НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.043

«До захисту допущено»

Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: Система для безпечного доступу до даних навчального закладу

Виконав: студент другого курсу, групи ІТ-84мп
(шифр групи)

Оксенчук Тарас Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Корнага Я.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)

«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Оксенчуку Тарасу Васильовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Система для безпечного доступу до даних навчального закладу»,

_____

науковий керівник дисертації__доцент, к.т.н., доцент Корнага Я. І.,_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – захищений доступ до даних.

4. Предмет дослідження – автентифікація користувача системи.

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; аналіз і реалізація методу; розробка методу; розробка програмного забезпечення; дослідження ефективності розробленого методу.

6. Орієнтовний перелік ілюстративного матеріалу – шість плакатів

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області	13.09.2019 р.	
2	Постановка задачі	15.09.2019 р.	
3	Аналіз інформаційного забезпечення	20.09.2019 р.	
5	Аналіз алгоритмічного забезпечення	25.09.2019 р.	
6	Розробка алгоритмічного забезпечення	15.10.2019 р.	
7	Розробка програмного забезпечення	01.11.2019 р.	
8	Маркетинговий аналіз стартап-проекту	10.11.2019 р.	
9	Висновки	15.11.2019 р.	

Студент _____ Т. В. Оксенчук
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Я. І. Корнага
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У роботі розглянуто проблему в області освіти з доступом до даних про власні успіхи в навчанні в електронному вигляді, показано основні особливості існуючих систем, їх переваги та недоліки.

Розроблено систему, що надає можливість швидкого та безпечного доступу до даних навчального закладу. Основною складовою системи є алгоритм для автентифікації користувача, визначення його ролі, генерації токену доступу, котрий містить електронний підпис. Дана система може бути використана в будь-яких освітніх закладах, як то школи, професійні училища чи вищі навчальні заклади. Завдяки використанню кешу дозволяє збільшити швидкість обробки запитів та зменшити навантаження на базу даних.

Ключові слова: автентифікація, доступ до даних, RESTful, Redis, .NET Core.

Розмір пояснювальної записки – 80 аркушів, містить 19 ілюстрацій, 34 таблиці, 6 додатків.

ABSTRACT

Examines the problem of the field of education with access to data on their successes in studying in electronic form, shows the main features of existing systems, their advantages, and disadvantages.

Developed a system that provides the ability to quickly and safely access educational data. The main component of the system is the user authentication algorithm, determining its role, generating an access token that contains an electronic signature. This system can be used in any educational institutions, such as schools, vocational schools or higher educational institutions. Due to the use of the cache, it allows us to increase the speed of query processing and reduce the load on the database.

Keywords: authentication, data access, RESTful, Redis, .NET Core.

Explanatory note size – 80 pages, contains 19 illustrations, 34 tables, 6 applications.

Пояснювальна записка
до магістерської дисертації

на тему:

Система для безпечного доступу до даних навчального закладу

Київ – 2019 року

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IT – інформаційні технології.

СУБД – система управління базами даних;

БД – база даних;

SQL (Structured Query Language) – мова структурованих запитів;

MVC (Model-View-Controller) – модель–представлення–контролер, шаблон проектування;

API (Application Programming Interface) – прикладний програмний інтерфейс;

ВСТУП

Завдяки великій доступності технологій та глобальній діджиталізації процесів наше повсякденне стає набагато простішим. Ми з легкістю отримуємо доступ до інформації, що нас цікавить. І дуже неприємно, коли ми цю інформацію не можемо легко отримати. Нажаль, прикладом такої ситуації може слугувати процес ознайомлення з власними результатами навчання в системі «Кампус». Якщо вам пощастило і ви не загубили логін та пароль, котрий отримали при вступі, то це ще не означає, що ви безперешкодно отримаєте, те чим цікавитесь. Потрібно бути певним, що хтось вніс актуальні дані. Окрім того, існує проблема з відмовоздатністю комплексу, часто в періоди пікових навантажень сервіс не доступний для використання. Можна довго перераховувати та шукати причини, але важко посперечатися з тим фактом, що система не здобула бажаної популярності, а ні серед студентів, а ні серед викладачів та працівників університету.

Метою моєї роботи є надання можливості швидкого та безпечного доступу до даних навчального закладу. Для цього потрібно спроектувати та реалізувати прикладний програмний інтерфейс, котрий зможуть використовувати всі бажаючі, для побудови додатків із візуальним інтерфейсом.

Для вирішення даної задачі потрібно розділити її на декілька підзавдань, котрі будуть інкапсульовані відносно один одного. Таким чином можна сконцентруватися на вирішенні кожної проблеми окремо, і в результаті отримати вирішення для великої комплексної проблеми. Я виділив такі завдання, для розв’язання:

- визначення та детальний опис кожної API-кінцевої точки, що будуть доступні для використання;
- вибір алгоритму автентифікації користувача в системі, для чіткого розмежування доступу до інформації;
- вибір сховища даних та проектування його структури;

– реалізація описаних вимог у вигляді веб-застосунку з прикладним програмним інтерфейсом;
– тестування реалізованої системи та виправлення знайдених помилок.

Провівши аналіз вимог, що були визначені вище, та спираючись на власний досвід, для реалізації описаної системи я вибрав платформу ASP.NET Core MVC, REST – архітектуру для побудови зрозумілого прикладного програмного інтерфейсу, для постійного збереження даних СУБД Oracle та Redis для збереження тимчасових даних.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Об’єкт та предмет дослідження

Для продуктивного навчання та вчасного реагування на можливі прогалини у знаннях, потрібно мати своєчасний доступ до інформації про поточний контроль. В сучасному світі технологій та тотального оцифрування всіх можливих даних, здавалося б, з доступом до такої інформації ніяких труднощів не повинно виникати. Але, як би це сумно не звучало, студенти нашого університету часто стикаються з цією проблемою. Можна довго сперечатися чому існуючі рішення не здобули великої популярності, але, на мою думку, краще витратити цей час на більш корисні речі. А саме: проектування та розробку альтернативної системи для безпечного доступу до даних навчального закладу. Під формулюванням «безпечний доступ», мається на увазі чітке розподілення користувачів системи на ролі, кожна з яких має певний набір базових можливостей. Також одним з головних чинників, котрі вливають на надійність системи, є захист від несанкціонованого доступу до даних. Окрім цього, система повинна бути простою у використанні, та мати легку можливість самостійного оновлення даних для автентифікації користувача.

Отже, об’єктом дослідження є захищений доступ до даних, за допомогою чіткого виділення ролей користувачів системи та закріплення за кожною з них певної області санкціонованого доступу. Також варто звернути увагу на те, що користувачі повинні мати доступ тільки до власних даних, і не мати можливості отримати конфіденційні дані інших користувачі, навіть якщо вони належать до однієї ролі.

З вище сказаного, випливає, що предметом дослідження є автентифікація користувача системи. Цей процес можна умовно розділити на два типи: ідентифікація користувача за номером телефону, якщо такі дані присутні в системі, та ідентифікація користувача для додавання номеру телефону в систему, якщо така інформація відсутня або застаріла (номер втрачено або змінено). Для другого типу потрібно здійснити перевірку інформації, котру знає лише

користувач, щоб уникнути несанкціонованої зміни номеру телефону та подальшого доступу до конфіденційних даних.

