість верстки. LaTeX є популярним серед академічних спільнот, особливо в галузях, де математика та наука важливі компоненти.

Загалом, LaTeX є потужним інструментом для створення наукових і технічних документів з високою якістю верстки та математичних формул. Однак він може вимагати певного часу та зусиль для оволодіння його синтаксисом та функціональністю.

Саме його і буде використано в проекті Mathpar Learning

Приклад тексту написаного з використанням розмітки LaTeX:

a^n = $(-1)^5$ = (−1) • (−1) • (−1) • (−1) • (−1) = −1;

Даний текст транслюється в наступне візуальне відображення:

a^n=(-1)^5= $(-1) \bullet (-1) \bullet (-1) \bullet (-1) \bullet (-1) = -1;$

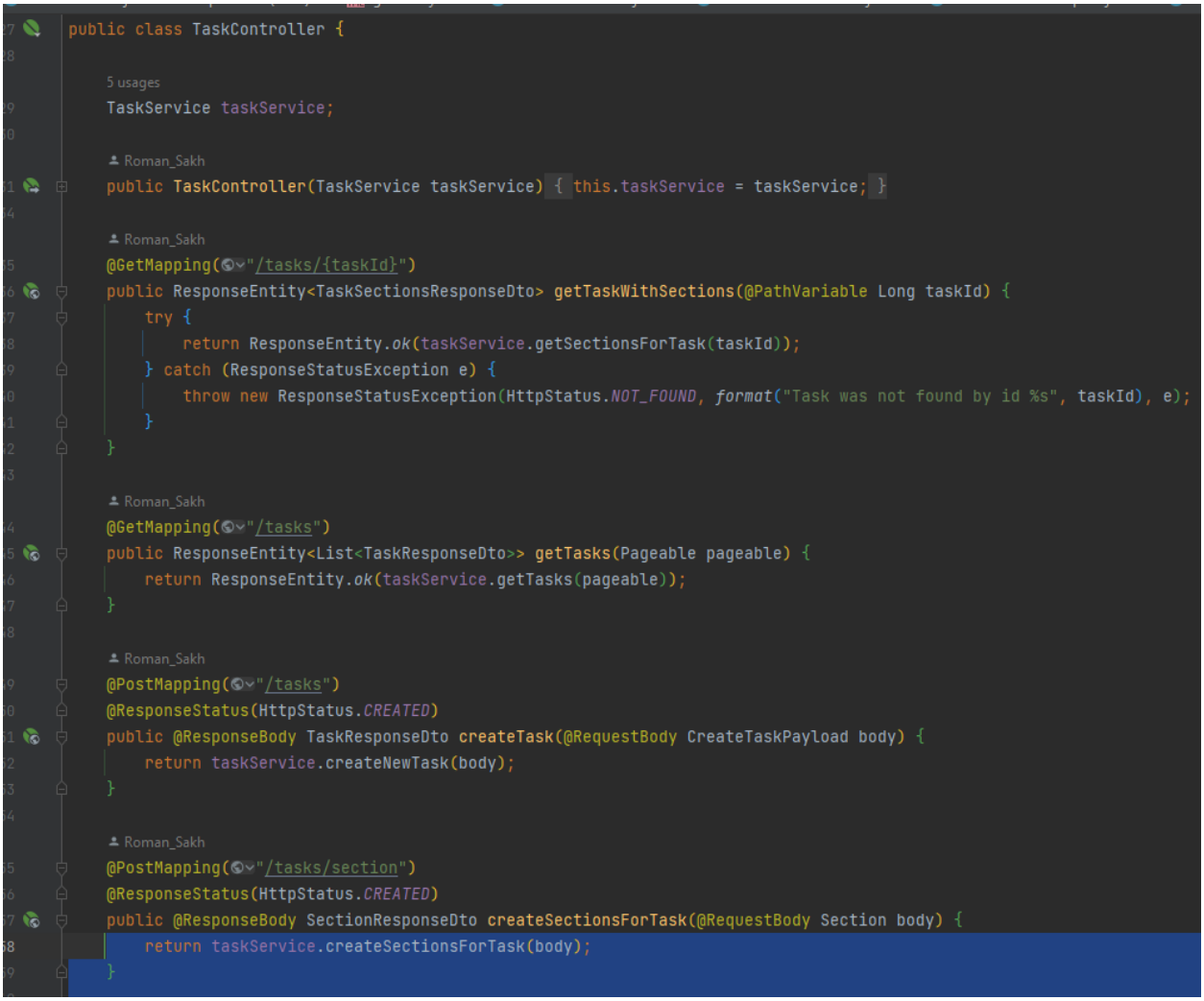
2.1.2 Створення підручника викладачем

Оскільки вчителі та автори підручників є основними джерелами матеріалів для системи після створення початкової бази підручників, нам потрібно створити зручний і простий спосіб для створення цих матеріалів або перенесення їх з паперового формату. Так як електронні матеріали зазвичай зберігаються у форматі LaTeX, ми можемо скористатись існуючими інструментами для створення та редагування таких текстів.

Однак, ми також маємо надати можливість вчителям або авторам писати тексти вбудованому середовищі програми, щоб уникнути залежності від зовнішніх умов. Для цього потрібно дозволити писати текст матеріалів з використанням розмітки, а також здійснювати перевірку та попереднє відображення тексту у відповідному форматі.

Можливість писати текст та попередньо його переглядати можна легко реалізувати в одному клієнтському застосунку, використовуючи наявні бібліотеки та ресурси. Головне завдання для клієнтського застосунку полягає у зручності написання тексту матеріалів з використанням розмітки. Оскільки матеріали часто містять багато формул, необхідно забезпечити швидке перемикання між режимами введення тексту та спеціальних символів, а також додати систему автодоповнення формул, що спростить інтерфейс та заохотить вчителів до створення електронних матеріалів.

API для створення нових підручників реалізований у вигляди класичного CRUD API.



```java
public class TaskController {

    5 usages
    TaskService taskService;

    Roman_Sakh
    public TaskController(TaskService taskService) { this.taskService = taskService; }

