Міністерство освіти і науки України

Національний університет «Києво-Могилянська Академія»

Кафедра інформатики факультету інформатики

Хмарна система MathLearning: бекенд

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник курсової роботи

Малашонок Г. І.

_____

(підпис)

“___” _____ 2024 р.

Виконав студент ІПЗ-3

Поліщук Вадим Дмитрович

“___” _____ 2024 р.

Київ – 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

С. С. Гороховський

_____

(підпис)

“___” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Поліщуку Вадиму Дмитровичу факультету інформатики 3 курсу

ТЕМА Хмарна система MathLearning: бекенд

Індивідуальне завдання

Вступ

1 Огляд поточної системи MathLearning

2 Інструменти та технології розробки

3 Розробка системи MathLearning

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі “___” _____ 2024 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Хмарна система MathLearning: бекенд

№ п/п	Назва етапу дипломного проєкту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	жовтень 2023 р.	
2.	Огляд технічної літератури за темою роботи.	листопад-грудень 2023 р.	
3.	Аналіз архітектури програм, розроблених на основі Spring Framework.	грудень-січень 2023 р.	
4.	Проектування архітектури застосунку.	січень 2024 р.	
5.	Розробка застосунку.	лютий-березень 2024 р.	
6.	Написання текстової частини роботи.	квітень 2024 р.	
7.	Створення доповіді.	квітень 2024 р.	
8.	Захист курсової роботи	травень 2024 р.	

Студент Поліщук Вадим Дмитрович

Керівник Малашонок Г. І.

ЗМІСТ

АНОТАЦІЯ 5
ВСТУП 6
1 ОГЛЯД ПОТОЧНОЇ СИСТЕМИ MATHLEARNING 7
1.1 Аналіз поточного стану бекенд частини проєкту MathLearning 7
1.2 Аналіз продуктових вимог до проєкту MathLearning 9
1.3 Створення плану подальшої розробки 12
2 ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ 14
2.1 Використання Spring Cloud Configuration 14
2.2 Схеми міграцій з використанням Flyway 16
2.3 Обробка винятків в Spring Boot 17
2.4 Налаштування безпеки в Spring Boot 19
2.5 Проектування доменної області 20
3 РОЗРОБКА СИСТЕМИ MATHLEARNING 23
3.1 Впровадження покращень до поточної системи 23
3.2 Дизайн домену системи 26
3.3 Написання коду додатку MathLearning 28
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 37
ВИСНОВКИ 38
ДОДАТОК А 39
ДОДАТОК Б 40
ДОДАТОК В 43

АНОТАЦІЯ

Ця курсова робота присвячена вдосконаленню системи MathLearning. Робота включає аналіз поточного стану проєкту та визначення продуктових вимог, на основі яких розроблено план подальшої роботи. Використовуючи сучасні інструменти та технології розробки RESTful застосунків було впроваджено покращення в існуючу систему. Окремо розглянуто питання проектування та оптимізації доменної області. Завдяки цьому дослідженню, система MathLearning отримала значні вдосконалення, що підвищили її ефективність та наблизили до першої користувацької версії.

ВСТУП

Сучасний розвиток інформаційних технологій вимагає постійного вдосконалення існуючих систем та впровадження нових підходів до їх розробки і підтримки. Однією з таких систем є MathLearning — платформа, призначена для інтерактивного навчання математики з потужною математичною базою обчислень.

Метою цієї курсової роботи є вдосконалення системи MathLearning через впровадження сучасних технологій та інструментів. У процесі роботи було проведено аналіз поточного стану бекенд частини проєкту та продуктових вимог, що дозволило визначити основні напрямки подальшої розробки.

Завдяки використанню таких технологій, як Spring Cloud Configuration, Flyway, та різних аспектів налаштування Spring Boot, вдалося не лише підвищити ефективність роботи системи, але й забезпечити її стабільність та захищеність. Особлива увага приділена проектуванню доменної області, що є критично важливим для оптимального функціонування будь-якої програмної системи.

Ця робота спрямована на підвищення якості та функціональних можливостей MathLearning, забезпечуючи тим самим її відповідність сучасним вимогам та стандартам у галузі розробки програмного забезпечення.

1 ОГЛЯД ПОТОЧНОЇ СИСТЕМИ MATHLEARNING

Поточна система проєкту MathLearning складається з бекенд та фронтенд частин. Обидві частини знаходяться в незавершеній стадії та потребують багато роботи для доведення до першої користувацької версії. В цьому розділі проаналізовано поточний стан і функціонал поточної бекенд частини та основні продуктові вимоги на основі магістерської роботи Скуратівського М. Ю. на тему «Хмарна система MathLearning: імплементація продуктових вимог та документації» [1], а також створено план подальшої розробки.

1.1 Аналіз поточного стану бекенд частини проєкту MathLearning

Поточна бекенд частина складається з наступних мікросервісів:

- сервіс акаунтів користувачів.
- сервіс керування школою.
- сервіс навчальних планів.
- конфігураційний сервіс.
- шлюз (gateway).

Сервіс акаунтів слугує для створення та керування акаунтами користувачів, управління їхніми сесіями, а також слугує засобом автентифікації користувачів для інших мікросервісів. Управління сесіями відбувається за рахунок створення власного токену AuthenticationToken, який зберігається в базі даних та містить в собі інформацію про ідентифікатор користувача. Також сервіс містить відкритий ендпойнт для перевірки цього токену, яким користуються інші сервіси для автентифікації користувачів. Крім того, сервіс використовує інструменти для роботи з електронною поштою для надсилання електронних листів користувачам з метою підтвердження профілю, відновлення паролю, тощо. Цей функціонал працює лише в

production середовищі. Відділення логіки керування акаунтами до окремого мікросервісу є поширеною практикою в розробці REST додатків. Це сприяє масштабованості та дозволяє фокусуватися на розробці окремих частин програми незалежно та паралельно.

Сервіс керування школою був створений для керування школами, а саме:

- керування персоналом та студентами. Керування відбувається за допомогою сутності UserProfile та правил, які визначають можливості кожного окремого типу профілю та їх взаємодію.
- керування групами та класами школи. Керування відбувається за допомогою сутностей SchoolGroup та SchoolClass. Обидві сутності слугують для об’єднання списку студентів у логічні групи.
- керування дисциплінами. Дисципліни – основні одиниці навчальних матеріалів, на основі яких відбувається навчання студентів. Цей функціонал не є розробленим в поточній версії додатку.

Також важливою частиною сервісу керування школою є автентифікація користувачів за профілем. Для авторизації використовується сервіс для керування акаунтами, проте також присутня додаткова логіка авторизації, суть якої полягає у пошуку профіля поточного користувача, визначення його ролі в системі школи та правил доступу.

Сервіс для керування навчальними планами відповідає за створення окремих навчальних матеріалів – лекцій, практичних занять, іспитів – та їх об’єднання в навчальні плани, а також збереження та відслідковування прогресу студентів по окремим навчальним матеріалам за допомогою електронного щоденника. В поточній версії додатку реалізовано лише створення окремих завдань для практичних занять та їх об’єднання в навчальні плани. Також сервіс має передбачати належну безпеку, автентифікуючи профілі користувачів та проводячи їх авторизацію для окремих складових частин REST API. На даний момент реалізована лише автентифікація користувачів без належної авторизації.

Конфігураційний сервіс в даній системі MathLearning виконаний у вигляді окремого сервісу, що надає конфігурації для усіх інших частин системи. Такий підхід є дуже поширеним у розробці REST додатків та надає численні переваги. Основна перевага такого сервісу полягає в централізації конфігураційних файлів, що забезпечує відповідність та стабільність різних конфігурацій, а також слугує єдиним джерелом правди для всієї системи. Поточна реалізація конфігураційного сервісу є ручною, тобто отримання та зберігання конфігурацій відбувається через власноруч прописані ендпойнти. Це створює певні складнощі, особливо в клієнтській частині, тому що обробка власної системи конфігурацій мусить бути визначена самостійно та прописана в коді. Основний недолік полягає в тому, що така система є негнучкою, адже потребує змін у кожній частині системи при найменшій зміні в центральному конфігураційному сервісі.

Сервіс шлюз відповідає за маршрутизацію запитів між мікросервісами та є повністю реалізованим в поточній системі MathLearning.

