Основною рисою сучасних веб-технологій є кросс-платформенність та здатність надавати користувачу доступ до персоналізованого контенту на різних пристроях з різними операційними системами, зберігаючи при цьому цілісність користувацького досвіду та універсальну доступність.

Для потреб навчальних закладів було розроблено багато продуктів, призначених для тестування знань студентів. Але переважна більшість цих продуктів орієнтована на роботу в класі та мають за платформу настільні операційні системи. Недоліком такого підходу є потрібність в установці спеціалізованого програмного забезпечення, прив’язка користувача системи до певної програмної та апаратної платформи та обмеження доступності системи поза навчальним класом.

В даній дипломній роботі розробляється система тестування знань студентів, яка має розширити можливості дистанційного доступу як викладачів так і студентів до створення та проходження тестів. Це розширяє можливості викладачів та студентів, збільшує кількість користувацьких сценаріїв, дозволяє ефективніше використовувати позакласний час для підготовки та самого тестування. Це має особливу важливість для дистанційного навчання, яке в останній час набуває все більшої та більшої популярності.

1. **Система тестування знань**
   1. **Аналіз предметної області та постановка задачі**

Предметною областю є тестування знань. Тестування знань у навчальному процеси призначено, перед усім, для вирішення завдань діагностики та навчання. Діагностична функція полягає у визначенні ступеню оволодіння студентами знаннями та навичками, які вони вивчають. Ця функція тестування є основною. Навчальна функція тестування полягає у мотивації студента до більш активного та поглибленого вивчення навчального матеріалу. Тестування в якості засобу контролю знань має такі переваги:

* Стандартизація процедури контролю
* Неупередженість та об’єктивність як процедури контролю, так і оцінки її результатів
* Можливість оцінювати знання як по певній частині курсу, так і по курсу в цілому, на відміну від усного чи письмового екзамену
* Можливість кількісної оцінки знань, точнішої за традиційну чотирибальну та, навіть, дванадцятибальну шкалу.
* Економія часу та коштів завдяки можливості одночасно тестувати велику кількість студентів

**1.1.1 Види тестування**

Тести можуть бути класифіковані за різноманітними критеріями:

* За метою — інформаційні, діагностичні, навчальні, екзаменаційні
* За процедурою створення — стандартизовані, не стандартизовані
* За способом формування завдань — детерміновані, стохастичні, динамічні
* За технологією проведення — паперові, із ручним контролем, паперові із автоматизованим контролем за допомогою технології оптичного зчитування, апаратні аналогові (із застосуванням спеціально виготовленого обладнання), апаратні цифрові (із застосуванням програмного забезпечення)
* За формою завдань — закриті, відкриті, встановлення відповідності, впорядкування послідовності
* За інтерактивністю — традиційні та адаптивні.

Традиційний тест містить перелік питань та варіантів відповідей. Кожна відповідь оцінюється в певну кількість балів. Результат тесту залежить від кількості питань та сумарної кількості набраних студентом балів.

Як правило питання в тестах розташовуються по мірі зростання складності або, якщо тестуються знання цілого навчального курсу, групуються по темам, які входять до курсу.

Існують тести, в яких питання, які отримує студент залежать від відповідей на попередні запитання. В такому тесті кожен студент може отримувати завдання в різній послідовності чи навіть отримати різну кількість завдань.

Головна перевага комп’ютерного тестування перед традиційним тестуванням на папері це можливість динамічно формувати набір тестових завдань для кожного студента на випадковій основі чи на основі попередніх результатів тестувань. Це дозволяє покрити тестами більшу область знань, не збільшуючи при цьому розмір та тривалість самого тесту.

**1.1.2 Аналіз програм-аналогів**

**1.1.3 Постановка завдання**

Головним завданням розробника було створення кросс-платформеної системи, яка надавала би доступ до формування тестів, тестування та контролю результатів тестування як на десктопних системах, так і на мобільних платформах. Тому головним питанням, яке стояло перед розробником був вибір платформи клієнта, який зменшив би час та складність розробки за рахунок універсалізації.