    Roman_Sakh
    @GetMapping("/tasks/{taskId}")
    public ResponseEntity<TaskSectionsResponseDto> getTaskWithSections(@PathVariable Long taskId) {
        try {
            return ResponseEntity.ok(taskService.getSectionsForTask(taskId));
        } catch (ResponseStatusException e) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, format("Task was not found by id %s", taskId), e);
        }
    }

    Roman_Sakh
    @GetMapping("/tasks")
    public ResponseEntity<List<TaskResponseDto>> getTasks(Pageable pageable) {
        return ResponseEntity.ok(taskService.getTasks(pageable));
    }

    Roman_Sakh
    @PostMapping("/tasks")
    @ResponseStatus(HttpStatus.CREATED)
    public @ResponseBody TaskResponseDto createTask(@RequestBody CreateTaskPayload body) {
        return taskService.createNewTask(body);
    }

    Roman_Sakh
    @PostMapping("/tasks/section")
    @ResponseStatus(HttpStatus.CREATED)
    public @ResponseBody SectionResponseDto createSectionsForTask(@RequestBody Section body) {
        return taskService.createSectionsForTask(body);
    }
```

Одним з етапів написання матеріалів є виконання обчислень. Система Mathpartner не лише конвертує LaTeX-розмітку в графічний вигляд, але також обчислює значення змінних та формул, що містяться у тексті. Це створює інтерактивність в лекційному матеріалі, про яку вже згадувалося. Більш того, система дозволяє створювати графіки за допомогою спеціальної розмітки. Щоб отримати такі результати, текст матеріалу з розміткою повинен бути відправлений на сервер для обробки. Оскільки цей процес вимагає багато ресурсів, використовувати його під час кожної зміни тексту є неефективним. Тому пропонується можливість переглянути кінцевий вигляд матеріалу (з рендерингом на серверній стороні) при натисканні на спеціальну кнопку.

2.2. Розробка самостійних і контрольних робіт, із прикладами вирішення та підказками.

Крім лекційного матеріалу, система також має надавати можливість створювати завдання для вирішення. Ці завдання можуть бути як тестовими, так і завданнями з відкритою відповіддю. Реалізація тестових завдань вже має бути включена в багатьох відкритих проектах, тому ми не будемо розглядати її реалізацію в межах проекту Mathpar Learning.