1.2 Аналіз продуктових вимог до проєкту MathLearning

Список продуктових вимог до системи MathLearning був складений та представлений у магістерській роботій Скуратівського М. Ю. на тему «Хмарна система MathLearning: імплементація продуктових вимог та документації» [1]. Вимоги діляться за модулями та епіками, кожен епік має перелік функціоналу, який має бути виконаний. Список розглядає загальні вимоги до продукту, розглянемо детально кожну його частину лише в контексті бекенд частини додатку:

- авторизація та автентифікація – до цього пункту входить реєстрація, вхід та визначення ролей користувачів. За цей функціонал відповідає окремий мікросервіс для управління акаунтами, а також частково

мікросервіс для управління школою, який визначає ролі користувачів;

- створення та редагування школи – сюди належить функціонал по створенню школи, який може виконати лише директор, профіль якого має бути створеним власноруч адміністратором системи. Створення кожної школи має адмініструватися на початку роботи продукту. В майбутньому цей процес можна буде автоматизувати;
- створення та редагування особистих профілів користувачами – за цей функціонал відповідає сервіс школи, який дозволяє менеджменту школи (директору, вчителям, старшим вчителям) створювати профілі інших користувачів, а також надає можливість перегляду власного профілю або профілю інших користувачів. Важливим елементом цього функціоналу є правильне налаштування авторизації, яке має забезпечити безпеку та унеможливити неочікувані та небажані випадки, що теоретично можуть статися в системі. Наприклад, студент не може створювати профілі інших студентів, вчитель не може змінити свою роль на роль директора, тощо;
- створення та керування дисциплінами – менеджмент школи має мати можливість створювати дисципліни, наповнювати їх навчальними матеріалами та ділитися з різними класами учнів.
- створення та керування навчальними групами та класами – менеджмент школи має мати можливість створювати дисципліни, наповнювати їх навчальними матеріалами та ділитися з різними класами учнів.
- створення навчальних матеріалів – менеджмент школи має мати можливість створювати навчальні матеріали для окремих дисциплін. До навчальних матеріалів входять: лекції, практичні заняття та іспити.
- проходження навчальних матеріалів студентами – студенти, які були додані до класу мають мати можливість переглядати навчальні

матеріали з різних дисциплін та проходити їх. Проходження навчального матеріалу залежить від типу. Для лекцій, студенти можуть переглядати матеріали лекцій та позначати їх як переглянуті. Щодо практичних занять, то студенти можуть починати їх проходження, відповідати на окремі запитання, перевіряти свої відповіді на окремі запитання, а також завершувати свою спробу з переглядом усіх правильних та неправильних відповідей і загального балу за заняття. Широкі налаштування занять та їх завдань забезпечуватимуть багато схем проходження завдань для студентів. Наприклад, до кожного завдання можна додати підказки і пропонувати їх студентам у разі невірної спроби; відкорегувати дозволену кількість спроб на проходження завдання, виставити обмеження в часі, тощо. Це забезпечить цікавий досвід користування системою для студентів, а також дозволить краще оцінювати їх успішність. Проходження іспиту є ідентичним до проходження практичних занять, проте для них можуть бути визначені інші налаштування;

- керування щоденником студента – щоденник студента містить в собі інформацію про результати проходження навчальних матеріалів окремим студентом. Функціонал вимагає надання можливості перегляду щоденника як студентам, так і менеджменту школи, а також додавання нотаток.

Загалом, основні продуктові вимоги є чітко визначеними та зрозумілими, проте багато деталей потребують подальшого визначення шляхом уточнення або обговорення з ідейними засновниками системи та іншими частинами команди, що також ведуть активну роботу над системою MathLearning.

1.3 Створення плану подальшої розробки

На основі аналізу поточного стану системи та основних продуктових вимог, закладених в основу, було визначено наступний план розробки продукту MathLearning. Мета створення цього плану полягала в структуризації роботи для кращого її розуміння та оцінки, а також визначенні основних пріоритетів.

1. Об’єднати мікросервіси для керування школою та навчальними планами. Ці сервіси пов’язані між собою багатьма сутностями, наприклад сутність навчального плану має містити в собі посилання на школу, дисципліну та профіль користувача, що її створив. Таке об’єднання значно спростить процес розробки та зменшить зв’язаність мікросервісів.
2. Додати належну авторизацію на основі ролей профілів користувачів школи до усіх частин системи школи, щоб забезпечити безпеку та цілісність системи.
3. Визначити дизайн доменної структури даних, взаємозв’язки між сутностями та основні правила бізнес логіки. Продумати всі можливі нюанси, щоб зменшити фактор неточностей та їх виправлення на майбутнє.
4. Налаштувати створення профілів користувачів школи та керування ними.
5. Додати сутності для відображення дисциплін, логіку їх створення, редагування та отримання.
6. Додати сутності для відображення класів та груп студентів, логіку їх створення, редагування та отримання.
7. Додати сутності для відображення навчальних матеріалів, логіку їх створення, редагування та отримання. До цього пункту також входить створення окремих типів навчальних матеріалів: лекцій,

практичних занять та іспитів. Кожен тип має підтримувати гнучке налаштовування відповідно до потреб школи.

8. Додати логіку проходження студентами різних навчальних матеріалів, збереження їх прогресу та оцінювання. Забезпечити безпеку під час проходження практичних занять та іспитів для унеможливлення списування за рахунок взлому системи.
9. Додати сутність щоденника студента і автоматичне зберігання та відображення прогресу студента по різним навчальним дисциплінам. Додати опцію перегляду щоденників студентів менеджментом школи та додавання нотаток до окремих результатів школи.

Отже, цей план допоможе розробникам структуризувати їхню роботу над застосунком, що в свою чергу підвищить їх ефективність.

2 ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ

У роботі було використано ряд нових технологій, що є необхідними для виконання усіх пунктів поставленого плану. Усі нові інструменти та підходи до розробки мають на меті покращити загальну систему, пришвидшити процес роботи та полегшити її підтримку в майбутньому.

2.1 Використання Spring Cloud Configuration

Spring Cloud Configuration - це потужний інструмент, який спрощує управління зовнішніми конфігураціями для розподілених систем. Він забезпечує як серверну, так і клієнтську підтримку зовнішньої конфігурації в розподіленій системі, що є важливим для того, щоб підтримувати консистентність між різними середовищами.

Серверна архітектура Spring Cloud Configuration складається з Spring Cloud Config Server, який діє як центральне сховище конфігурацій, яке зазвичай підтримується системою контролю версій, наприклад, Git, або ж локальним сховищем файлів. Цей сервер підтримує широку кастомізацію конфігурацій, наприклад підтримує різні формати даних, такі як YAML, JSON та .properties файли, дозволяє різним середовищам (dev, test, prod) мати окремі конфігурації, а також динамічно оновлює конфігурації без перезапуску сервісів.

На стороні клієнта, кожен мікросервіс в розподіленій системі може отримати свою конфігурацію з сервера через кінцеві точки REST. Це дозволяє всім сервісам залишатися узгодженими та актуальними.

Переваги Spring Cloud Configuration полягають в:

- централізованому управлінні – всі конфігурації зберігаються в централізованому місці, що полегшує підтримку та оновлення налаштувань;

- консистентності – забезпечує узгодженість між усіма сервісами, надаючи конфчігурації, що відповідають конкретному середовищу;
- масштабованості – централізована конфігурація легко масштабується з додаванням нових сервісів до екосистеми;
- безпеці – конфіденційна інформація може бути зашифрована, що забезпечує безпечне зберігання конфіденційних даних.

Існують й альтернативні підходи, проте вони мають певні обмеження в порівнянні з Spring Cloud Configuration. Відомими є наступні альтернативи:

- зберігання конфігураційних файлів на кожному окремому сервері – це призводить до дублювання даних, підвищує ризики неузгодженості та створює додаткові складнощі в управлінні складними системами, що включають в себе велику кількість мікросервісів;
- використання змінних середовища – є незручним методом для керування складними системами, особливо при наявності різних конфігурацій в різних середовищах;
- використання сторонніх сервісів (наприклад, AWS Parameter Store, HashiCorp Vault) – збільшує затримку (latency), а також вимагає додаткового навчання для опанування та використання цих сервісів.

На фоні цих альтернатив Spring Cloud Configuration є ідеальним рішенням, адже воно повністю прибирає дублювання даних завдяки централізованому сховищу, забезпечує належну безпеку, підтримує різні середовища та є повністю інтегрованим в екосистему Spring, що спрощує процес навчання та зменшує затримки.

Використання Spring Cloud Configuration у проєкті MathLearning замінить ручний сервер конфігурацій, тим самим спростивши подальшу розробку та розгортання проєкту у хмарі.

2.2 Схеми міграцій з використанням Flyway

REST-додатки часто ітеративно розробляються для включення нових функцій або виправлення проблем. Ці зміни часто впливають на моделі даних. Схеми міграцій гарантують, що база даних залишається узгодженою з цими мінливими вимогами. Також вони служать єдиним джерелом правди в додатку, забезпечуючи видимість змін та їх послідовне застосування, що дозволяє безперешкодно співпрацювати над додатком декільком розробникам одночасно. Ще одне корисне застосування схем міграцій – створення відкату у разі помилкових змін або неочікуваних проблем. Це допомагає підтримувати цілісність даних і забезпечує шлях для безпечного відновлення.