На сьогодні найбільш універсальною платформою, яка здатна відображати користувацький інтерфейс та відтворювати користувацьку логіку практично на будь-якому пристрої є web-браузер. Тому основною платформою для клієнтської частини додатку був обраний саме він. Враховуючи складність клієнтського коду, необхідність складної взаємодії між різними частинами клієнтських даних, було вирішено будувати браузерний клієнт за ідеологією односторінкового додатку (SPA).

Одним з базових завдань, які ставилися при проектуванні додатку, була доступність повного клієнтського функціоналу та цілісного користувацького досвіду для студента на мобільних платформах. Сучасні мобільні браузери підтримують повний обсяг веб-технологій, аналогічний браузерам настільних систем, а сучасні технології адаптивного верстання дозволяють в значній мірі адаптувати веб-сторінки для перегляду на екранах мобільних пристроїв. Але якість користувацького досвіду складних односторінкових додатків на мобільних пристроях все одно може страждати. Тому било вирішено, окрім web-інтерфейсу, створити також мобільних додаток, який би надавав доступ до всіх функцій студента та забезпечував би користувацький досвід максимально близький до web-додатку.

Серверна частина має забезпечувати доступ до даних та їх обробку незалежно від того, якою платформою користується клієнт. Тому вона має містити базу даних, яка зберігатиме тестові матеріали, дані користувачів та результати тестів та web-службу, яка має надавати доступ до цих даних клієнтським додаткам та містити певну частину бізнес-логіки, яку доцільніше розмістити на сервері, а не у клієнтському додатку.

Таким чином система повинна мати такі компоненти: база даних, служба доступу до даних, сервер веб-клієнта, веб-клієнт, мобільний додаток (Android, опціонально iOS).

Було б дуже бажано об’єднати всі ці платформи єдиним стеком технологій. Це значно спростило б розробку та дозволило б максимальне повторне використання коду у додатках для різних платформ.

Тому для розробки була обрана платформа .NET Core та мова програмування C#. Стек технологій проекту включає в себе реляційну базу даних MS SQL Server 2019, ORM платформу Entity Framework Core, платформу для веб-додатків ASP MVC Core, набір інструментарію для створення мобільних додатків на C# Xamarin Forms та web-фреймворк Blazor wasm для створення браузерних додатків на платформі WebAssembly.

Такий вибір технологій дозволяє використати для розробки одну мову програмування (C#), максимально універсалізувати та полегшити повторне використання коду, запобігти дублюванню коду при створенні моделей даних, локалізації додатку, верифікації введених даних і т. ін.

Особливо хотілося б зупинитися на вибори Blazor wasm у якості платформи для браузерного клієнту. На момент планування додатку ця технологія ще знаходилася в режимі попереднього доступу, але до моменту захисту роботи має відбутися офіційний реліз. Вона використовує WebAssembly — нізкорівневе браузерне середовище, яке дозволяє завантажувати в браузер та виконувати в ньому бінарний код. В грудні 2019 року Wide Web Consortium рекомендував WebAssembly як четвертий компонент стеку веб-технологій на рівні з HTML, CSS та JavaScript і на даний момент всі найпопулярніші браузери, окрім Internet Explored та Opera Mini підтримують цю технологію. Ця технологія дозволяє використовувати в браузерах скомпільований код, що значно прискорює виконання користувацького коду у порівнянні із JavaScript та підсилює безпеку, тому що користувач більше не має можливості переглядати код сторінки та втручатися в нього. Найголовніше те, що вона значно розширює можливості браузерного програмування, дозволяючи використовувати для нього інші, крім JavaScript мови програмування, включаючи C#.