Давайте розглянемо створення завдань з відкритою відповіддю. Ці завдання складаються з двох компонентів: умови та остаточної відповіді у формульному або цифровому вигляді. Важливо зрозуміти, що завдання з відкритою відповіддю не обов'язково мають цифрову відповідь, оскільки деякі з них можуть бути параметризованими, що вимагатиме дослідження можливих значень параметрів або інших параметризованих відповідей. Стандартні математичні операції недостатньо для перевірки відповіді.

Ми можемо розглянути можливість використання сервісу Mathpar Calculator для вирішення цієї проблеми. Оскільки Mathpar Calculator дозволяє виконувати обчислення з формульними виразами, ми можемо визначити відповідь на завдання з відкритою відповіддю як формульний вираз, записаний у мові розмітки LaTeX. Таким чином, будь-який вираз може бути прийнятий як відповідь. Крім того, це дозволить відображати відповіді у зручному для перегляду форматі без потреби додаткової взаємодії з умовами та лекційними матеріалами. Під час створення завдання, нам достатньо буде приймати вираз, написаний мовою LaTeX, як відповідь, і зберігати його у базі даних для подальшого порівняння з відповідями студентів. Крім того, відповідь може бути представлена не тільки як формульний рядок, але і як повний крок-за-кроком розв'язок. Таким чином, за допомогою сервісу Mathpar Calculator,

покроковий розв'язок може бути конвертований у кінцевий формульний вираз для порівняння з відповідями студентів.

Умови завдань фактично є аналогічними до прикладів завдань у лекційних матеріалах. Оскільки приклади завдань у матеріалах зберігаються як частина лекційного тексту у вигляді LaTeX-розмітки, умови завдань будуть зберігатися так само. Для відображення умови завдання буде використовуватись така ж розмітка.

2.3. Використання можливостей Mathpartner, для перевірки правильності відповіді учня або демонстрації правильного рішення.

Сервіс Mathpar Learning стикається з важливою задачею - перевіркою коректності відповідей на практичні завдання з відкритою відповіддю. У цих завданнях немає єдиного правильного варіанту, який можна просто порівняти. Однак, Mathpar Learning вирішує цю проблему, використовуючи функціонал, наданий сервісом Mathpar Calculator.

Mathpar Calculator дозволяє виконувати операції, записані в LaTeX форматі, а також порівнювати два списки операцій на рівність. Ця друга функція є необхідною для перевірки коректності відповідей студентів. Завдяки цій функції можна порівняти відповідь, надану студентом у вигляді LaTeX-розміченого тексту, зі збереженим в базі даних текстом правильної відповіді. Якщо результат порівняння дає порожнє значення, це означає, що відповіді еквівалентні, тобто студент надав коректну відповідь.

Цей підхід також дозволяє зберігати історію наданих відповідей студентів у базі даних, що записані у форматі LaTeX. Таким чином, студент може подавати кілька спроб розв'язання одного завдання, а система зможе

перевірити їх на коректність. Це особливо корисно для практичних завдань, створених для навчання, де процес навчання є важливішим за оцінку знань.

Також, завдяки тому, що правильні відповіді зберігаються у базі даних у форматі LaTeX, студент може отримати правильну відповідь у візуальному вигляді, якщо це потрібно. Наприклад, якщо студент не знає, як розв'язати завдання, він може скористатися можливістю "здатися". При цьому студент не отримує балів за завдання, але може побачити коректну відповідь для подальшого вивчення. Ця можливість доступна лише для обмеженого кола завдань і не поширюється на екзаменаційні ситуації.