Flyway - це популярний інструмент міграції баз даних з відкритим вихідним кодом, який спрощує і впорядковує контроль версій баз даних. Він зосереджений на міграції схем і сумісний з різними системами баз даних, такими як MySQL, PostgreSQL, Oracle та іншими. Flyway дозволяє командам відстежувати, керувати та застосовувати зміни в базі даних у послідовний та повторюваний спосіб, узгоджуючи стан бази даних з кодом програми. Flyway організовує зміни в базі даних в окремі сценарії міграції, що мають префікс з номером версії, наприклад, V1__Initial_Schema.sql або V2__Add_Users_Table.sql. Така система нумерації версій гарантує, що зміни застосовуються в правильній послідовності. Flyway підтримує таблицю бази даних для відстеження того, які скрипти вже були виконані. При запуску Flyway порівнює записані міграції з тими, що доступні в каталозі міграцій, і застосовує ті, які ще не були запущені. Цілісність кожного скрипта міграції підтримується за допомогою контрольних сум. Якщо скрипт змінюється після застосування, Flyway виявляє невідповідність і попереджає розробника, запобігаючи випадковій втраті або пошкодженню даних. Flyway підтримує таблицю бази даних для відстеження того, які скрипти вже були виконані. При запуску Flyway порівнює записані міграції з тими, що доступні в каталозі

міграцій, і застосовує ті, які ще не були запущені. Основні переваги системи міграцій баз даних Flyway: легкість інтеграції в Spring Boot, простота використання та велика онлайн спільнота.

Flyway - це надійний інструмент для управління міграцією баз даних в середовищі Spring Boot. Автоматизуючи процес, надаючи надійні контрольні суми та підтримуючи чітку структуру міграцій, він допомагає розробникам підтримувати узгодженість баз даних у різних середовищах. Це призводить до більш плавних і безпомилкових процесів розгортання і добре узгоджується з гнучкими і DevOps практиками.

Використання Flyway у поточній версії бекенду системи MathLearning зумовлено в першу чергу тим, що поточна система містить ряд неточностей в схемі бази даних. Ці неточності необхідно виправити, щоб мати змогу продовжувати подальшу розробку. Також використання інструменту для схем міграцій забезпечить узгодженість проєкту, а конкретно схеми бази даних, серед усіх розробників, що будуть працювати над проєктом в майбутньому.

2.3 Обробка винятків в Spring Boot

Обробка виключень є критично важливим компонентом при розробці REST-сервісів за допомогою Spring Boot. Він гарантує, що коли щось піде не так, система зможе коректно обробити помилку і надати клієнту чітку, зручну для нього відповідь. У цьому підмодулі розглядається використання глобального обробника винятків і важливість надійного управління помилками в REST-додатках.

Spring Boot пропонує потужний спосіб глобальної обробки виключень за допомогою анотації @ControllerAdvice. Цей підхід дозволяє розробникам створити центральну точку для обробки винятків, яка може застосовуватися до всіх контролерів у додатку. У межах цього обробника можна визначити різні методи за допомогою анотації @ExceptionHandler, кожен з яких

націлений на різні типи винятків. Таке налаштування допомагає підтримувати чистий код контролера і сприяє повторному використанню загальної логіки обробки винятків.

Належна обробка винятків має вирішальне значення з кількох причин:

- спілкування з користувачем: це допомагає спілкуватися з користувачами API, надаючи змістовні повідомлення про помилки, які можуть підказати їм, як продовжити або виправити їхні запити.
- ремонтопридатність: спрощує обслуговування програми, централізуючи логіку обробки помилок, роблячи кодову базу чистішою і простішою в управлінні.
- безпека: належна обробка помилок запобігає потраплянню конфіденційної інформації про помилки до клієнта, таким чином підвищуючи безпеку програми.
- надійність сервісу: завдяки ефективній обробці винятків додаток може залишатися працездатним навіть у разі виникнення непередбачуваних проблем.

Обробка винятків у Spring Boot REST додатках має важливе значення для побудови надійних, безпечних та зручних API. Використовуючи глобальний обробник винятків або досліджуючи альтернативні методи, розробники можуть гарантувати, що їхні додатки не тільки стійкі до помилок, але й надають чіткий і корисний зворотній зв'язок користувачам, покращуючи таким чином загальний користувацький досвід. Такий підхід підтримує довгострокову розробку, полегшуючи налагодження, підтримку та розширення програми.

Налаштування коректної системи обробки винятків в додатку є важливим на початковому етапі розробки, тому що це забезпечує прозорість системи та покращує взаємодію з клієнтською частиною. Саме тому цьому пункту було приділено особливу увагу під час роботи над системою MathLearning.

2.4 Налаштування безпеки в Spring Boot

Безпека є фундаментальним аспектом будь-якого REST-сервісу. Spring Boot забезпечує надійну систему безпеки за допомогою фреймворку Spring Security, який надає широкі можливості щодо захисту REST-сервісів. Безпека в Spring Boot управляється за допомогою фільтрів, які перехоплюють HTTP-запити, щоб застосувати перевірку автентичності та авторизації до того, як вони досягнуть контролера. Фільтри автентифікації гарантують, що запит містить дійсні облікові дані, а фільтри авторизації, з іншого боку, підтверджують, що автентифікований користувач має необхідні дозволи для доступу до запитуваного ресурсу. Також авторизація може здійснюватися на рівні окремих контролерів за допомогою об’єкту @PreAuthorize. Цей об’єкт надає можливість обмежувати доступ до контролера лише певним групам автентифікованих користувачів відповілно до їхніх ролей.

Налаштування загальних правил безпеки передбачає використання об'єкта HttpSecurity, де розробники можуть налаштувати механізми автентифікації, визначити правила авторизації для обмеження доступу на основі ролей, увімкнути захист від підробки міжсайтових запитів (CSRF) для запобігання несанкціонованим діям від імені користувачів, а також налаштувати Cross-Origin Resource Sharing (CORS), щоб контролювати, які зовнішні домени можуть отримати доступ до API.

Контекст безпеки у Spring Boot є ще одним фундаментальним аспектом, який містить дані автентифікації та авторизації поточного користувача. Цей контекст доступний у всьому додатку, що дозволяє розробникам отримувати інформацію про користувача, який увійшов в систему, та надані йому повноваження.

Система безпеки Spring Boot має численні переваги. Вона надає комплексні функції безпеки, які легко розширюються, що дозволяє налаштовувати їх відповідно до конкретних вимог додатків. Інтеграція з Spring

Boot є безперешкодною, що зменшує потребу в обширній конфігурації та шаблонному коді. Дотримуючись галузевих стандартів, Spring Security гарантує, що додатки захищені від поширених вразливостей безпеки. Завдяки великій спільноті підтримки та обширній документації, Spring Security є надійним вибором для захисту сучасних веб-додатків, гарантуючи, що вони не тільки безпечні, але й підтримувані та відповідають найкращим практикам безпеки.

Налаштування безпеки є надзвичайно важливим для будь якої системи, особливо для такої складної та масштабної, як система MathLearning, тому цьому аспекту було приділено особливу увагу в процесі розробки.

2.5 Проектування доменної області

Проектування домену – важливий аспект розробки REST-сервісу. Він передбачає визначення основних сутностей, зв'язків та поведінки, які представляють бізнес-логіку та структури даних додатку. По суті, це план, який описує, як дані організовані в системі, як з ними працювати, а також визначає основні бізнес правила.

Думати про дизайн домену на початку циклу розробки важливо з кількох причин. По-перше, це гарантує, що додаток побудований на міцному фундаменті, де модель даних точно відображає реальні сценарії, з якими він має працювати. Таке узгодження між моделлю предметної області та бізнес-вимогами зменшує ймовірність значного рефакторингу в подальшому, який може бути дорогим і трудомістким. По-друге, добре продуманий дизайн домену сприяє кращій комунікації між членами команди та зацікавленими сторонами, забезпечуючи чітке і спільне розуміння основних концепцій і структур системи. Це також сприяє узгодженості та багаторазовому використанню різних частин програми, що робить кодову базу більш зручною для підтримки та легшою для розширення в майбутньому.