Таким чином архітектура додатку складається з бази даних MS SQL Server 2019, веб-служби на платформі ASP MVC Core, яка здійснює взаємодію між базою даних та клієнтськими додатками за допомогою ORM фреймворку Entity Framework Core та інтерфейсу REST WebApi, серверу статичних файлів веб-клієнту та мобільного клієнту. Всі три серверних компонента (база даних, служба доступу даних та сервер веб-клієнту) можуть розташовуватися як на одному фізичному сервері, так і на різних, здійснюючи взаємодію між собою через мережу.

Для побудови моделей використовується мова UML. UML є універсальним відкритим стандартом графічного відображення абстрактних моделей, які використовуються при плануванні та розробці складних систем.

Для розробки UML діаграм використаний редактор Microsoft Office Visio. В якості інтегрованого середовища розробки використовується Microsoft Visual Studio 2019. В якості системи контролю версій використовується git.

**1.2 Аналіз бізнес-процесів предметної області**

В предметній області додатку можна виділити такі бізнес-процесі

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Процес | Виконавець | Вхідні дані | | Вихідні дані | |
| Постачальник | Зміст | Споживач | Зміст |
| 1 | Додати предмет | Клієнтський додаток | Викладач | Новий предмет | База даних | Новий предмет |
| 2 | Додати курс | Клієнтський додаток | Викладач | Новий курс | База даних | Новий курс |
| 3 | Додати тест | Клієнтський додаток | Викладач | Новий тест | База даних | Новий тест |
| 4 | Змінити предмет | Клієнтський додаток | Викладач | Предмет, який підлягає зміні | База даних | Змінений предмет |
| 5 | Змінити курс | Клієнтський додаток | Викладач | Курс, який підлягає зміні | База даних | Змінений курс |
| 6 | Змінити тест | Клієнтський додаток | Викладач | Тест, який підлягає зміні | База даних | Змінений тест |
| 7 | Додати користувача | Клієнтський додаток | Адміністратор, студент | Дані нового користувача | База даних | Дані нового користувача |
| 8 | Отримати список тестів | Клієнтський додаток | База даних | Ознака, за якою мають бути відібрані тести (предмет, курс) | Викладач, студент | Список тестів |
| 9 | Тримати список питань | Клієнтський додаток | База даних | Тест, по якому треба отримати питання | Викладач/студент | Список питань |
| 9 | Додати інформацію про тестування | Клієнтський додаток | Студент | Інформація про проведене тестування | База даних | Інформація про проведене тестування |
| 10 | Переглянути результати тестування | Клієнтський додаток | База даних | Результати тестування | Викладач | Результати тестування. |

Означені бізнес-процеси використовуватимуться при проектуванні та реалізації серверної та клієнтської частин програмного продукту

**1.2.1. Побудова вихідної концептуально моделі даних предметної області**

Концепутальне проектування бази даних полягає у розробці концептуальної моделі бази даних, яка не враховує обрану модель даних та особливості цільової СКБД. Концепутальна модель, зазвичай представлена у вигляді сукупності типів сутностей та зв’язків між ними. Побудова концептуальнох моделі даних предметної області виконана в Microsoft Office Visio.

Можна виділити такі типи сутностей:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Назва | Опис | Категорія | Кількість екзкмплярів | Режим оновлення |
| 1 | Відповідь | Містить дані про відповідь на питання | Стрижньова | 10000 | 50% на рік |
| 2 | Питання | Містить питання до тестів | Асоціативна | 1000 | 50% на рік |
| 3. | Тест | Містить організовану колекцію питань | Асоціативна | 1000 | 50% на рік |
| 4 | Курс | Містить структуру навчальних курсів | Асоціативна | 1000 | 50% на рік |
| 5 | Предмет | Містить перелік предметів, до яких відносяться курси | Асоціативна | 100 | 10% на рік |
| 6 | Результат тестування | Містить результат проходження тесту студентом | Стрижньова | 10000 | 90% на рік |
| 7 | Викладач | Містить дані викладачів | Стрижньова | 100 | 10% на рік |
| 8 | Студент | Містить дані студента | Стрижньова | 10000 | 90% на рік |