1.2 Аналіз існуючих рішень

Для більш детального поглиблення в проблему пропоную розглянути кілька вже існуючих рішень, котрі частково задовольняють поставлені вимоги. Але ці рішення через ті чи інші причини не отримали великої популярності чи взагалі не були інтегровані для використання.

1.2.1 Аналіз системи «Електронний Кампус»

Автоматизована інформаційна система «Електронний Кампус» – це система для інформатизації та діджиталізації навчального процесу в Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського». В системі можна знайти інформацію про контрольні заходи та їх розподіл за семестрами, методичне забезпечення, поточний контроль. Також можна ознайомитися з оголошеннями, кодексом честі, результатами атестацій та ректорського контролю, пройти опитування. Також в системі виділено кілька ролей користувачів для розподілення прав доступу, а саме: завідуючий кафедрою, методист кафедри, викладач-науковець та студент навчальної групи. З графічним інтерфейсом можна ознайомитись на рис. 1.1.

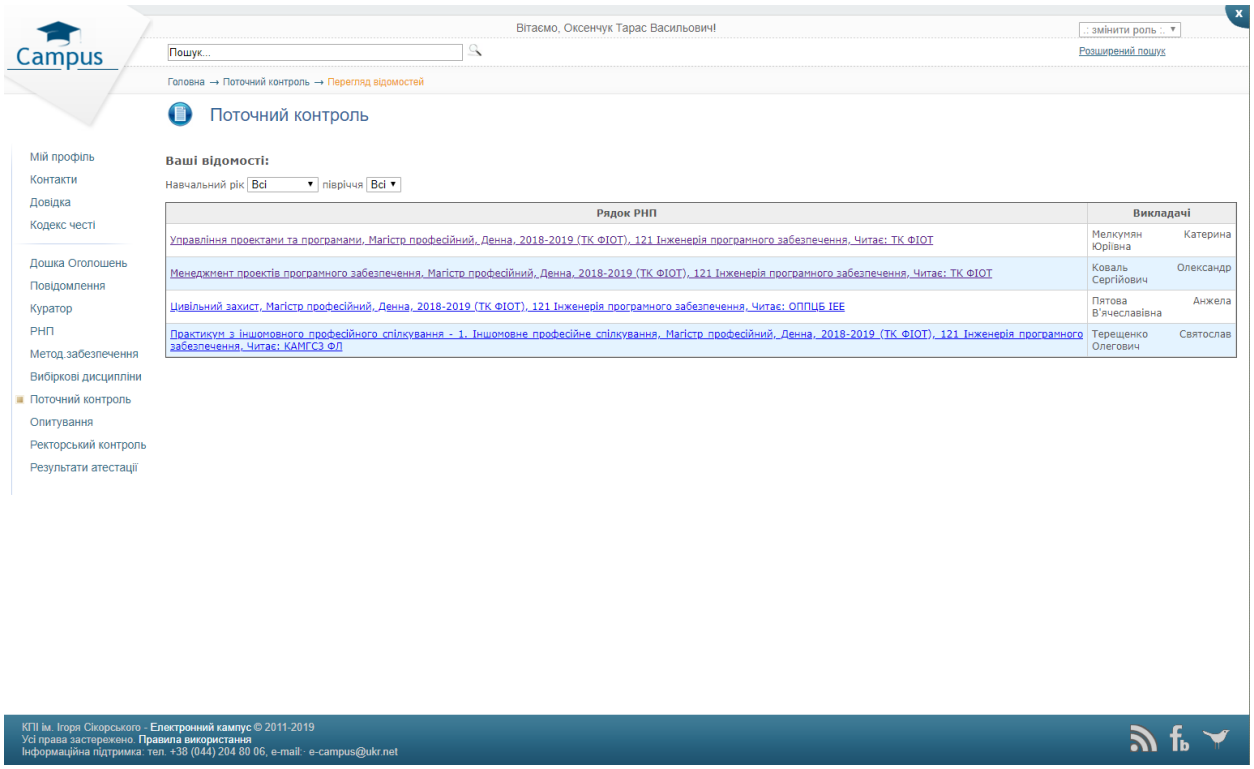


Рисунок 1.1. Зовнішній вигляд системи «Кампус»

Можна виділити такі основні переваги системи:

– розподілення користувачів за ролями;
– великий вибір інформації;

Серед недоліків можна виділити:

– труднощі при відновленню втраченого паролю;
– неспроможність системи працювати під навантаженням;
– відсутність інформації за деякими дисциплінами.

1.2.2 Аналіз системи «Google Classroom»

Google Classroom – це безкоштовна веб-система, розроблена для навчальних закладів, котра має за мету спростити завантаження, поширення та класифікацію завдань в електронному форматі. Основною метою системи є упорядкування процесу поширення файлів між викладачами та студентами. З візуальним інтерфейсом можна ознайомитись на наступному рис. 1.2.