Найкращі практики щодо дизайну домену включають наступні:

- чітке розуміння бізнес-вимог і контексту домену, що досягається завдяки тісній співпраці з експертами та зацікавленими сторонами. Це гарантує, що модель точно відображатиме основні аспекти домену.
- використання принципів доменно-орієнтованого проектування, такі як сутності, об'єкти-цінності, агрегати та репозиторії, щоб структурувати модель предметної області. Цей підхід допомагає створити модульний і цілісний дизайн, який може розвиватися відповідно до потреб бізнесу.
- визначення чітких меж для різних частин предметної області, відомих як обмежені контексти, щоб керувати складністю і підтримувати поділ проблем. Кожен обмежений контекст повинен мати свою власну модель і бути незалежно керованим.
- надання пріоритету простоті та уникнення надмірної інженерії. Максимально проста модель предметної області, яка при цьому відповідає бізнес-вимогам, допомагає підтримувати чітку і зрозумілу кодову базу.
- врахування майбутньої масштабованості та потенційного зростання обсягу даних і взаємодії користувачів. Це передбачає усвідомлений вибір структур даних, зв'язків і шаблонів доступу до них.

Spring Boot надає багато інструментів та фреймворків для ефективної реалізації доменної моделі. Spring Data JPA, наприклад, спрощує доступ до даних та їх збереження, дозволяючи розробникам зосередитися на логіці домену, а не на шаблонному коді. Дотримуючись усіх найкращих практик, розробники можуть створити надійний, гнучкий та підтримуваний дизайн домену, який слугуватиме міцною основою для їхнього REST-сервісу.

Система MathLearning має надзвичайно складну доменну систему завдяки амбітним цілям, що стоять в основі цього продукту. Тому правильний

підхід до розробки домену є неймовірно важливим елементом роботи над проєктом MathLearning, від якого залежить успішність продукту.

3 РОЗРОБКА СИСТЕМИ MATHLEARNING

3.1 Впровадження покращень до поточної системи

Роботу над проєктом MathLearning було почато з виправлення поточних недоліків та впровадження покращень, що мало на меті покласти правильний початок для подальшої розробки, а також визначити основні правила та практики для наслідування. Список впроваджених покращень складається з наступних пунктів:

- запровадження Spring Cloud Configuration;
- впровадження інструменту для схем міграцій Flyway;
- налаштування обробки помилок;
- налаштування Spring Security;
- об’єднання мікросервісів школи та завдань.

Запровадження Spring Cloud Configuration включало в себе заміну існуючого серверу конфігурацій на новий з використанням інструменту Cloud Config, перенесення усіх конфігураційних даних на новий сервер та заміну усього клієнтського коду на новий. Останній пункт виявився найпростішим, адже Spring Cloud Configuration є повністю інтегрованим в Spring Boot. Таке впровадження значно спростило клієнтський код, а також надало нові можливості у використанні, наприклад розділення конфігурацій на різні середовища: розробки, тестування та продакшену.

Впровадження інструменту Flyway було необхідним для видалення таблиць для зберігання лекцій, іспитів та навчальних планів з метою їх заміни на нові, удосконалені сутності з новими залежностями, а також редагування таблиць для зберігання практичних занять та завдань з метою їх розширення новими атрибутами. Вбудованого інструментарію Spring Boot не було достатньо для виконання цих завдань нормальним способом, адже він не підтримує ручного редагування таблиць, тобто видалення непотрібних рядків

є неможливим – видалення цих атрибутів з коду не несе за собою їх видалення з бази даних. Фреймворк Flyway було впроваджено наступним чином: при запуску програми після конфігурації бази даних MySQL автоматично створюється об’єкт Flyway, куди передається щойно створений об’єкт бази даних, а також встановлюється параметр baselineOnMigrate = true, який забезпечує ініціалізацію схем міграцій у випадку, коли в проєкті вже є створена база даних з таблицями. Далі викликається метод .migrate(), який виконує усі ті міграційні скрипти, які ще не були виконані. У разі помилки під час міграції, Flyway надає інструмент repair для відкату змін і нової спроби проведення міграцій.

Також використання Flyway є важливим з тієї причини, що над проєктом ведеться і буде в майбутньому вестися робота декількома людьми. Flyway дозволяє досягти максимальної узгодженості між усіма членами команди розробників та усіма середовищами.

Налаштування обробки помилок було вирішено закласти з самого початку для внесення прозорості та ясності в проєкт. Завдяки чітким помилкам у розробників виникає краще розуміння бізнес логіки та неочікуваних кейсів, що можуть виникнути під час її виконання.

Налаштування Spring Security включало в себе наступні етапи:

- налаштування автентифікації користувачів за профілем школи. Тобто, крім автентифікації особистого токену користувача, також отримуються дані про профіль поточного користувача для визначення його школи, ролі та ідентифікатора профілю. Така автентифікація відбувається за рахунок прив’язки кожного профілю користувача до акаунту. Отримуючи токен, ми також отримуємо ідентифікатор акаунту, який дозволяє підтягнути дані про профіль користувача. Дані з профілю користувача зберігаються в об’єкті UserDetails, що належить до об’єкту UserAuthentication, який в свою чергу зберігається в контексті запиту і є доступним на всіх рівнях програми. Весь процес автентифікації відбувається в фільтрах, що

надає інструмент Spring Security. Він дозволяє створювати власні фільтри шляхом наслідування базового класу та додавати їх до ланцюга обробки запиту. Базовий клас Filter має єдиний метод для імплементації – doFilter(ServletRequest, ServletResponse, FilterChain). В попередній версії додатку використовувався окремий http header для зберігання ідентифікатору профіля користувача, проте такий підхід не є повністю безпечним чи ефективним.

- налашування правил авторизації для окремих ендпойнтів. Кожен ендпойнт має різні правила доступу, тому важливо було акуратно визначити правила авторизації для кожного з них. Наприклад, не можна давати можливість студентам створювати дисципліни, а вчителям втручатися в процес проходження завдань студентами. Налаштування окремих правил доступу було виконано завдяки декоратору @PreAuthorize, який надає широкі можливості в конфігуруванні доступу на основі контексту запиту, де зберігається інформація про автентифікованого користувача. Ми можемо налаштовувати доступ на основі ролі поточного користувача за допомогою @PreAuthorize(“hasAuthority(‘RoleName’)” або @PreAuthorize(“hasAnyAuthority(‘RoleName1’, ‘RoleName2’, …)” для налаштування доступу за декількома ролями. Також було використано @PreAuthorize(“!isAnonymous()”) для фільтрації анонімних користувачів для загально доступних запитів.
- налаштування окремих частин бізнес логіки на основі контексту запиту та даних поточного автентифікованого користувача. Деяка бізнес логіка, така як, наприклад, створення профілів користувачів потребувала додаткового налаштування на основі ролей користувачів. Для цього було використано контекст запиту, який заповнювався на етапі автентифікації, шляхом отримання даних про профіль поточного користувача з UserDetails та передачі цих даних до методів бізнес логіки.

Об’єднання мікросервісів школи та завдань було зумовленим великою зв’язаністю цих двох сервісів між собою. Такий крок значно спростить реалізацію плану розробки системи. Технічних складнощів під час виконання цього пункту не виникло.

3.2 Дизайн домену системи

Основна доменна структура проєкту MathLearning – School – містить в собі мета інформацію про школу і слугує основним ідентифікатором для усіх інших структур. До мета інформації школи також відноситься SchoolAddress, яка є окремою сутністю і зберігає інформацію про адресу школи. Сутність UserProfile відображає окремого користувача в системі школи. Основне призначення цієї сутності – ідентифікація користувачів та визначення їх ролей.

Для групування студентів існують сутності SchoolClass та SchoolGroup.

Сутність EducationPlan відображає дисципліну школи, яка містить список навчальних матеріалів (лекції, практичні заняття та іспити), а також загальну інформацію про дисципліну. Сутність ClassEducationPlan є копією EducationPlan, яка створюється для кожного шкільного класу та зберігає в собі прогрес усього класу по дисципліні. Такий підхід дозволяє зберігати статичні навчальні плани дисциплін в системі школи, а також проводити одну й ту саму дисципліну для декількох класів зі збереженням та відслідковуванням прогресу й внесенням корективів до плану незалежно один від одного.

Сутність Lecture відображає лекцію в навчальному плані системи й складається з списку матеріалів лекції – LectureMaterial. Кожен окремий матеріал лекції може бути текстом, відео або картинкою. Такий підхід додає гнучкості системі, дозволяючи налаштування різних підходів до відображення лекції на клієнтській частині застосунку, а також забезпечує простоту у додаванні змін до матеріалів лекції у майбутньому – можна з легкістю додавати нові типи матеріалів лекції, наприклад інтерактивні запитання або

опитування без великих змін до структури. StudentLecture є структурою, що поєднує лекції та студентів й має на меті зберігання прогресу студентів під час перегляду ними лекцій. Для кожного окремого студента створюється ця сутність після того, як він почав перегляд лекції. В цій сутності можна зберігати як загальний стан перегляду лекції студентом: переглянув або не переглянув, так і поступовий прогрес студента по перегляду лекції, або ж іншу інформацію про перегляд, наприклад запитання студента до лекції.