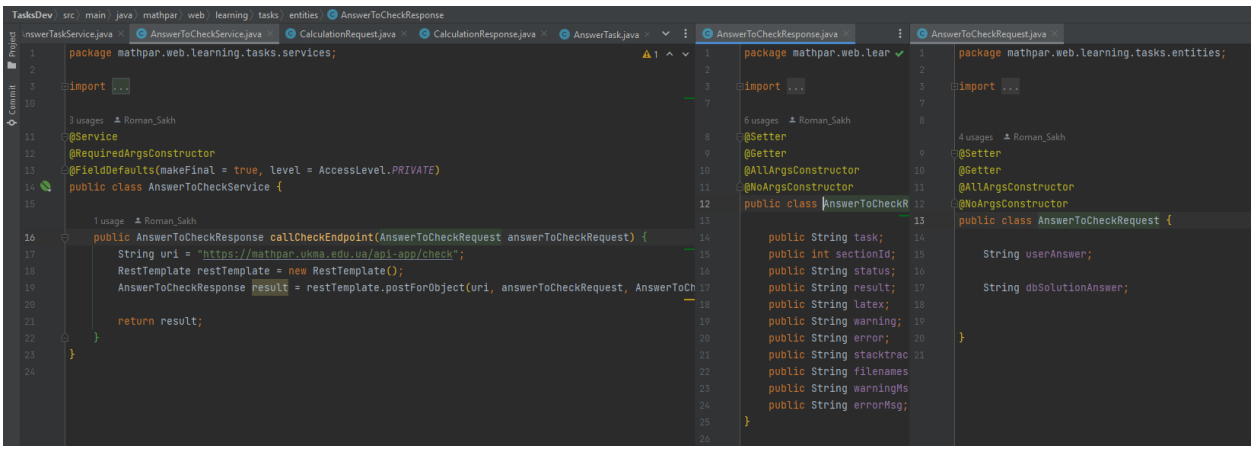


Рис 2.3.1. Сутність для зберігання запиту і відповіді від компоненту перевірки правильності відповіді

2.4. Відслідковування успішності і результатів навчання учнів. Доступ до інформації про успішність

Система Mathpar Learning повинна надавати вчителям широкий спектр функціональності для оцінювання та аналізу можливостей студентів. Ось деякі можливості системи, які допомагають учителям в оцінюванні та виставленні оцінок:

1. Оцінювання завдань: Mathpar Learning може містити функціонал для створення та оцінювання завдань, тести, та інших форматів контролю знань. Вчителі можуть створювати завдання, встановлювати критерії оцінювання, та оцінювати відповіді студентів в системі.

2. Аналіз успішності: Вчителі можуть проглядати успішність студентів на основі їх відповідей та результатів оцінювання. Система може надавати зведену інформацію про відповіді студентів, оцінки, а також статистику про загальну успішність групи або окремих учнів.

3. Відстеження зусиль та часу: Mathpar Learning може записувати і відстежувати зусилля та час, витрачені студентами на розв'язання завдань та виконання вправ. Вчителі можуть аналізувати цю інформацію для оцінки ступеня залученості студентів, їхнього прогресу та виявлення можливих труднощів.

4. Виставлення оцінок: Система може надавати функціонал для виставлення оцінок студентам на основі їхніх відповідей та результатів оцінювання. Вчителі можуть встановлювати вагу різних завдань, визначати критерії оцінювання та виставляти оцінки в системі.

5. Персоналізоване навчання: Mathpar Learning може аналізувати результати студентів та рекомендувати індивідуальні матеріали та завдання на основі їхніх потреб та рівня знань. Це допомагає вчителям підлаштовувати навчальний процес до кожного учня окремо та покращувати їхні результати.

Оскільки використання часових міток може створювати моральний тиск на студентів і негативно впливати на їхні зусилля, потрібно знайти інший спосіб визначення кількості часу, витраченого на завдання. Крім того, враховуючи, що вчителі часто працюють з великою кількістю студентів

одночасно, необхідно мати можливість швидко та компактно оцінити успішність будь-якого студента без детального аналізу кожного завдання.

Для вирішення цих проблем була розроблена система рядкового подання успішності. Її суть полягає в кодуванні прогресу конкретного студента в конкретному наборі завдань у вигляді єдиного рядка символів, використовуючи певні правила. Основні правила цієї системи такі:

- Якщо студент вирішив конкретне завдання з N спроби, то задача кодується як цифра від 1 до 9, в залежності від N. Якщо N більше 10, використовується цифра 9 як максимально можлива кількість спроб для статистики.

- Якщо студент зробив N спроб і не зміг правильно вирішити завдання, то задача кодується літерою від A до I. При N більше 10 використовується літера "I" як максимально можлива кількість спроб для статистики.

- Якщо студент зробив N спроб і здався (переглянув правильну відповідь), то задача кодується літерою від a до i. При N більше 10 використовується літера "i" як максимально можлива кількість спроб для статистики.

Цим способом будь-яке завдання може бути представлене лише одним символом, а ми можемо швидко отримати статистику по великій кількості завдань у вигляді одного рядка.