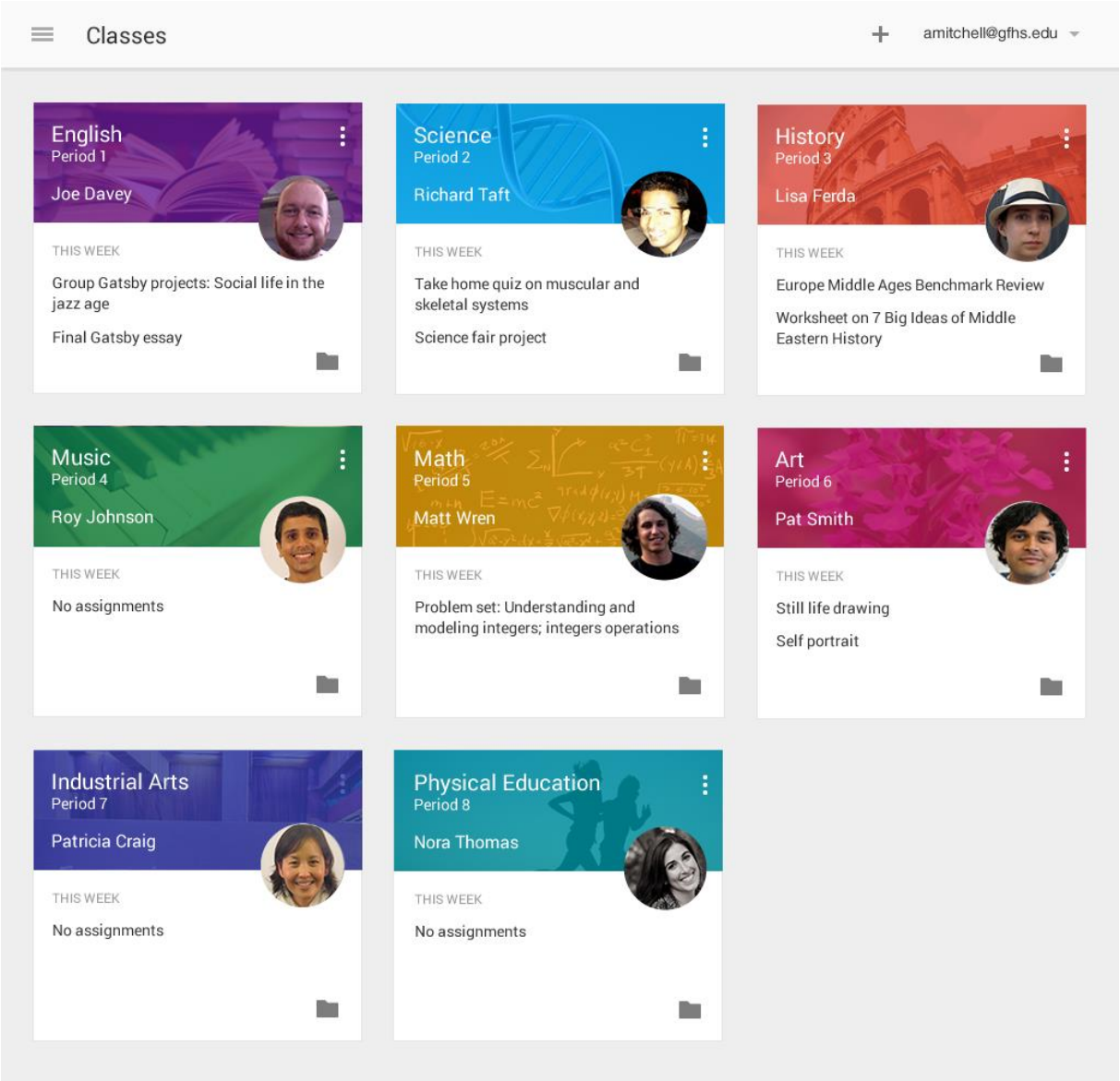


Рисунок 1.2. Зовнішній вигляд системи «Google Classroom»

Можна виділити такі основні переваги системи:

– висока стійкість до навантаження;
– зручний та інтуїтивно зрозумілий інтерфейс;
– мобільний додаток для постійного доступу до системи;
– легкість відновлення втраченого паролю;

Серед недоліків можна виділити:

– труднощі з інтеграцією у вже існуючі системи;
– збереження персональних даних на власних серверах.

1.3 Постановка задачі

Метою роботи є надання можливості швидкого та безпечного доступу до даних навчального закладу. Для досягнення поставленої мети потрібно вирішити наступні завдання:

- аналіз найбільш затребуваної інформації серед студентів та викладачів;
- проектування прикладного програмного інтерфейсу, за допомогою якого можна буде отримувати потрібні дані;
- розробка веб-застосунку, котрий буде реалізувати визначений інтерфейс та надавати доступ до сховища даних;
- реалізація алгоритму автентифікації користувача та розмежування доступу до даних.

Для коректної та зручної роботи системи було визначено такі основні ролі: анонімний користувач, студент, викладач.

Можливості, котрі будуть доступні для анонімного користувача:

- пошук інформації про студентів за ім’ям та прізвищем;
- пошук інформації про викладача за ім’ям та прізвищем;
- додавання власного номеру телефону в постійне сховище даних, для подальшої авторизації;
- оновлення номеру телефону, при втраті чи його зміні.

Можливості, котрі будуть доступні для користувача з роллю «Студент»:

- пошук інформації про студентів за ім’ям та прізвищем;
- пошук інформації про викладача за ім’ям та прізвищем;
- перегляд атестацій з предмету;
- перегляд результатів попередніх сесій;
- замовлення довідок про навчання;
- перегляд середнього балу за попередні сесії.

Можливості, котрі будуть доступні для користувача з роллю «Викладач»:

- пошук інформації про студентів за ім’ям та прізвищем;
- пошук інформації про викладача за ім’ям та прізвищем;

– перегляд власних дисциплін поточного семестру;
– перегляд оцінок групи за певною дисципліною;
– виставлення оцінок за дисципліну;
– виставлення оцінок за перескладання дисципліни;
– виставлення атестацій за власні дисципліни.

Створений веб-застосунок з прикладним програмним інтерфейсом повинен відповідати загальноприйнятим стандартам та визначеним вище вимогам, а саме:

– максимально проста та точна автентифікація користувача системи;
– забезпечення швидкого доступу до даних;
– висока стійкість до відмови;
– можливість самостійного відновлення працездатності після помилки;
– максимально ефективне використання ресурсів;
– відповідність REST-архітектурі;
– простота у використанні;
– простота в масштабуванні;
– простота а налаштуванні.

Висновки до розділу

В даному розділі описано об’єкт та предмет дослідження, оглянуто та проаналізовано деякі існуючі рішення, виділено основні інкапсульовані завдання, котрі потрібно вирішити для побудови визначеної системи. Для зручності та розподілення доступу було виділено ролі користувачів системи та визначено можливості, котрі повинна мати кожна роль. Також сформовано вимоги до системи, за якими буде вестися розробка, і вони ж будуть слугувати критеріями для перевірки на відповідність отриманого результату до поставлених цілей.

РОЗДІЛ 2. АНАЛІЗ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Засоби збереження даних

Для комфортного користування системою потрібно мати можливість збереження даних на довготривалий час на енергонезалежному носієві. Адже, якщо зберігати дані безпосередньо у внутрішніх об’єктах середовища, то, за замовчуванням, вони зберігаються в оперативній пам’яті. А, як всім відомо, цей тип пам’яті є енергозалежним.

Енергонезалежна пам’ять – це такий тип пам’яті, котрий може зберігати записані дані на довготривалий термін, навіть після відключення електроживлення від носія. Такий тип пам’яті, зазвичай, використовується для постійного довготривалого збереження даних. Вона, як правило, має меншу швидкість доступу до даних.

Енергозалежна пам’ять – це пам’ять, котра зберігає дані тільки на той час, доки до носія підключено електроживлення. Як перевагу, можна виділити велику швидкість доступу до даних, що зберігаються. Цей тип пам’яті використовують для зберігання проміжних даних програми, збереження кешу та іншої інформації, котра може бути потрібна під час виконання.