Сутність Exercise відображає практичне заняття як один з типів навчального матеріалу. Ця сутність зберігає в собі статичні дані про заняття: загальну інформацію – назву, опис, мету; налаштування – відведений час, формат здачі тощо; список вправ для виконання. Сутність вправи представляє клас Task, в якому зберігається умова завдання, тип, додаткові налаштування, а також список відповідей Answer. Для відображення відповідей було обрано саме модель масиву, щоб досягти додаткової гнучкості у формуванні завдань – завдяки цьому вчителі можуть створювати різноманітні завдання: з вибором правильної відповіді або з ручним введенням відповіді. Тип завдання зберігається в TaskType enum. Додаткові налаштування завдань та занять включають у себе наступні:

- максимальна оцінка;
- вага завдання;
- можливість переглядати правильну відповідь до завершення виконання;
- можливість відповісти ще раз після надання неправильної відповіді;
- кількість доступних спроб для відповіді (активна лише при включенні попереднього налаштування);
- підказка;
- кількість знятих балів за використання підказки (може бути 0).

Для відображення прогресу студента у виконанні завдань було використано схожий підхід, що й з лекціями – було створено сутність StudentExercise та StudentTask для зберігання прогресу студента по всьому

заняттю та окремим завданням відповідно. StudentExercise містить в собі дані про те, чи завершив студент виконання заняття, або його прогрес, а також яку оцінку за нього отримав. StudentTask зберігає інформацію про відповідь студента на конкретне завдання, його правильність, кількість невдалих спроб виконання завдання та чи завершив студент виконання завдання. Сутність StudentAnswer зберігає відповідь студента на завдання. StudentTask містить масив відповідей на той випадок, якщо студент відповість на питання не з першої спроби. Такий підхід до зберігання прогресу студентів дозволяє легко відслідковувати роботу учнів на заняттях, а гнучкість самих вправ відкриває вчителям різноманітні можливості для оцінювання студентів.

Іспит може бути представленим сутністю Exercise, адже згідно продуктовим вимогам іспит не відрізняється від завдання. Єдина різниця заключається в налаштуванні, тому окремої сутності іспиту не було розроблено для уникнення дублювання даних.

Сутність StudentDiary є сутністю агрегатом та представляє щоденник студентів. В цю сутність входять переглянуті лекції та виконані вправи учня за весь період часу. Ця сутність дозволяє легко переглядати успішність та активність учнів за увесь час або за певний період.

Отже, було продумано доменну систему додатку та взято до уваги всі продуктові вимоги. Така система забезпечить легкість та прозорість у написанні коду та зменшить кількість помилок, що стосуються логіки додатку у майбутньому. Схематичне зображення дизайну доменної системи представлено у Додатку А.

3.3 Написання коду додатку MathLearning

Після запровадження покращень до системи MathLearning та дизайну доменної частини додатку, було написано код, що виконує поставлені задачі

відповідно до створеного плану розробки. Під час розробки було використано інструменти Spring Boot для написання RESTful застосунків, а саме:

- Spring MVC – це основний фреймворк для створення веб-додатків та RESTful веб-сервісів у Spring Boot. Він надає різні анотації, такі як @RestController, @RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping тощо, для зіставлення HTTP-запитів з методами обробників;
- Spring Data JPA – потужний інструмент, який забезпечує легку інтеграцію з JPA (Java Persistence API) і підтримує CRUD-операції через інтерфейси репозиторіїв;
- Jackson – інструмент для серіалізації та десеріалізації JSON. Він дозволяє конвертувати об'єкти Java в JSON і навпаки, що важливо для REST API, які обмінюються даними JSON;
- додаткові анотації Spring Boot – @Component, @Service, @Repository та @Controller для автоматичного створення бінів та ін'єкції залежностей для керування компонентами програми.

Для побудови бекенд частини додатку було використано багаторівневу архітектуру, суть якої заключається в тому, що кожен рівень програми має конкретну зону відповідальності і існує незалежно від інших рівнів. Такий підхід є дуже поширеним у розробці серверних додатків, адже він дозволяє підвищити модульність, зрозумілість і масштабованість коду. Кожен рівень може бути розроблений та протестований окремо, що спрощує підтримку та розвиток системи. Також багаторівнева архітектура забезпечує кращу організацію коду та розподіл відповідальності між різними компонентами додатку. Код серверу додатку MathLearning був поділений на 4 основні рівні – рівень контролера, рівень сервісів, рівень репозиторіїв та доменний рівень.

Рівень контролера відповідає за роутинг ендпойнтів серверу, обробку вхідних даних, отримання додаткової інформації з контексту, передачу запиту на наступний рівень сервісів та обробку відповіді й надсилання її назад

клієнту. Для роутингу та парсингу вхідних даних було використано SpringMVC, для отримання додаткової інформації про користувача – контекст запиту від Spring Security. Передача даних на наступний рівень відбувається за допомогою техніки Dependency Injection (DI). DI, або ін’єкція залежностей, полягає в тому, що об'єкти отримують свої залежності від зовнішніх джерел замість створення їх самостійно. Це дозволяє більш гнучко керувати залежностями та спрощує тестування компонентів, оскільки їх можна легко замінити на мок-об'єкти. Для кожної окремої частини логіки додатку було створено окремий контролер, відділений від інших базовим шляхом. Доступ до усіх контролерів відбувається через базовий шлях /api. Розлянемо деякі контролери:

- ProfileController (api/profile) – відповідає за створення та отримання профілів користувачів школи, а також за підвищення менеджменту школи на іншу посаду;
- EducationPlanController (api/plans) – відповідає за CRUD операції над навчальними планами;
- ExerciseController (api/exercises) – відповідає за адміністрування практичних занять: їх створення, редагування, видалення, створення та редагування окремих завдань для занять. Також має ендпойнт для отримання занять для учнів;
- StudentExerciseContoller (api/student-exercises) – контролер для учнів, який надає наступний функціонал по проходженню завдань та зберігання прогресу:
    - створення копії Exercise для окремого студента задля зберігання його прогресу. Кількість можливих копій занять залежить від налаштування кількості спроб – якщо для заняття виставлена лише одна можлива спроба, то учень може створити копію лише один раз;
    - надання відповіді на окреме завдання. Відбувається зберігання відповіді до бази даних. Залежно від налаштувань завдання

кожна відповідь може бути або відхилена (якщо максимальну кількість спроб використано), або прийнята;
  - перевірка виконаного завдання. В залежності від налаштування заняття може або повернутися результат і правильна відповідь, або помилка про те, що відповідь не може бути отримана;
  - закінчення виконання заняття – завершується здача усіх завдань, кожне з них перевіряється на правильність та визначається загальний бал учня;
  - отримання окремого заняття учня або їх список.
- StudentDiaryController (api/student-diary) – має ендпойнти для отримання щоденника або списку щоденників. Якщо запит відбувається від імені учня, то повертається лише щоденник цього учня;
- LectureController (api/lectures) – відповідає за адміністрування лекцій школи;
- StudentLectureController (api/student-lectures) – відповідає за функціонал відслідковування прогресу учнів по лекціям. Має ендпойнти для створення копії лекції для учня й завершення перегляду лекції.

Отже, даний рівень відділяє усю іншу логіку додатку від обробки запитів й бере цю роль на себе. Деякі фрагменти коду наведені у Додатку Б, а приклади виконання запитів – у Додатку В.

Рівень сервісної логіки відповідає за основну логіку додатку – тут визначені основні методи та засоби, що маніпулюють даними. Попередній рівень контролерів передає процес виконання коду саме на цей рівень. Також цей рівень виступає у ролі розділювача між доступом до бази даних та рівнем контролерів. Для доступу до бази даних було також використано ін’єкцію залежностей, за допомогою якої було надано доступ до окремих репозиторіїв сервісному коду. До визначених сервісів в додатку входять наступні:

- SchoolService – відповідає за отримання даних про школу, менеджмент школи та учнів;
- UserProfileService – надає логіку створення та редагування профілів користувачів школи, а також логіку підвищення керівництво школи на іншу посаду;
- EducationPlanService – створення та редагування навчальних планів;
- ExerciseService – надає функціонал створення та налаштування практичних занять;
- TaskService – надає функціонал створення та налаштування завдань для практичних завдань.
- StudentExerciseService – відповідає за наступний функціонал:
    - створення копій практичних занять для студентів. Тут відбувається перевірка кількості дозволених спроб виконання заняття й кількість уже пройдених спроб студентом;
    - отримання практичних занять – повертаються дані про прогрес студента, а також умова та можливі відповіді завдання. Важливим моментом є приховування даних про правильність відповідей у респонсі, щоб ніхто з учнів не міг списати правильну відповідь. Також, якщо тип завдання – з відкритою відповіддю, то не надається жодна можлива відповідь;
    - надання відповідей на завдання. Відбувається зберігання відповіді до бази даних. Якщо ж студент вичерпав можливу кількість відповідей, йому повернеться помилка;
    - перевірку відповідей учня на окремі завдання. Відбувається співставлення відповіді студента й правильної відповіді й повертається результат. Якщо ж в налаштуванні заняття виставлено заборону на перегляд відповідей на окремі завдання під час їх виконання, то користувачу повернеться помилка про це;

    - завершення занять – перевірка правильності усіх виконаних завдань та виставлення загальної оцінки.
- LectureService – відповідає за CRUD операції над лекціями та матеріалами до лекцій;
- StudentLectureService – відповідає за зберігання прогресу студента по лекціях та його відображення;
- StudentDiaryService – відповідає за отримання та створення щоденників учнів.