І такі зв’язки між сутностями

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Тип сутності 1 | Тип сутності 2 | Опис | Потужність | Обов’язковість |
| 1 | Питання | Відповідь | Питання має відповідь, відповідь відноситься до питання | 1:Б | Обов’язкова (питання обов’язково має відповіді, відповідь обов’язково відноситься до питання) |
| 2 | Тест | Питання | Тест має питання, питання відноситься до тестів | 1:Б | Обов’язкова (тест обов’язково має питання, питання обов’язково відноситься до тесту) |
| 3 | Курс | Тест | Курс має тести, тест відноситься до курсу | Б:Б | Необов’язкова (курс може не мати тестів, тест може не відноситися до курсу) |
| 4 | Предмет | Курс | Предмет має курси, курс відноситься до предмету | 1:Б | Необов’язкова (предмет може не мати курсів, курс може не відноситися до предмету) |
| 5 | Предмет | Тест | Предмет має тести, тест відноситься до предмету | Б:Б | Необов’язкова(предмет може не мати тестів, тест може не відноситися до предмету |
| 6 | Студент | Тест | Студенту отримує завдання пройти тест. Тест заданий студенту | Б:Б | Необов’язкова(студенту можуть не бути задані тести, тест може бути заданий жодному студенту) |
| 7 | Студент | Курс | Студент зарахований на курс. Курс проходиться студентом | Б:Б | Необов’язкова (студент може бути не зарахований на жоден курс, курс може не мати жодного студента) |
| 8 | Результат тестування | Студент | Результат отриманий студентом, студент пройшов тест та отримав результат | Б:1 | Обов’язкова (результат обов’язково отриманий студентом) |
| 9 | Результат тестування | Тест | Результат отриманий при проходженні тесту. Тест пройдений певним результатом | Б:1 | Обов’язкова (результат обов’язково отриманий при проходженні тесту) |
| 10 | Викладач | Предмет | Викладач викладає певні предмети. Предмети викладаються викладачами | Б:Б | Необов’язкове (викладач може не мати жодного предмету, можуть бути предмети без викладача) |

Сутності мають відповідні атрибути

Атрибути сутності «Відповідь»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id відповіді | long |  | PK |
| Відповідь | Текст відповіді | String |  | Ні |
| Відсоток | Відсоток правильності відповіді | Int | 0-100 | Ні |
| Порядковий номер | Положення відповіді в списку відповідей | Int |  | Ні |
| Id питання | Id питання, до якого належить відповідь | long | FK | Ні |

Атрибути сутності «Питання»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id питання | long |  | PK |
| Питання | Текст питання | String |  | Ні |
| Порядковий номер | Положення питання в списку питань у тесті | Int |  | Ні |
| Id тесту | Id тесту, до якого належить питання | Long | FK |  |

Атрибути сутності «Тест»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id тесту | long |  | PK |
| Назва | Назва тесту | string |  | Ні |
| Id автора | Id викладача, який є власником тесту | long | FK | Ні |

Атрибути сутності «Курс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id курсу | long |  | PK |
| Назва | Назва курсу | string |  | Ні |
| Id предмету | Id предмету, до якого належить курс | long | FK | Так |
| Id автору | Id власника курсу | long |  | Ні |

Атрибути сутності «Предмет»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id предмету | long |  | PK |
| Назва | Назва предмету | string |  | Ні |

Атрибути сутності «Результат тестування»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id результату | Long |  | PK |
| Id тесту | Id тесту, який проходив студент | long | FK | Ні |
| Id студенту | Id студенту, який проходив тест | long | FK | Ні |
| Час проходження | Дата і час проходження студентом тесту | DateTime |  | Ні |

Атрибути сутності «Відповідь студента»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id відповіді | Long |  | PK |
| Id результату | Id результату тесту, до якого відноситься відповідь | long | FK | Ні |
| Id відповіді | Id відповіді, даної студентом | long | FK | Ні |