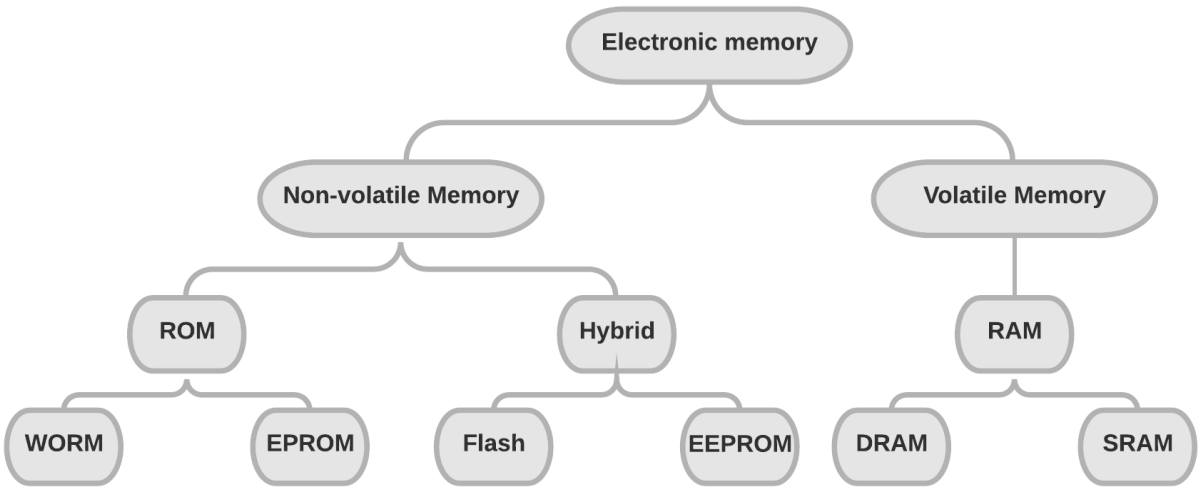


Рисунок 2.1. Типи пам’яті за залежністю від електроживлення

На рис. 2.1 можна побачити, на які типи розподіляється пам’ять. Зважаючи на попередню інформацію, можна зробити висновок, що для різних завдань використовують різні типи пам’яті. Для більш зручного користування системою, я вирішив додати кешування інформації, котра буде часто використовуватися. Кешування – це збереження інформації, котра може бути запрошена під час використання певної системи чи програми та зберігається в пам’яті, котра має велику швидкість зчитування даних.

2.1.1 Короткотривале збереження інформації

Для короткотривалого збереження інформації можна використовувати внутрішні об’єкти виконуваної програми або стороннє сховище. Оскільки, не малу роль відіграє можливість подальшого масштабування системи вирішено вибрати стороннє сховище даних. Тому що, при додаванні додаткових екземплярів виконуваного додатку, кожен із них буде мати власний набір внутрішніх об’єктів. Тобто, якщо створити і запустити 3 екземпляри застосунку, то кожен із них буде мати свій набір об’єктів. При запиті даних спочатку перевіряється кеш, якщо потрібні дані відсутні, то вони отримуються із постійного сховища, створюються об’єкти, котрі відповідають отриманим даним. При повторному запиті, ми не можемо гарантувати, що його буде опрацьовувати той же екземпляр додатку. Отже, існує велика ймовірність, що потрібно буде знову ж отримувати інформацію із постійного реляційного сховища. Натомість всі ці проблеми вирішуються за рахунок використання стороннього сервісу. Redis – це нереляційне розподілене сховище даних, котре легко масштабувати. Дані в ньому зберігаються безпосередньо в оперативній пам’яті у вигляді пар ключ-значення. Крім того, можна налаштувати персистентне збереження даних, для подальшого довготривалого їх зберігання. Але в даному проекті така можливість не буде використовуватися.

Для кращого розуміння можливостей цього сховища давайте розглянемо основні типи даних.

Redis Keys є бінарно-безпечними, це означає, що ви можете використовувати у якості ключа бідь-яку двійкову послідовність, від рядку «my_key» до вмісту файлу PNG. Порожній рядок також є допустимим ключем. Максимально допустимий розмір ключа становить 512 МБ. Але за рекомендацією, варто створювати ключі якомога меншого розміру, оскільки це може негативно вплинути на швидкодію сховища.

Redis Strings – це найпростіший тип значення, який можна пов'язати з ключем Redis. Це звичайний рядок. Але в ньому можна зберігати різноманітну інформацію, навіть серіалізовані об’єкти. Значення також не може перевищувати 512 МБ.

Redis Lists – реалізуються через зв’язний список. Це означає, що навіть якщо у вас є мільйони елементів всередині списку, операція додавання нового елемента в початок або в кінець списку виконується за постійний час. Який мінус? Доступ до елемента за допомогою індексу можливий у списках, реалізованих на основі масиву (постійний час для доступу по індексу). Списки Redis реалізовані на основі зв’язного списку, тому що для сховища даних важливо мати можливість додавати елементи до дуже довгого списку дуже швидко.

Redis Hashes виглядає так, як і загальноприйнята структура даних «хеш». Хеші рекомендують використовувати для зберігання об’єктів. Де у вигляді ключа буде виступати назва поля об’єкту, а значенням буде власне інформація, котра зберігається в цьому полі. Завдяки такому підходу досить зручно оновлювати інформацію частково, тобто змінювати значення тільки конкретних полів, що є великою перевагою при виникненні проблеми конкурентного оновлення даних. Однак, це лише рекомендація і можна знайти безліч застосувань для цієї структури даних.

Redis Sets – це неупорядкована колекція рядків.

Redis Sorted sets – це тип даних, схожий на поєднання між набором та хешом. Як і звичайні набори, відсортовані набори складаються з унікальних, неповторюваних рядків.

2.1.2 Довготривале збереження інформації

Стійкий – це об'єкт, який продовжує існувати навіть після того, як його батьківський об'єкт зупиняється або система, яка запускає його, вимикається. Коли створений об'єкт потребує стійкості, замість енергонезалежної пам’яті на зразок оперативної пам’яті використовується енергонезалежне сховище, включаючи жорсткий диск.

Наприклад, уявіть, що ви працювали в браузері Chrome в операційній системі Windows, і через деякі технічні проблеми процес перегляду браузера був зупинений; веб-переглядач перезапускається наступного разу, коли ви відкриєте його та намагається знову відкрити будь-які вкладки, які були відкриті під час аварії. Таким чином, існують стійкі об’єкти, навіть якщо програма була аварійно закінчена з якихось технічних причин. Стійкий стан – це стан, який залишається навіть після припинення процесу. Стійкі стани процесу зберігаються в постійному сховищі (енергонезалежне сховище, як жорсткі диски) до вимкнення або відмови системи, і їх легко отримувати, як тільки система вмикається. Основні об’єкти створюваної системи повинні бути стійкими для підтримання даних у подібному стані при включенні системи.

Реляційна база даних – це набір формально описаних таблиць, з яких можна отримати доступ або повторно зібрати дані різними способами без необхідності реорганізації таблиць бази даних. Стандартний прикладний програмний інтерфейс для користування реляційною базою даних – це структурована мова запитів (SQL). Оператори SQL використовуються як для інтерактивних запитів щодо інформації з реляційної бази даних, так і для збору даних для звітів.

Oracle – це один з найбільших постачальників на ринку IT на підприємстві та скорочена назва його флагманського продукту – системи управління реляційними базами даних (RDBMS), яка офіційно називається Oracle Database. Програмне забезпечення бази даних знаходиться в центрі багатьох

корпоративних IT-середовищ, підтримуючи поєднання програм обробки транзакцій, бізнес-аналітики та аналітики.

У наступні десятиліття після впровадження технології RDBMS Oracle значно розширив свій асортимент продукції за рахунок внутрішньої розробки та численних придбань. Зараз він також продає кілька інших баз даних, безліч ліній ділових додатків, програмне забезпечення для аналізу даних, проміжне програмне забезпечення, комп'ютерні системи, обладнання для зберігання даних, засоби розробки та інші технології. Крім того, Oracle працює над тим, щоб зарекомендувати себе як провідний постачальник хмарних обчислень, хоча спочатку він прохолодно сприйняв хмару.