Це дозволяє в подальшому отримувати статистичні дані успішності на рівні школи, відділу освіту.

Глава 3.Створення бази задач та завдань для використання в освітньому процесі

3.1 Навчальні матеріали

Система Mathpar Learning дійсно має значну корисність, яка проявляється не тільки у функціях для дистанційного та додаткового навчання, але і в її базі попередньо набраних навчальних матеріалів. Ця база дозволяє вчителям швидко створювати власні навчальні плани, не витрачаючи багато часу на створення матеріалів з нуля або перетворення існуючих.

Наявність попередньо набраних навчальних матеріалів в системі Mathpar Learning дає вчителям можливість скористатися готовими ресурсами для підготовки уроків та матеріалів для учнів. Це значно спрощує їх роботу, оскільки вони можуть знайти в системі відповідні матеріали за потрібною темою і використати їх безпосередньо або внести необхідні зміни.

Крім того, наявність бази попередньо набраних навчальних матеріалів дозволяє школам легше впроваджувати систему Mathpar Learning у навчальний процес. Замість того, щоб починати зі створення всіх матеріалів з нуля, вони можуть скористатися наявними ресурсами і адаптувати їх до своїх потреб.

Таким чином, система Mathpar Learning полегшує роботу вчителям, дозволяючи їм швидко створювати навчальні плани та матеріали і впроваджувати цю систему у навчальний процес шкіл.

Крім того в подальшій перспективі така база підручників може бути уніфікованою на рівні держави. Так, створення бази підручників на рівні держави та її впровадження скрізь може бути вельми корисним кроком у

поліпшенні навчального процесу. Ось кілька переваг, які можуть виникнути в результаті такої ініціативи:

1. Стандартизація навчальних матеріалів: Загальнодержавна база підручників дозволить стандартизувати навчальні матеріали і забезпечити їх відповідність державним навчальним програмам. Це сприятиме однаковості та якості освіти у всіх регіонах.

2. Економія часу та ресурсів вчителів: Завдяки наявності готових матеріалів в базі, вчителям не потрібно буде витрачати багато часу на пошук та створення власних навчальних матеріалів. Вони зможуть скористатися наявними ресурсами, або вносити необхідні зміни в існуючі, що спростить їх роботу.

3. Покращення доступності освіти: Загальнодержавна база підручників може забезпечити рівну можливість отримання якісної освіти для учнів з різних регіонів та шкіл. Всі школи матимуть доступ до однакових навчальних матеріалів, що допоможе зменшити нерівності в освітньому процесі.

4. Збереження та поширення знань: База підручників може стати джерелом знань та навчальних матеріалів для всіх зацікавлених сторін, включаючи студентів, батьків та інших навчальних працівників. Це сприятиме поширенню і обміну знаннями в широкому контексті.

Звичайно, створення та впровадження загальнодержавної бази підручників вимагає значних зусиль, координації та фінансування. Але в перспективі це може мати значний позитивний вплив на освіту, сприяючи підвищенню якості навчання та рівності можливостей для всіх учнів.

3.2. Створення бази навчальних планів дисциплін для різних рівнів складності.

3.2.1 Національні навчальні плани

Введення поняття навчального плану дійсно допомагає систематизувати та контролювати навчальний процес в системі Mathpar Learning. Навчальний план визначає структуру та наповнення навчання, включаючи розподіл тем, лекційних матеріалів, практичних завдань та інших ресурсів на певний період часу.

Навчальний план в системі Mathpar Learning може включати наступні елементи:

1. Послідовність тем: Навчальний план визначає послідовність тем, які мають бути вивчені учнями протягом навчального року або семестру. Це допомагає вчителям організувати навчальний процес і забезпечити прогресивне навчання.

2. Лекційні матеріали: Навчальний план може включати посилання на конкретні лекційні матеріали, які відповідають кожній темі. Це дозволяє вчителям легко знайти та надати доступ до відповідних матеріалів для учнів.

3. Практичні завдання: Навчальний план також може містити завдання, які пов'язані з кожною темою. Ці завдання можуть бути використані для практичного застосування знань та вмінь, отриманих учнями під час навчання.