Атрибути сутності «Викладач»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id викладача | Long |  | PK |
| Ім’я | Ім’я викладача | string |  | Ні |
| Прізвище | Прізвище викладача | string |  | Ні |
| По-батькові | По-батькові викладача | string |  | Так |
| Телефон | Телефон викладача | string |  | Так |
| Електронна пошта | Електронна пошта викладача | string |  | Ні |
| Логін | Логін викладача | string |  | Ні |
| Пароль | Пароль викладача | string |  | Ні |

Атрибути сутності «Студент»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Опис | Тип даних | Область допустимих значень | Можливість значення NULL |
| Id | Id студента | Long |  | PK |
| Ім’я | Ім’я студента | string |  | Ні |
| Прізвище | Прізвище студента | string |  | Ні |
| По-батькові | По-батькові студента | string |  | Так |
| Телефон | Телефон студента | string |  | Так |
| Електронна пошта | Електронна пошта студента | string |  | Ні |
| Логін | Логін студента | string |  | Ні |
| Пароль | Пароль студента | string |  | Ні |



Концептуальна модель системи

**1.3 Аналіз вимог**

**1.3.1 Аналіз функціональних вимог**

В системі що розробляється можна визначити 2 користувацьких ролі:

* Викладач
* Студент

Класи та характеристики користувачів представлені в таблиці

Класи та характеристики користувачів

|  |  |
| --- | --- |
| Клас користувачів | Характеристики користувачів |
| Викладач | Викладач додає предмети, курси та тести до бази даних, передивляється результати тестування. Йому дозволено змінювати та видаляти питання і відповіді тестів |
| Студент | Студент може обирати для проходження тести, відкриті для всіх, або тести, які входять до курсу, членом яких є студент. Він може проходити тести стільки разів, скільки йому дозволено умовами курсу. Студент бачить лише власні результати тестування |

Відповідно до поставленого завдання була побудована модель предметної області. Діаграма варіантів користування представлена на малюнку



Діаграма користувацьких сценаріїв

Після створення діаграми користувацький сценаріїв та виявлення ролей можна перейти до визначення кожного сценарію — деталізації сценаріїв. В результаті ми отримаємо деталізовані сценарії, які включають назву сценарію та його специфікації.

|  |
| --- |
| Сценарій: Додати предмет |
| ID:1 |
| Короткий опис:  Додавання нового предмету |
| Головні актори:  Викладач |
| Передумови:  Сценарій починається коли викладач вибирає опцію «Додати предмет» |
| Основний потік:   1. Викладач вводить в систему назву нового предмету 2. Система перевіряє коректність введених даних (наявність назви предмету) 3. Система зберігає предмет в базі даних 4. В базі даних зберігається новий перелік предметів |
| Післяумови:  Система оновлює список предметів |
| Альтернативні потоки: немає |

|  |
| --- |
| Сценарій: Додати курс |
| ID:2 |
| Короткий опис:  Додавання нового курсу |
| Головні актори:  Викладач |
| Передумови:  Сценарій починається коли викладач вибирає опцію «Додати курс» |
| Основний потік:   1. Викладач вводить в систему назву нового курсу 2. Система перевіряє коректність введених даних (наявність назви курсу) 3. Система зберігає курс в базі даних 4. В базі даних зберігається новий перелік курсів |
| Післяумови:  Система оновлює список курсів |
| Альтернативні потоки: немає |

|  |
| --- |
| Сценарій: Додати тест |
| ID:3 |
| Короткий опис:  Додавання нового тесту |
| Головні актори:  Викладач |
| Передумови:  Сценарій починається коли викладач вибирає опцію «Додати тест» |
| Основний потік:   1. Викладач вводить в систему назву нового тесту 2. Викладач додає до тесту принаймні одне питання 3. Система перевіряє коректність введених даних (наявність питань та назви тесту) 4. Система зберігає тест в базі даних 5. В базі даних зберігається новий перелік тестів |
| Післяумови:  Система оновлює список тестів |
| Альтернативні потоки: немає |