Отже, рівень сервісної логіки визначає основну бізнес логіку додатку та є ізольованим від інших сервісів для забезпечення модульності. Деякі фрагменти коду наведені у додатку Б.

Для реалізації рівня репозиторіїв було використано Spring Data JPA – цей інструмент полегшує визначення CRUD операцій над сутностями, а також надає можливість створення додаткових SQL запитів для більш складної логіки. Завдяки виділенню логіки роботи з базою даних в окремий рівень, інші рівні додатку не є прив’язаними до деталей реалізації. Тобто, зміна цих деталей, наприклад зміна системи зберігання даних, не впливатиме на інші рівні, і, відповідно, така зміна потребуватиме менше зусиль для впровадження. В рамках написання коду було визначено базові репозиторії для усіх можливих сутностей, а саме: UserProfileRepository, SchoolRepository, SchoolDisciplineRepository, EducationPlanRepository, ExerciseRepository, StudentExerciseRepository, TaskRepository, StudentTaskRepository, LectureRepository, LectureMaterialRepository, StudentLectureRepository, StudentDiaryRepository. Для більш складних випадків, де логіка виходить за рамки звичайних CRUD операцій, було написано власні методи всередині базових репозиторіїв. Наприклад, пошук за додатковими параметрами, фільтрація даних, отримання даних з декількох таблиць одночасно (join-и), тощо. Деякі фрагменти коду наведені у додатку Б.

Рівень домену наслідує дизайн доменної системи, проведений раніше. Для кожної сутності було визначено свій клас та методи, що визначають дозволені маніпуляції над даними. Розглянемо деякі класи:

- UserProfile – містить загальну інформацію про профіль користувача, посилання на школу та роль;
- Role – enum, що визначає список дозволених ролей в системі. Має методи, які визначають ролі адміністрації, ролі вчителювання та правила створення інших користувачів (чи може користувач з однією роллю створити користувача з іншою);
- Exercise – відображає сутність практичного заняття або екзамену відповідно до значення поля isExam. Також містить загальну інформацію про заняття та налаштування. До налаштувань входять: maxNumberOfAttempts (максимальна кількість дозволених спроб виконання), timeLimit (часовий ліміт виконання), canSeeTaskResultsBeforeComplete (чи є доступ до перегляду правильності відповіді на індивідуальні завдання під час проходження заняття), maxGrade (максимальна оцінка за заняття).
- Task – відображає сутність завдання до практичного заняття або іспиту. Зберігає тип завдання (enum TaskType), умову завдання в декількох форматах (текст, latex, посилання на медіа файл), значимість завдання (weight) у відсотках відносно всього заняття, або ж максимальний бал за заняття (maxGrade), підказку (tip) та список можливих відповідей до завдання (клас Answer). Також цей клас містить в собі додаткові налаштування: maxNumberOfAttempts (кількість дозволених спроб на виконання завдання; грає роль лише якщо значення налаштування заняття canSeeTaskResultsBeforeComplete є true), minusGradeForTip (кількість віднятих балів за використання підказки). Також цей клас містить в собі методи, що визначають основні правила домену: метод check перевіряє правильність відповіді користувача, метод

listAvailableAnswers повертає список можливих відповідей користувачу на основі типу завдання (для завданню з відкритою відовіддю буде повернутий порожній список, для завдання з вибором правильної відповіді – список усіх можливих відповідей без зазначення правильної відповіді;

- TaskType – enum, що визначає доступні типи завдань:
    - YES_NO – завдання, де потрібно вказати «Так» або «Ні»;
    - SELECT_OPTION – завдання, де потрібно обрати правильну відповідь зі списку можливих;
    - INPUT – завдання, в якому потрібно ввести правильну відповідь самостійно.
- Answer – клас, що відображає відповідь на запитання. Зберігає в собі власне текст відповіді (answer), а також правильність наданої відповіді (isCorrect). Містить основний метод, який визначає чи вхідна відповідь є правильною, тобто співпадає з answer;
- StudentExercise – клас, що з’єднує учнів та завдання та представляє результати учнів по завданням. StudentExercise містить в собі посилання на заняття (Exercise), студента (UserProfile) та список виконаних завдань (StudentTask). Також цей клас містить в собі поля, що відображають прогрес студента: attemptNumber (номер спроби виконання), isCompleted (чи завершив студент виконання завдання), progress (прогрес у відсотках по завданню), grade (виставлена оцінка за завдання), а також методи, що визначають основну логіку над сутністю – calculateGrade (приватний метод для вирахування оцінки), calculateProgress (приватний метод, який визначає прогрес студента по заняттю на основі кількості виконаних завдань) та submit (метод, який означає закінчення виконання заняття та підраховує усі потрібні значення й позначає заняття як виконане);
- StudentTask – клас, що з’єднує завдання з практичних занять та студентів та зберігає результати учнів у виконанні окремих завдань.

Містить посилання на завдання (Task), студента (UserProfile) та StudentExercise, якому належить. Також зберігає в собі поля, що відображають відповіді учня, правильність відповіді (isCorrect), кількість невдалих спроб (numberOfFailedAttemts) та факт використання підказки (hasUsedTip). Для позначення завдання як виконане використовується метод complete, що перевіряє правильність виконання та позначає завдання як виконане на основі налаштувань

- StudentDiary – клас, що агрегує в собі усі виконанні заняття (StudentExercises) та переглянуті лекції (StudentLecture) учнів.

Крім основних рівнів реалізації коду в додатку MathLearning, були використанні також додаткові компоненти, що покращували роботу застосунку. Компонент SecurityConfiguration відповідає за налаштування безпеки серверу; клас UserAuthenticationFilter слугує middleware, який авторизує профіль та акаунт користувача, що виконує запит; пакет utils.exceptions зберігає усі можливі винятки програми, тим самим створюючи єдине місце, куди розробники можуть звернутися для перегляду усіх очікуваних, але негативних кейсів поведінки програми; GlobalExceptionHandler – засіб Spring Boot, який дозволяє визначати загальні правила обробки помилок для всього додатку; моделі DTO, які дозволять фільтрувати дані, котрі повертають клієнту для кожного запиту, аби не повертати зайву, або недоступну інформацію.

Отже, було розроблено основні класи, що відображають доменну систему та визначено методи для маніпуляції над ними згідно рекомендованих практик, наведених в Domain Driven Design [9]. Деякі приклади виконання запитів наведені у Додатку В.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скуратівський М. Ю. Хмарна система MathLearning: імплементація продуктових вимог та документації: Магістерська робота. НаУКМА. Київ, 2023. 60 с.
2. Java from EPAM: навч.-метод. посібник / І. М. Блінов, В. С. Романчик, 2020.
3. Mate Academy Документація. Mate Academy. URL: https://blog.mate.academy/java-development/java-spring-hibernateframeworks/.
4. Microsoft документація. Microsoft Azure. URL: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/what-is-java-spring-boot/.
5. Spring документація. *Spring*. URL: https://spring.io/.
6. Spring Security. *Spring*. URL: https://spring.io/projects/spring-security.
7. Flyway документація. *Flyway*. URL: https://flywaydb.org/.
8. Mastering Exception Handling in Spring Boot: A Comprehensive Guide. *Naveen Metta*. URL: https://naveen-metta.medium.com/mastering-exception-handling-in-spring-boot-a-comprehensive-guide-fa3f916d1981.
9. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional, 2003. 560 p.
10. Larsson M. Microservices with Spring Boot and Spring Cloud. 2nd ed. Packt Publishing, 2021. 774 p.