Але Oracle Database – це все-таки технологія, яка найчастіше асоціюється з великими компаніями; це також основна платформа управління даними для додатків Oracle та сховища даних, BI та систем аналітики, які Oracle пропонує своїм клієнтам.

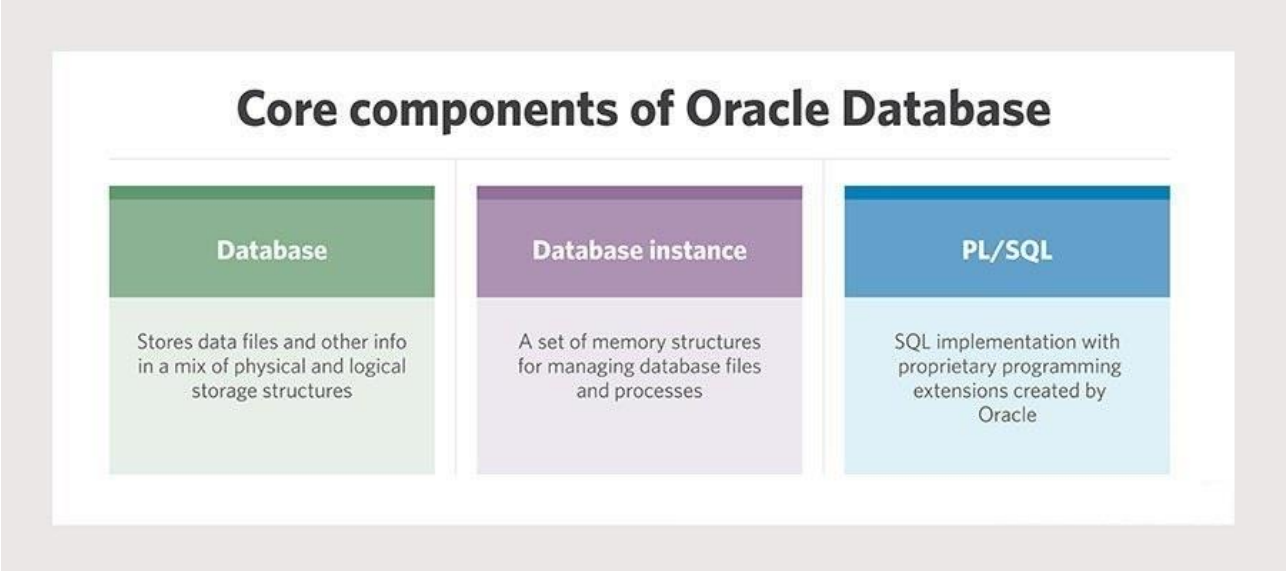


Рисунок 2.3. Основні компоненти бази даних Oracle

Як і інше програмне забезпечення RDBMS, Oracle Database побудований на основі SQL, стандартизованої мови програмування, яку адміністратори бази даних, аналітики та інші IT-фахівці використовують для управління базами даних та запиту даних, що зберігаються в них. На рис. 2.3 зображено основні компоненти бази даних, а саме: сховище даних, екземпляр бази даних та власне

розширення структурованої мови запитів. Програмне забезпечення Oracle прив’язане до PL/SQL – мови, розробленої Oracle, яка додає набір розширень фірмового програмування до стандартних SQL - поширена практика серед постачальників RDBMS. Oracle Database також підтримує програмування на Java, а програми, написані на PL/SQL або Java, можна викликати з іншої мови. Саме завдяки цій можливості ми можемо з легкістю використовувати цю базу даних в тандемі мовою C#, використовуючи відповідний провайдер даних.

2.2 Платформа та мова програмування

Дуже важливою є платформа, котра використовується для виконання написаного коду та безпосереднього запуску додатку. Платформа – це реальна або концептуальна структура, що слугує опорою або керівництвом для побудови чогось, що розширює структуру на щось корисне.

У комп’ютерних системах платформа часто є шаруватою структурою, яка вказує на те, які програми можуть бути або повинні бути побудовані та як вони взаємопов'язані. Деякі платформи комп'ютерної системи також включають фактичні програми, задають інтерфейси програмування або пропонують інструменти програмування для використання. Основою може бути набір функцій в системі та спосіб взаємозв'язку; шари операційної системи; шари підсистеми програми; як комунікація повинна бути стандартизована на певному рівні мережі; і так далі. Платформа, як правило, більш вичерпна, ніж протокол, і більш рецептивна, ніж структура.

2.2.1 Платформа для побудови системи

Платформа ASP.NET Core представляє технологію компанії Microsoft, спроектовану для створення різного роду веб-сайтів: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів. З однієї сторони, ASP.NET Core є продовженням платформи ASP.NET. Але з іншої сторони, це не просто черговий реліз. Вихід ASP.NET Core фактично означає революцію всієї платформи, її якісне оновлення. Розробка платформи почалася ще в 2014 році. Тоді платформа

мала тимчасову назву ASP.NET vNext. У липні 2016 року вийшов перший реліз платформи. А у вересні 2019 року вийшла версія ASP.NET Core 3.0, про яку я розповім детальніше.

ASP.NET Core є платформою з повністю відкритим сирцевим кодом. Усі вихідні файли доступні на GitHub. ASP.NET Core може працювати поверх крос-платформного середовища .NET Core, котрий може бути запущений на основних популярних операційних системах: Windows, Mac OS, Linux. І таким чином, за допомогою ASP.NET Core, ми можемо створювати крос-платформні додатки. І хоч Windows досі переважає в якості платформи для розробки та розширення додатків, але тепер вже ми не обмежені лише однією операційною системою. Для публікації веб-додатків можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Завдяки модульності платформи всі необхідні компоненти веб-додатків можна завантажувати як окремі компоненти через пакетний менеджер NuGet. Крім того, на відміну від попередніх версій платформ немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API та Web Pages. У попередніх версіях платформи ці технології реалізувалися окремо, і тому містили багато дублюючого функціоналу. Наразі вони є об'єднаними в одну програмну модель ASP.NET Core MVC. А Web Forms повністю залишилися в минулому.

Окрім об’єднання вищезгаданих технологій в одну модель MVC, додано ряд нових функцій. Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис HTML з кодом C#. ASP.NET Core має багато можливостей для розширення. Фреймворк побудований з набору відносно незалежних компонентів. І ми можемо використовувати вбудовану реалізацію цих компонентів, або розширювати їх за допомогою механізму наслідування, або створювати свої компоненти та використовувати їх разом зі своїм функціоналом.

Також спрощено керування залежностями та конфігурування проекту. Фреймворк має власний легкий контейнер для ін’єкції залежностей, і більше немає необхідності застосовувати сторонні контейнери, такі як Autofac, Ninject. Хоча, при бажанні їх також можна продовжити використовувати.