|  |
| --- |
| Сценарій: Додати питання |
| ID:4 |
| Короткий опис:  Додавання нового питання |
| Головні актори:  Викладач |
| Передумови:  Сценарій починається коли викладач обрав тест та обрав опцію «Додати питання» |
| Основний потік:   1. Викладач вводить в систему назву текст нового питання 2. Викладач вводить перелік варіантів відповідей та помічає вірні відповіді 3. Система перевіряє коректність введених даних (наявність питання, наявність відповідей, наявність правильних відповідей, що дають в сумі 100% правильності) 4. При виявленні некоректних даних    1. Видати повідомлення про некоректні дані 5. Інакше    1. Система додає нове питання 6. Нове питання зберігається в базі даних |
| Післяумови:  Система оновлює список питань |
| Альтернативні потоки: немає |

|  |
| --- |
| Сценарій: Пройти тест |
| ID:5 |
| Короткий опис:  Студент виконує проходження тесту |
| Головні актори:  Студент |
| Передумови: Немає |
| Основний потік:   1. Студент обирає тест для проходження. 2. Система видає студенту перелік питань тесту 3. Студент обирає відповіді на питання 4. Студент обирає опцію «Завершити тест» 5. Система перевіряє, чи не перевищила кількість обраних відповідей допустиму 6. При перебільшенні кількості допустимих відповідей    1. Система виводить повідомлення про перебільшення кількості допустимих відповідей 7. Інакше    1. Система видає на екран результати тестування |
| Післяумови:  Система зберігає результат проходження тесту д бази даних |
| Альтернативні потоки: Виконання тесту перервано |

|  |
| --- |
| Сценарій: Проходження тесту преревано |
| ID:5.1 |
| Короткий опис:  Студент відміняє проходження тесту |
| Головні актори:  Студент |
| Передумови:  Студент активує проходження тесту |
| Основний потік:   1. Студент обирає опцію «Припинити виконання тесту» 2. Система запитує у студента підтвердження припинення виконання тесту 3. Якщо студент підтвердив вибір    1. Система відкидає результати проходженя тесту 4. Інакше    1. Система повертається до виконання тесту |
| Післяумови:  Немає |
| Альтернативні потоки: немає |

Для наочного представлення поведінки сценаріїв можна використати діаграму діяльності.



Діаграма діяльності сценарію «Додати тест»



Діаграма діяльності сценарію «Проходження тесту»

Для відображення взаємодії між окресленими класами на етапі побудови моделі аналізу розглянемо діаграму послідовностей для сценарію «Додати питання»



Діаграма послідовності сценарію «Додати питання»

Діаграма послідовності допомагає встановити часовий взаємозв’язок між класами, тобто показує реалізацію поведінки сценарію, але не є точним представленням кожного його кроку. Учасником діаграми є користувацький інтерфейс, який не розробляється до етапу проектування, тому на діаграмі послідовностей може бути не вказаний. В аналізі нас цікавить лише основна поведінка класів аналізу.

В даному випадку розглядається успішний розвиток сценарію, при якому викладач обрав тест, ввів вміст питання, можливі відповіді та вказав вірні. Система перевірила дані та за результатами перевірки додала нове питання до бази даних.

1. **Проектування**

**2.1.1 Базова архітектура системи**

Система, що проектується використовує дворівневу архітектуру (клієнт-сервер). Архітектура представлена на малюнку.



Архітектура системи тестування

На даному малюнку зображені компоненти системи тестування. Клієнтські додатки виконуються в веб-браузері на пристрої користувача, або як мобільний додаток на мобільному пристрої. Сервер зберігає базу даних, яка містить інформацію про тести, користувачів та результати тестування. Взаємодія з базою даних здійснюється за допомогою технології Entity Framework Core. Клієнтські додатки отримують дані з бази через мережеву службу, яка використовує технологію ASP MVC Core Web Api. Служба підключається до бази даних за допомогою класу, похідного від класу DbContext, який виконує операції з базою даних. Цей клас трансформує класи та методи мови C# на SQL запити до бази даних, а отримані результати запитів трансформує в класи моделі C#.