4. Контрольні завдання та оцінювання: Навчальний план може включати також контрольні завдання та оцінювання, які дозволяють вчителям виміряти прогрес та розуміння учнів. Це може включати тестування, домашні завдання, проекти тощо.

Структура та наповнення навчального плану визначаються вчителями або освітніми закладами, з урахуванням вимог державних стандартів та програм. Система Mathpar Learning надає зручні інструменти для створення та управління навчальним планом, що дозволяє вчителям організувати та контролювати освітній процес ефективніше.

Так, в системі Mathpar Learning кожен навчальний план може складатись з блоків, які включають матеріали, практичні завдання, екзаменаційні завдання та інші компоненти. Блоки є універсальною структурною одиницею, яка дозволяє гнучко організовувати та структурувати навчальний процес.

Кожен блок може містити різні типи матеріалів, наприклад:

1. Лекційні матеріали: Це можуть бути текстові матеріали, відео-лекції, діаграми, графіки, таблиці тощо, які пояснюють певну тему або концепцію.

2. Практичні завдання: Блок може містити практичні завдання, що допомагають учням застосувати отримані знання та вміння у практичних ситуаціях. Це можуть бути задачі, лабораторні роботи, проекти тощо.

3. Екзаменаційні завдання: Блок може включати завдання, призначені для перевірки розуміння та оцінювання успішності учнів. Це можуть бути тести, контрольні роботи, заліки тощо.

Крім того, в системі Mathpar Learning також можливо налаштувати систему контролю та оцінювання, яка дозволяє вчителям встановлювати параметри оцінювання, стежити за прогресом учнів та надавати зворотний зв'язок.

За допомогою блоків, учителі можуть гнучко структурувати навчальний процес, додавати та змінювати матеріали, завдання та оцінки відповідно до потреб учнів та цілей навчання.

3.2.2. Групові навчальні плани

Так, система Mathpar Learning надає можливість гнучкої модифікації та створення власних навчальних планів вчителями загальноосвітніх закладів. Хоча система має попередній набір стандартизованих навчальних планів, рекомендованих освітньою адміністрацією, вчителі можуть вносити корективи до цих планів або створювати власні плани з нуля, враховуючи свої особисті потреби та підходи до викладання.

Вчителям надаються інструменти для гнучкої модифікації навчальних планів, включаючи можливість:

1. Додавати, видаляти або змінювати блоки: Вчителі можуть редагувати навчальні плани шляхом додавання, видалення або зміни блоків, що включають матеріали, завдання та оцінки.

2. Переставляти та змінювати порядок блоків: Вчителі можуть змінювати порядок блоків у навчальному плані, щоб адаптувати його до власних потреб та вподобань.

3. Вибирати матеріали: Вчителі можуть вибирати матеріали з попередньо набраної бази лекційних матеріалів та практичних завдань, або вносити власні матеріали, щоб персоналізувати навчання.

4. Встановлювати параметри оцінювання: Вчителі можуть налаштовувати систему оцінювання, встановлювати критерії оцінювання та вагу різних завдань в навчальному плані.

Таким чином, система Mathpar Learning дозволяє вчителям гнучко модифікувати та створювати власні навчальні плани, відповідно до їхніх потреб та вподобань, що дозволяє забезпечити більш індивідуалізований та ефективний навчальний процес. Таким чином будь-який викладач має можливість створювати власну базу навчальних планів залежно від власних потреб.

Приклад сутності навчальних планів:

```java
@Entity
@Table(name = "edu_plan")
public class EduPlan {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    private String eduPlanTitle;

    @OneToMany(mappedBy = "eduPlanId", fetch = FetchType.LAZY)
    private Set<EduPlanTask> taskList;

}
```

```java
@Entity
@Table(name = "edu_plan_task")
public class EduPlanTask {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @Column(name = "section_id")
    private Long sectionId;

    @ManyToOne
    @JoinColumn(name = "edu_plan_id", nullable = false)
    private EduPlan eduPlanId;

    @Column(name = "TASK", nullable = false, columnDefinition = "TEXT")
    String task;

    @Column(name = "ANSWER", columnDefinition = "TEXT")
    String answer;

    @Column(name = "LATEX", columnDefinition = "TEXT")
    String latex;

    @Column(name = "LEGACY_SECTION_ID")
    Integer legacySectionId;


    @Column(name = "TASK_ID", nullable = false)
    Long parentTask;

    @Column(name = "HAS_ANSWER", columnDefinition = "bit default true")
    Boolean hasAnswer;

    @Column(name = "TIP", columnDefinition = "TEXT")
    String tip;
```