ВИСНОВКИ

У цій курсовій роботі було проаналізовано всі продуктові вимоги до продукту MathLearning – сервісу для інтерактивного вивчення математики, створено детальний план розробки бекенд частини проєкту, який включав в себе впровадження покращень та загальних практик до поточної кодової бази, дизайн доменної системи застосунку та власне написання коду.

Для аналізу продуктових вимог було розглянуто магістерську роботу Скуратівського М. Ю. на тему «Хмарна система MathLearning: імплементація продуктових вимог та документації» [1], а також детально розписано всі пункти й доповнено їх новими деталями по ходу виконання роботи. Також було вивчено додаткові наукові джерела по темах Spring та Spring Boot, що забезпечило якісно написаний код.

В результаті роботи було реалізовано основну частину бекенд серверу для продукту MathLearning та закладено основні принципи розробки для подальшої розробки.

Рекомендації для подальших досліджень:

- тестування веб-застосунків. Тестування забезпечує надійність веб-застосунків під час активної фази розробки, тому його дослідження та впровадження буде корисним та цікавим для подальших досліджень;
- розширення функціональності. Застосунок MathLearning знаходиться в активній фазі розробки, і розширення його можливостей є обов’язковим елементом цього стану, тому наступні дослідження можуть зосередитися на уточненні всіх продуктових вимог, комунікації з іншими частинами команди для впровадження нового функціоналу до системи.

ДОДАТОК А

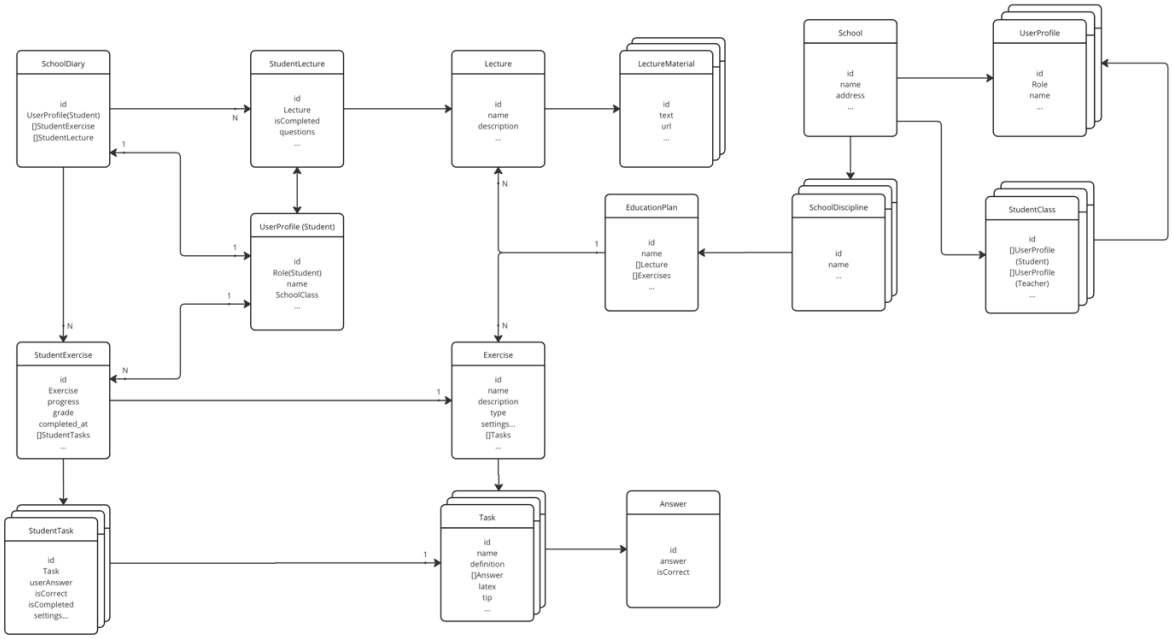


Рисунок 1. Умовна схема доменної моделі додатку

ДОДАТОК Б

```
22  @PublicApi
23  @RestController
24  @Api(tags = "StudentExercise")
25  @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
26  public class StudentExerciseController {
        10 usages
27      StudentExerciseService studentExerciseService;
28
        new *
29      public StudentExerciseController(StudentExerciseService studentExerciseService) {
30          this.studentExerciseService = studentExerciseService;
31      }
32
        new *
33      @GetMapping("/student-exercises")
34      @PreAuthorize("!isAnonymous()")
35      public ResponseEntity<List<StudentExerciseDTO>> getStudentExerciseById(
36              @RequestParam(value = "studentId", required = false) Long studentId
37      ) {
38          var isStudent = SecurityUtils.getUserAuthentication().getDetails().getProfile().getRole().equals("Student");
39          var accountId = SecurityUtils.getUserAuthentication().getPrincipal();
40          if (isStudent) {
41              studentId = accountId;
42          }
43
44          List<StudentExercise> exercises = studentExerciseService.listStudentExercises(studentId);
45
46          return ResponseEntity.ok(exercises.stream() Stream<StudentExercise>
47                  .map(StudentExerciseDTO::fromEntity) Stream<StudentExerciseDTO>
48                  .collect(Collectors.toList()));
49      }
50
```

Фрагмент коду 1. Контролер, що відповідає за роботу з StudentExercise та StudentTask, а також метод для отримання списку занять студента

```
10  public interface StudentLectureRepository extends JpaRepository<StudentLecture, Long> {
        1 usage  new *
11      @Query("SELECT sl FROM StudentLecture sl WHERE sl.lecture.id = :lectureId AND sl.studentId = :studentId")
12      Optional<StudentLecture> findByLectureIdAndStudentId(long lectureId, long studentId);
13
        1 usage  new *
14      @Query("SELECT sl FROM StudentLecture sl WHERE sl.studentId = :studentId")
15      List<StudentLecture> findAllByStudentId(long studentId);
16  }
17
```

Фрагмент коду 2. Репозиторій StudentLectureRepository для роботи з сутністю StudentLecture в базі даних

```java
23  @Service
24  @RequiredArgsConstructor
25  @FieldDefaults(makeFinal = true, level = lombok.AccessLevel.PRIVATE)
26  public class ExerciseService {
27      ExerciseRepository exerciseRepository;
28      TaskRepository taskRepository;
29      EducationPlanRepository educationPlanRepository;
30
        1 usage  new *
31      public Optional<Exercise> getExerciseById(long id) { return exerciseRepository.findById(id); }
34
        1 usage  new *
35      public List<Exercise> listExercises() { return exerciseRepository.findAll(); }
38
        1 usage  new *
39      @Transactional
40      public Exercise createExercise(CreateExerciseDTO exerciseDTO) throws EducationPlanNotFoundException {
41          Exercise exercise = new Exercise();
42          exercise.setName(exerciseDTO.getName());
43          exercise.setDescription(exerciseDTO.getDescription());
44          exercise.setIsPublic(exerciseDTO.getIsPublic());
45
46          // Fetch the EducationPlan entity using the provided id
47          Optional<EducationPlan> educationPlan = educationPlanRepository.findById(exerciseDTO.getEducationPlanId());
48          if (educationPlan.isEmpty()) {
49              throw new EducationPlanNotFoundException("Education plan not found");
50          }
51          exercise.setEducationPlan(educationPlan.get());
52
53          return exerciseRepository.save(exercise);
54      }
```

Фрагмент коду 3. ExerciseService, його залежності та приклад методу створення нової сутності Exercise.