Для того, щоб уникнути жорсткої залежності від класу контексту бази даних, полегшити підтримку, тестування, масштабування коду та дозволити, при необхідності, використати інші джерела даних, крім бази даних MS SQL без необхідності значних змін у коді додатку веб-служба споживає клас контексту бази даних через абстрактний інтерфейс за допомогою ін’єкції залежностей. Веб-служба отримує HTTP-запити від клієнтів, десеріалізує їх в об’єкти C# та, через посередництво інтерфейсу, робить запити до бази даних. Отримані результати запиту веб-служба серіалізує у формат JSON та передає за допомогою протоколу HTTP клієнтам.

Клієнтські додатки формують HTTP запити до веб-служби, використовуючи REST Api. Таким чином вони запитують дані з бази даних та відправляють дані до веб-служби для обробки.

Роботу системи можна описати таким чином: викладач входить систему, формує навчальні курси, наповнює їх тестами. В свою чергу студент обирає тест для проходження і проходить його. Інформація про результати тестування у вигляді відповідей, даних студентом зберігається в базі даних, і може буди переглянута викладачем.

**2.1.2 Логічна модель бази даних**

Логічна модель будується на основі концептуальної моделі з урахуванням обраної моделі даних, але не враховуючи особливості цільової СКБД.

Усунення надлишковості модель відбувається, як правило, за рахунок декомпозиції відношень таким чином, аби в кожному відношенні зберігалися лише первинні факти. Процес перетворення бази даних до виду, який відповідає нормальним формам називається нормалізацією.

Нормалізація призначена для приведення структури бази даних до виду, який забезпечує мінімальну надлишковість. Кінцевою метою нормалізації є зменшення потенціальної протиречивості інформації, яка зберігається у БД.

Нормальна форма це властивість відношення в реляційній моделі даних, яка характерізує його з точки зору надлишковості, яка потенційно може призвести до логічно помилкових результатів вибірки або змінення даних. Нормальна форма визначається як сукупність вимог, якими можна задовольнити це відношення. Описано шість нормальних форм, але зазвичай для рішення практичних задач достатньо третьої нормальної форми.

**2.1.2.1 Перша нормальна форма**

Ознаками першої нормальної форми є:

* Кожна таблиця бази даних має первинний ключ: мінімальний набір полів, які унікально ідентифікують запис
* Відсутнє повторення груп, тобто дані одного типу не повторюються в одному запису

Архітектура реляційних баз даних є такою, що таблиці в ній завжди знаходяться в першій нормальній формі.

**2.1.2.2 Друга нормальна форма**

Таблиця знаходиться в другій нормальній формі тоді, коли вона знаходиться в першій нормальній формі і будь-який атрибут в ній залежить лише від первинного ключа, а не від його частини.

**2.1.2.3 Третя нормальна форма**

Таблиця бази даних знаходиться в третій нормальній формі коли вона знаходиться в другій нормальній формі і коли кожен неключовий атрибут безпосередньо залежить лише від первинного ключа таблиці, а не від будь-яких інших потенційних ключів.

Наша база даних вже знаходиться в третій нормальній формі, тому ми можемо побудувати її логічну модель. Треба звернути увагу, що в моделі сутностей «Викладач» та «Студент» ми мали однакові поля, тому є доречним виділити окрему таблицю «Персона», з якою ми зв’яжемо таблиці викладачів та студентів відношенням один до одного. Також, через те, що для аутентифікації та авторизації користувачів ми будемо використовувати вбудовану у фреймворк ASP NET Core систему ідентифікацій ASP NET Core Identity, та зберігатимемо аутентифікаційн дані користувачів в окремій базі даних, ми вилучимо з персональних даних поля логіну та паролю. Таким чином структура нашої бази буде мати наступний вигляд:



Розроблена логічна модель не містить транзитивних залежностей у відносинах. Таким чином модель відповідає третій нормальній формі.