3.3. Моніторинг успішності навчання на національному рівні

Завдяки доступній базі щоденників можна організувати детальний моніторинг навчального процесу будь-якого навчального закладу або навіть вибірки навчальних установ. Збираючи дані про прогрес учнів (наприклад, практичні завдання) у вигляді рядків, можна створити великий рядок

успішності для будь-якої групи студентів і аналізувати його за допомогою електронних інструментів.

Наприклад, обласна освітня адміністрація може взяти вибірку учнів 11-х класів певної області, отримати їх щоденники та підрахувати загальну кількість символів у скомпонованому рядку. Це дозволить отримати уявлення про складність матеріалу, рівень старанності учнів та навантаження навчального плану шляхом аналізу кількості символів на студента. Крім аналізу практичних завдань, також можна легко отримати вибірку фінальних оцінок студентів за певний блок завдань, використовуючи записи про результати їх екзаменаційних робіт.

Такий аналіз може проводитись не тільки обласними адміністраціями, але й різними іншими вибірками. Адміністрація навчального закладу може вести моніторинг успішності учнів, міська адміністрація - аналізувати успішність в різних школах міста тощо. Використання системи електронних щоденників відкриває нові можливості для аналізу навчального процесу в реальному часі. Завдяки хмарній природі сервісу Mathpar Learning, такі вибірки та аналізи можна проводити в будь-який момент навчального процесу для швидкого реагування, коригування та покращення якості навчання.

Глава 4. Інтеграція хмарних обрахунків в освітній процес математики, фізики, хімії

4.1 Використання хмарних обчислень в процесі викладання математики

Mathpar Learning пропонує інтегрувати хмарні обчислення у освітній процес математики, зокрема за допомогою сервісу Mathpar Calculator. Цей сервіс дозволяє обчислювати значення математичних виразів, які введені у форматі LaTeX. Вирази можуть містити різні математичні операції, від базових арифметичних до складніших, таких як логарифми, тригонометрія, рівняння, інтеграли та похідні.

Проте, під час навчання математики важливо не лише отримувати відповіді, а й навчитися розв'язувати задачі самостійно. Mathpar Learning розуміє цю потребу і прагне розвивати логічне мислення студентів, їхню здатність до аналізу та розв'язання задач.

Ідея полягає в тому, щоб обмежити функціонал хмарних обчислень таким чином, щоб студентам було доступно автоматичне розв'язування завдань, які вони вже вивчили. При цьому обчислювальний сервіс Mathpar Calculator надає широкий набір функцій для автоматичного розв'язування практично будь-яких задач.

Такий підхід дозволяє спростити рутинну обчислювальну роботу, зменшити навантаження на студентів і збільшити їхній інтерес до розв'язання складних задач. Водночас, зберігаються можливості вивчати та практикувати розв'язання різних функцій шляхом штучного обмеження функціоналу хмарних обчислень.

Такий підхід до інтеграції хмарних обчислень у навчальний процес математики сприяє поліпшенню ефективності навчання, розвитку аналітичного мислення студентів і збереженню їхньо

4.2. Використання хмарних обчислень під час вивчення фізики

Учень, який вивчає фізику, використовує математичні методи для аналізу практичних задач. Однак, зрозуміння фізичних явищ і процесів, а також здатність інтерпретувати практичні задачі в контексті фізики, стають все більш важливими, оскільки акцент з переміщується від вміння розв'язувати математичні функції до знання фізичних формул.

Хмарні обрахунки можуть відігравати важливу та корисну роль у поліпшенні освітнього процесу фізики. Фізичні задачі можуть включати складні математичні функції, такі як інтегрування та диференціювання, а також графічні аспекти математики, наприклад, декартові площини.

Хмарні обчислення дозволяють нам обчислювати складні математичні функції з високою точністю та зосереджуватися на вирішенні фізичних задач. У відміну від обмежень, які можуть існувати щодо використання хмарних обчислень у математиці, їх використання в фізиці не потребує обмежень. Важливо мати повний функціонал для всіх можливих задач.