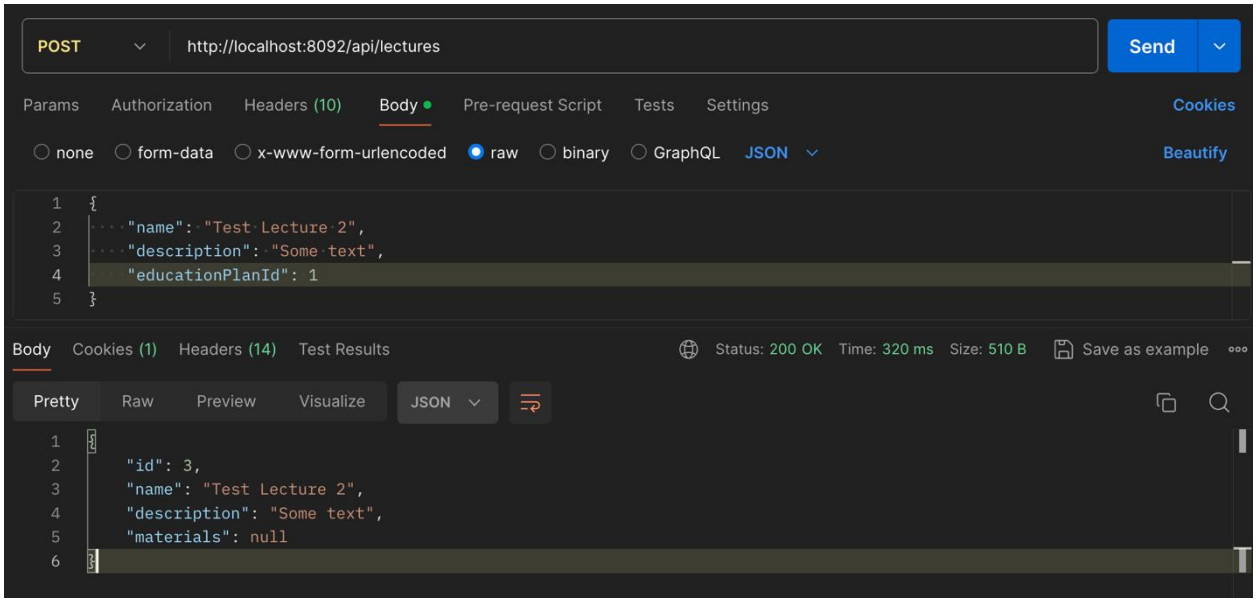
```java
18  @Entity
19  @Table(name = "student_exercises")
20  public class StudentExercise {
21
22      @Id
23      @Column(name = "id")
24      @GeneratedValue(strategy = GenerationType.IDENTITY)
25      Long id;
26
27      @Column(name = "student_id", nullable = false)
28      Long studentId;
29
30      @ManyToOne
31      @JoinColumn(name = "student_diary_id")
32      @JsonIgnore
33      StudentDiary studentDiary;
34
35      @ManyToOne
36      @JoinColumn(name = "exercise_id", nullable = false)
37      Exercise exercise;
38
39      @OrderBy("id ASC")
40      @OneToMany(mappedBy = "studentExercise", fetch = FetchType.LAZY, cascade = CascadeType.ALL, orphanRemoval = true)
41      @JsonManagedReference
42      Set<StudentTask> studentTasks;
```

Фрагмент коду 4. Клас сутності StudentExercise та деякі його поля та залежності

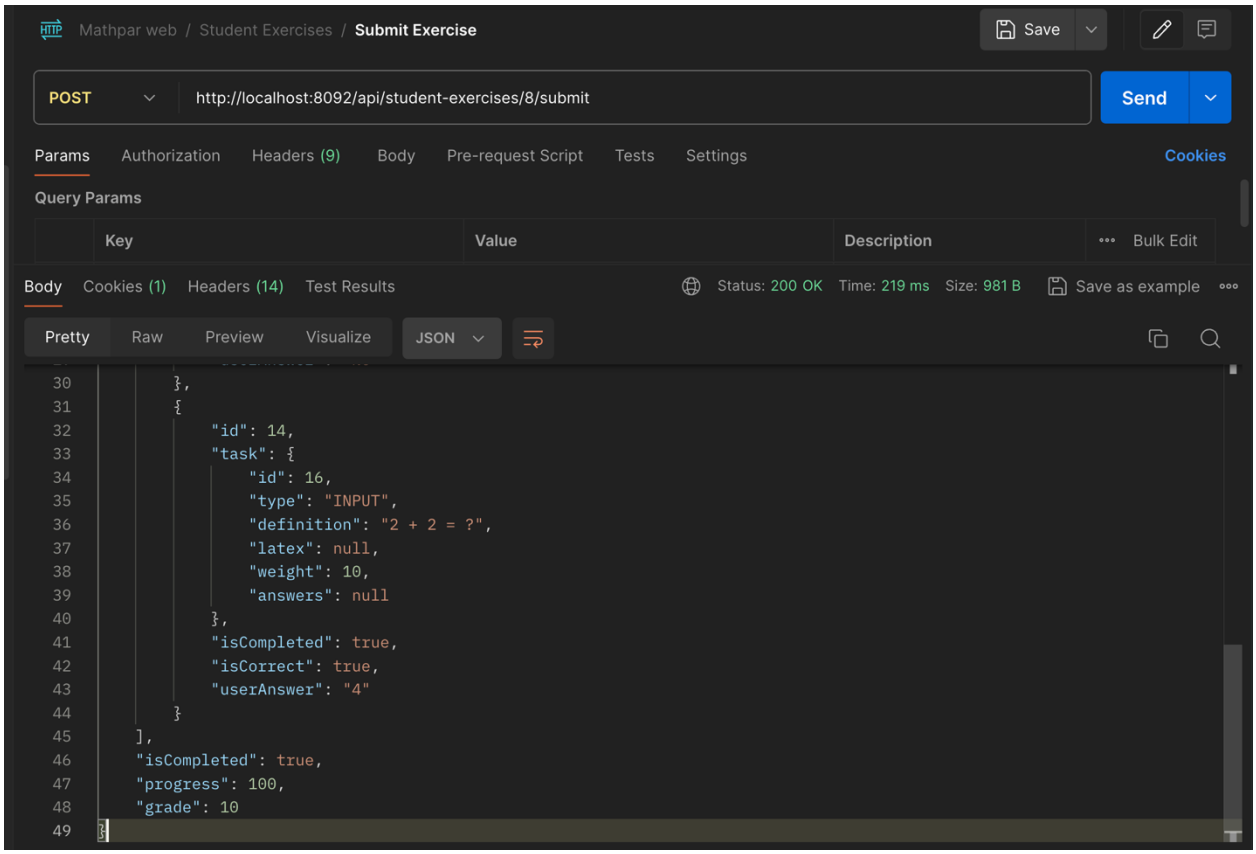
```
28      @Override
29      public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
30          var context = SecurityContextHolder.getContext();
31          var token = ((HttpServletRequest) request).getHeader(Constants.TOKEN_HEADER_NAME);
32
33          UserAuthentication userAuthentication = validateToken(token);
34          if (userAuthentication == null){
35              context.setAuthentication(new AnonymousAuthenticationToken( key: "anonymous",  principal: "anonymous", List.of(new SimpleGrantedAuthority( role: "ROLE_ANONYMOUS"))));
36          } else {
37              context.setAuthentication(userAuthentication);
38          }
39
40          chain.doFilter(request, response);
41      }
```

Фрагмент коду 5. Власний фільтр, який автентифікує профіль та акаунт користувача

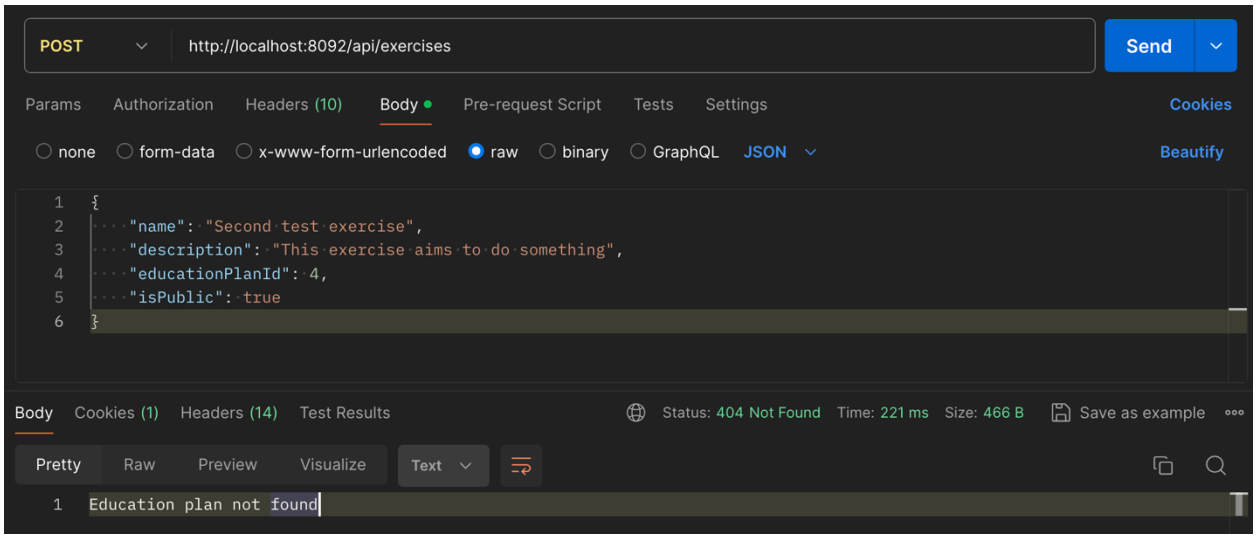
ДОДАТОК В



Приклад 1. Виконання запиту на створення лекції використовуючи сучасний інструмент Postman



Приклад 2. Результат виконання запиту завершення виконання практичного заняття студентом



Приклад 3. Приклад повернення очікуваної помилки у разі некоректних вхідних даних. Навчального плану з ідентифікатором 4 не існує в системі