Для обробки запитів зараз використовується новий конвеєр HTTP, котрий базується на компонентах Katana та специфікації OWIN. А його модульність дозволяє легко додати свої власні компоненти. Якщо підсумувати, то можна вивести наступні ключові відмінності ASP.NET Core від попередніх версій ASP.NET:

– новий легкий і модульний конвеєр HTTP-запитів;
– можливість публікації додатку як на IIS, так і в рамках свого власного процесу;
– використання платформи .NET Core та її функціональності;
– розповсюдження компонентів платформ через NuGet;
– інтегрована підтримка для створення та використання пакетів NuGet;
– єдиний стек веб-розробки, що поєднує Web UI и Web API;
– конфігурація для спрощеного використання у хмарі;
– вбудована підтримка ін’єкції залежностей;
– крос-платформність;
– розвиток в open source, відкритість до змін.

Ці та інші особливості та можливості стали основою для нової моделі програмування. І вони стали одними із ключових аргументів при виборі платформи для розробки системи безпечного доступу до даних навчального закладу. На рис. 2.4 можна побачити загальну структуру вибраної платформи.

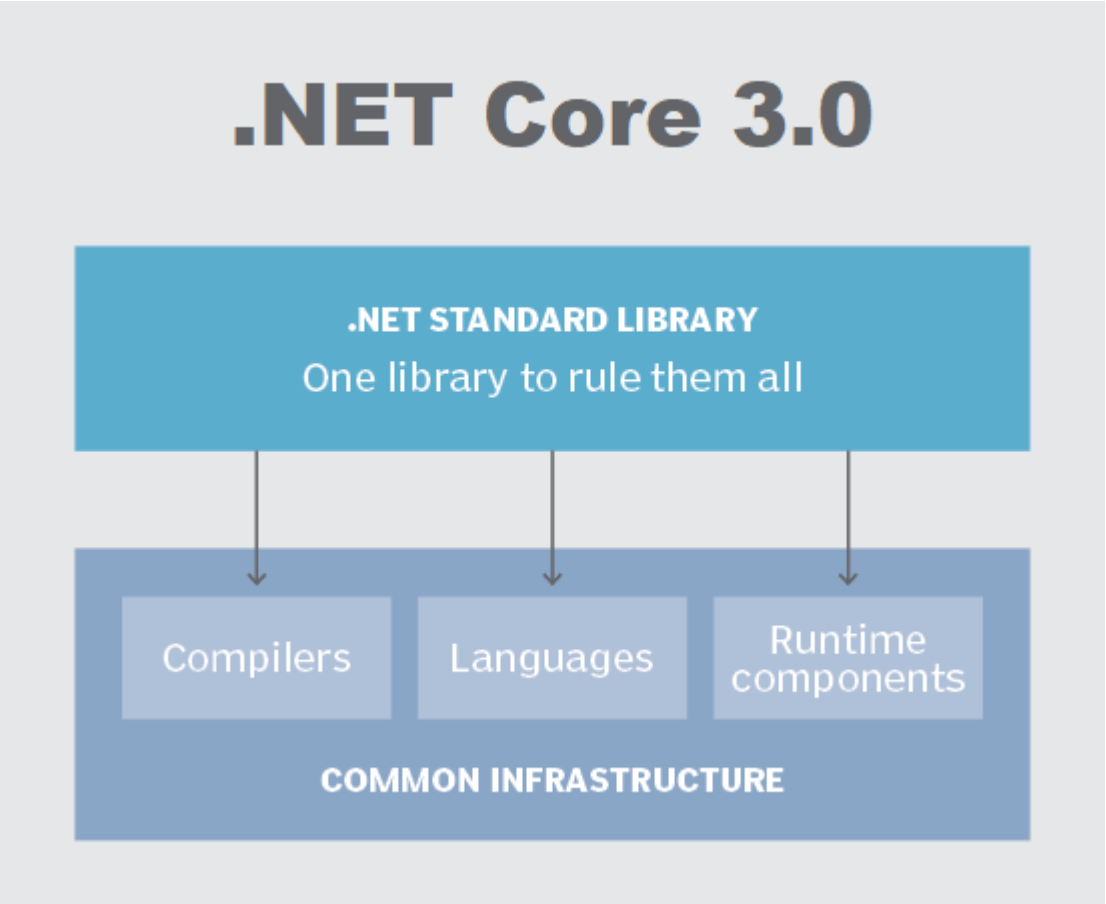


Рисунок 2.4. Структура платформи .NET Core 3.0

2.2.2 Мова програмування

C# – строго типізована об'єктно-орієнтована мова програмування. Вона має відкритий код, є простою, сучасною, гнучкою та універсальною. Далі я поясню, що собою являє, що може робити та чим відрізняється від C++ та інших мов програмування.

C# – мова програмування, розроблена та запущена Microsoft у 2001 році. C# – це проста, сучасна та об'єктно-орієнтована мова, яка забезпечує сучасних розробників гнучкістю та можливостями для створення програмного забезпечення, яке не тільки працюватиме сьогодні, але застосовуватиметься у майбутньому. Ось основні характеристики мови:

– сучасна і легка;
– швидкий та відкритий код;
– кросплатформність;

– безпечність;
– універсальність;
– еволюційність.

Основною метою при створені було розробити мову програмування, яку не тільки легко вивчити, але й яка підтримує сучасний функціонал для всіх видів розробки програмного забезпечення. Якщо подивитися на історію мов програмування та їх особливості, кожна мова програмування була розроблена з певною метою, щоб вирішити конкретну потребу в той час. Однак дана мова створена для того, щоб мати на увазі потреби бізнесу та підприємств. Вона розроблена для бізнесу для створення всіх видів програмного забезпечення за допомогою однієї єдиної мови програмування.

C# забезпечує функціональність для підтримки сучасної розробки програмного забезпечення. Вона підтримує потреби в веб-розробці, розробці для мобільних пристроїв та настільних додатків. Деякі з сучасних функцій цієї мови програмування: узагальнення (generics), автоматична ініціалізація типів і колекцій, лямбда-вирази, динамічне програмування, асинхронне програмування, кортежі, відповідність шаблонів (pattern matching), розширена відладка та обробка винятків та багато іншого.

На синтаксис мови C# вплинули C ++, Java, Pascal та кілька інших мов, які легко засвоїти. C# також уникає складних та неструктурованих конструкцій. На рис. 2.5 можна ознайомитись з основними типами, котрі використовуються в вибраній мові програмування.