Також слід відзначити, що сервіс MathparCalculator надає можливість працювати з 2D графікою для аналізу функцій. Використання цього функціоналу полегшує і поліпшує процес вивчення фізичних явищ.

4.3. Використання хмарних обчислень під час вивчення хімії

Інтеграція хмарних обчислень у процес вивчення хімії дійсно може мати багато переваг. Вона дозволяє студентам проводити точні обрахунки з використанням потужних обчислювальних ресурсів, що прискорює процес

отримання відповідей на задачі. Окрім того, використання хмарних обчислень забезпечує високу точність результатів, зменшуючи ймовірність механічних помилок.

Однією з переваг хмарних обчислень є можливість виконання складних математичних операцій та обрахунків, які можуть бути важкими або часомісткими для виконання вручну. Сервіси, як Mathpar Calculator, дозволяють виконувати розрахунки за складними формулами, спрощувати вирази та визначати значення змінних в оброблюваному тексті. Це дозволяє студентам зосередитися на розумінні хімічних концепцій та процесів, замість того, щоб витрачати багато часу на механічні обчислення.

Застосування хмарних обчислень також сприяє зручності та доступності. Вони можуть бути доступні з будь-якого пристрою з підключенням до Інтернету, що дозволяє студентам використовувати їх навіть поза класною кімнатою. Вчителі також можуть використовувати ці сервіси для створення інтерактивних завдань та вправ, що сприяють активному навчанню та залученню студентів.

Узагальнюючи, інтеграція хмарних обчислень в процес вивчення хімії допомагає підвищити ефективність студентів, знижує ймовірність помилок та забезпечує високу точність обчислень. Це дозволяє студентам краще розуміти хімічні концепції та процеси, не витрачаючи зайвий час на обчислення.

Висновки

У результаті виконання дипломної роботи був проведений аналіз, де розглядалися можливості інтеграції хмарних технологій та хмарних обчислень в сучасну STEM освіту з метою поліпшення її якості та ефективності із врахуванням сучасних обставин карантину і війни. Крім того, був розроблений прототип продукту, що спрощує інтеграцію згаданих технологій та підходів у будь-який навчальний проект. Розглянемо отримані результати:

- Розглянута можливість впровадження підходів B/G розгортання та представлено приклади реалізації цих підходів.
- Розроблені та впроваджені процедури для CI/CD процесів.
- Сформульовані рекомендації щодо впровадження мікросервісної архітектури в навчальні проекти, а також наведено приклад її реалізації.
- Створено проект з відкритим вихідним кодом, який реалізує всі вищезазначені підходи.
- Розглянута можливість інтеграції хмарних обчислень в освітні процеси математики, фізики та хімії.
- Розроблені та описані способи створення навчальних матеріалів та навчальних планів.
- Створено власні засоби автоматизованого тестування функціоналу, юніт-тести та інтегровані тести.

Наявний результат з невеликим доопрацюванням можна використовувати як демонстраційний продукт для залучення IT-компанії до розвитку продукту. Великі компанії мають окремі підрозділи, які займаються соціально корисними проектами. Покращення освіти загалом, а тим більше STEM, може бути прямою зоною інтересів таких компаній.

Література

1. Carnell J. Spring Microservices in Action / John Carnell.. – 384 с.
2. Microservice Architecture: Aligning Principles, Practices, and Culture / N.Nadareishvili, R. Nadareishvili, M. Matt, A. Mike.. – 146 с.
3. Microservices architecture style [Електронний ресурс]. – 2019. – Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices
4. Микросервисы (Microservices) [Електронний ресурс]. – 2019. – Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/post/249183/
5. How To Set Up a Continuous Deployment Pipeline with GitLab CI/CD on Ubuntu 18.04 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/how-to-set-up-a-continuous-deployment-pipeline-with-gitlab-ci-cd-on-ubuntu-18-04
6. Nedelcu C. Nginx HTTP Server: Adopt Nginx for Your Web Applications to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever / Clement Nedelcu., 2010. – 327 с.
7. Cucumber Guides [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://cucumber.io/docs/guides/
8. Git.Lab CI/CD [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://docs.gitlab.com/ee/ci/
9. Building REST services with Spring [Електронний ресурс]. – 2019. – Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/guides/tutorials/rest/.