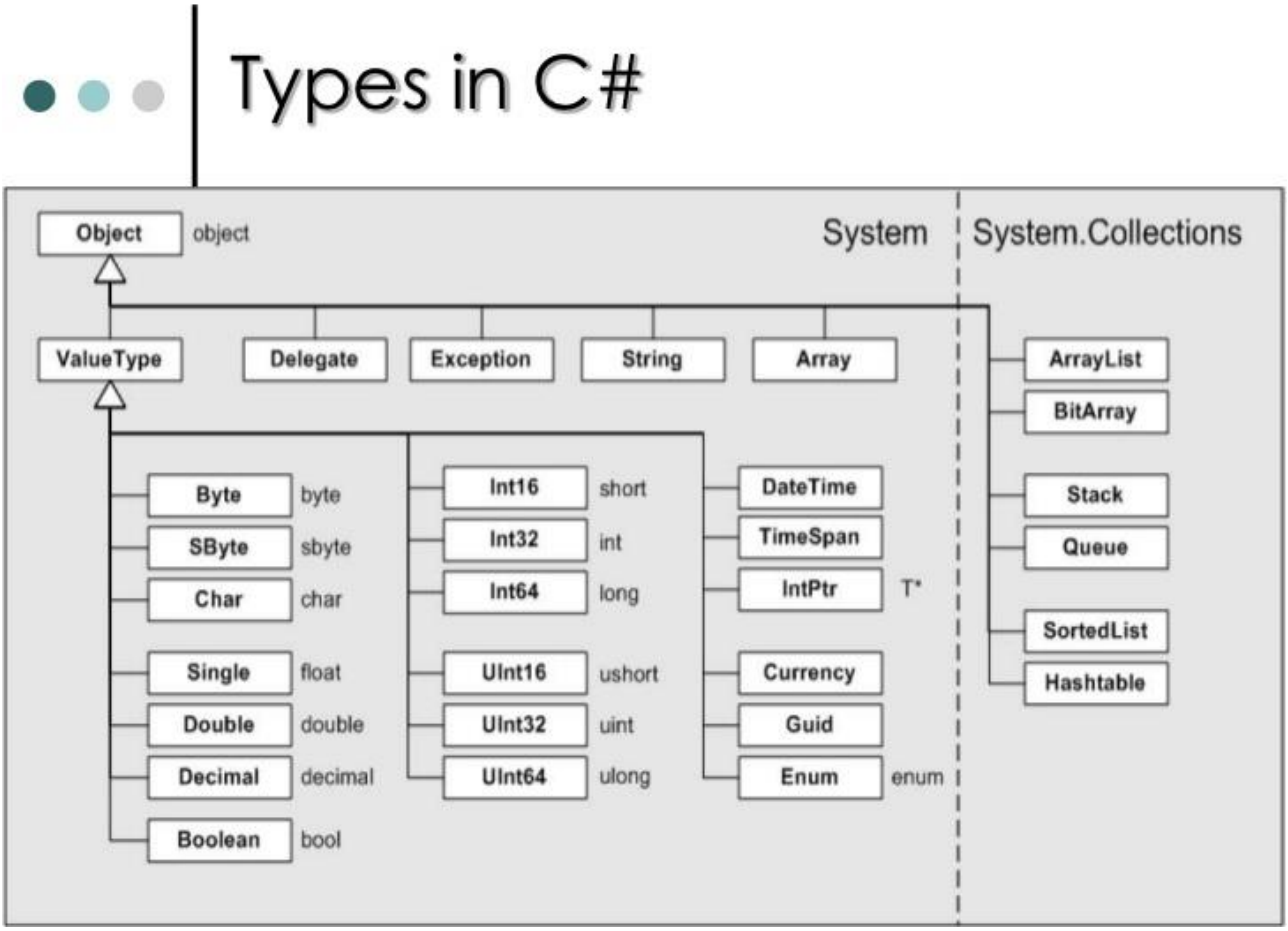


Рисунок 2.5. Типи даних в мові програмування C#

Дана мова має відкритий код у .NET Foundation, який керується та працює незалежно від Microsoft. Специфікації, компілятори та пов'язані мови – це проекти з відкритим кодом у Github. Хоча дизайном функцій мови керує корпорація Майкрософт, спільнота з відкритим кодом дуже активна у розробці та удосконаленні мови. Вона швидша в порівнянні з іншими мовами програмування високого рівня. Версія 8 має багато поліпшень продуктивності. C# не дозволяє такі перетворення типів, які можуть призвести до втрати даних або інших проблем. Що дозволяє розробникам писати безпечний код. Ось перелік деяких ключових понять у C#, які допомагають писати безпечний та ефективний код:

– заборонено робити небезпечні (unsafe) перетворення об’єктів;
– підтримка nullable та non-nullable типів;

– позначивши структуру як readonly, тип стає immutable і при використанні в якості параметру створюються копії об’єкту, замість зміни початкового;
– ніколи не передайте структуру як параметр, якщо вона не оголошена модифікатором readonly, оскільки це може негативно вплинути на продуктивність і може призвести до незрозумілої поведінки;
– для роботи з пам'яттю як послідовностю байтів можна використовувати Span<T> або ReadOnlySpan<T>.

2.2.3 Об'єктно-реляційне відображення

Для полегшення роботи з базою даних нам потрібно мати якийсь інструмент, що буде перетворювати реляційні моделі в об’єкти, котрі можна використовувати безпосередньо в коді програми. Для вирішення цієї задачі я вибрав інструмент для об’єктно-реляційного відображення. Dapper – це мікро-ORM або це простий інструмент для відображення об'єктів, котрий допомагає зіставити вихідний запит на клас C#. Це високоефективна система доступу до даних, побудована командою StackOverflow та випущена з відкритим код. Оскільки в проекті я планую надавати перевагу написанню збережених процедур та написанню нативних запиту замість використання повноцінних інструментів ORM, таких як EntityFramework або NHibernate, то Dapper є дуже хорошим вибором. За допомогою Dapper дуже легко запустити SQL-запит на базу даних і отримати результат, відображений на C# клас.

Dapper Framework фактично розширює інтерфейс IDbConnection, наявний у просторі імен System.Data. У ньому є багато методів розширення для доступу до даних та відображення результату до типу C#, визначеного під класом SqlMapper. Отже, щоб використовувати цей інструмент, спочатку нам потрібно оголосити об’єкт IDbConnection та ініціалізувати його до SqlConnection для підключення бази даних. Далі ми описуємо об’єкти, в які ми будемо відображати результати запитів. А потім просто викликаємо потрібні запити або процедури та оброблюємо отриману інформацію.

2.3 Прикладний програмний інтерфейс

REST – це абревіатура для REpresentational State Transfer. Це архітектурний стиль для розподілених гіпермедіа-систем, котрий вперше був представлений Роєм Філдінгом у 2000 році у своїй знаменитій дисертації. Як і будь-який інший архітектурний стиль, REST має свої 6 ключових обмежень, які повинні бути реалізовані, якщо інтерфейс потрібно називати RESTful. Ці принципи перераховані нижче:

- **клієнт-сервер** – відокремлюючи проблеми користувальницького інтерфейсу від проблем зберігання даних, ми покращуємо портативність користувальницького інтерфейсу на декількох платформах та покращуємо масштабованість, спрощуючи серверні компоненти;
- **без збереження стану** – кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння запиту, і не може скористатися будь-яким збереженим контекстом на сервері. Тому стан сесії повністю зберігається на клієнті;
- **кешування** – обмеження кешу вимагають, щоб дані у відповіді на запит неявно або явно позначали як кешовані або некешовані. Якщо відповідь є кешованою, то клієнтський кеш має право повторно використовувати цю відповідь для наступних, рівнозначних запитів;
- **уніфікований інтерфейс** – за допомогою застосування принципу загальної інженерії програмного забезпечення до компонентного інтерфейсу спрощується загальна архітектура системи та покращується видимість взаємодій. Щоб отримати єдиний інтерфейс, для керування поведінкою компонентів потрібно кілька архітектурних обмежень. REST визначається чотирма обмеженнями інтерфейсу: ідентифікація ресурсів; маніпулювання ресурсами через представництва; повідомлення з самоописанням; і, гіпермедіа як двигун стану застосування;

– **багатошарова система** – шаруватий стиль системи дозволяє архітектурі складатися з ієрархічних шарів, обмежуючи поведінку компонентів, таким чином, щоб кожен компонент не міг «бачити» поза безпосереднім шаром, з яким вони взаємодіють;
– **код на вимогу** (необов'язково) – REST дозволяє розширити функціональність клієнта, завантаживши та виконавши код у вигляді аплетів чи сценаріїв. Це спрощує клієнтів, зменшуючи кількість функцій, необхідних для попередньої реалізації.

Ключовою абстракцією інформації в REST є ресурс. Будь-яка інформація, яку можна назвати, може бути ресурсом: документом або зображенням, тимчасовою службою, колекцією інших ресурсів, невіртуальним об’єктом (наприклад, людиною) тощо. REST використовує ідентифікатор ресурсу для ідентифікації конкретного ресурсу, що бере участь у взаємодії між компонентами.

Стан ресурсу в будь-яку конкретну часову позначку називається представленням ресурсу. Представлення складається з даних, метаданих, що описують посилання на дані та гіпермедіа, які можуть допомогти клієнтам у переході до наступного бажаного стану.

Формат даних представлення відомий як тип медіа. Тип медіа ідентифікує специфікацію, яка визначає, як представлення має оброблятися. Справді RESTful API схожий на гіпертекст. Кожна адресована одиниця інформації містить адресу або явно (наприклад, атрибути посилання та ідентифікатора), або неявно (наприклад, походить від визначення типу носія та структури представлення носія).

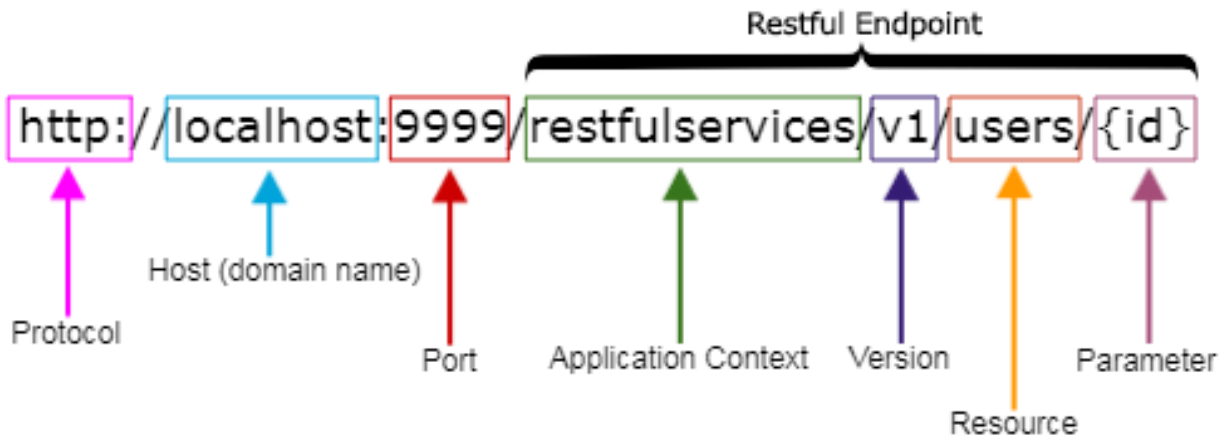


Рисунок 2.6. Загальний вигляд кінцевої точки прикладного програмного інтерфейсу

Крім того, представлення ресурсів має бути самоописовим: клієнту не потрібно знати, чи є ресурс співробітником чи пристроєм. Він повинен діяти на основі медіа-типу, пов'язаного з ресурсом. Так що на практиці ви в кінцевому підсумку створите безліч користувацьких типів медіа - як правило, один тип медіа, пов’язаний з одним ресурсом.

Інша важлива річ, пов'язана з REST, – це ресурсні методи, які використовуються для виконання бажаного переходу. Велика кількість людей неправильно ставиться до ресурсних методів до методів GET/PUT/POST/DELETE.

Рой Філдінг ніколи не згадував жодної рекомендації щодо того, який метод слід використовувати в якому стані. Все, що він наголошує, – це те, що він повинен бути єдиним інтерфейсом. Якщо ви вирішите, що HTTP POST буде використовуватися для оновлення ресурсу – а не як більшість людей рекомендує HTTP PUT – це добре, і інтерфейс програми буде RESTful.

В ідеалі все, що потрібно для зміни стану ресурсу, має бути частиною відповіді API для цього ресурсу – включаючи методи та в якому стані вони залишать представлення.

Інша річ, яка допоможе вам під час створення API RESTful – це те, що результати API на основі запитів повинні бути представлені переліком посилань

із підсумковою інформацією, а не масивами оригінальних представлень ресурсів, оскільки запит не замінює ідентифікацію ресурсів.

Дуже багато людей вважають доцільним порівнювати HTTP з REST. REST і HTTP не одне й те ж саме. Хоча, оскільки REST також має намір зробити Інтернет більш упорядкованим та стандартним, він виступає за більш чітке використання принципів. І саме звідси люди намагаються почати порівнювати REST з веб (HTTP). У своїй дисертації Рой Філдінг ніде не згадував жодної директиви щодо впровадження, включаючи будь-які параметри протоколу та HTTP.

Найпростіше кажучи, в архітектурному стилі REST дані та функціональність вважаються ресурсами і доступ до них здійснюється за допомогою Уніфікованих ідентифікаторів ресурсів (URI). Ресурси діють за допомогою набору простих, чітко визначених операцій. Клієнти та сервери обмінюються уявленнями про ресурси за допомогою стандартизованого інтерфейсу та протоколу, як правило, HTTP.

Ресурси відокремлюються від їх представлення, так що їх вміст може бути доступним у різних форматах, таких як HTML, XML, звичайний текст, PDF, JPEG, JSON та інші. Метадані про ресурс доступні та використовуються, наприклад, для кешування, виявлення помилок передачі, узгодження відповідного формату представлення та проведення автентифікації або контролю доступу. І найголовніше, що кожна взаємодія з ресурсом не має статусу. Усі ці принципи допомагають програмам RESTful бути простими, легкими та швидкими.

Для зручного тестування власних кінцевих точок прикладного програмного інтерфейсу потрібно використовувати якийсь базовий клієнт. Його можна згенерувати за допомогою Swagger. Swagger - це набір інструментів з відкритим сирцевим кодом, створених навколо специфікації OpenAPI, які можуть допомогти вам розробити, створити, документувати та споживати API REST. На рис. 2.7 можна ознайомитися з загальним прикладом згенерованого клієнту. До основних інструментів Swagger належать:

– **Swagger Editor** – редактор на базі браузера, куди можна писати характеристики OpenAPI;
– **Swagger UI** – надає специфікації OpenAPI як інтерактивну документацію API;
– **Swagger Codegen** – генерує заглушки сервера та бібліотеки клієнтів із специфікації OpenAPI.

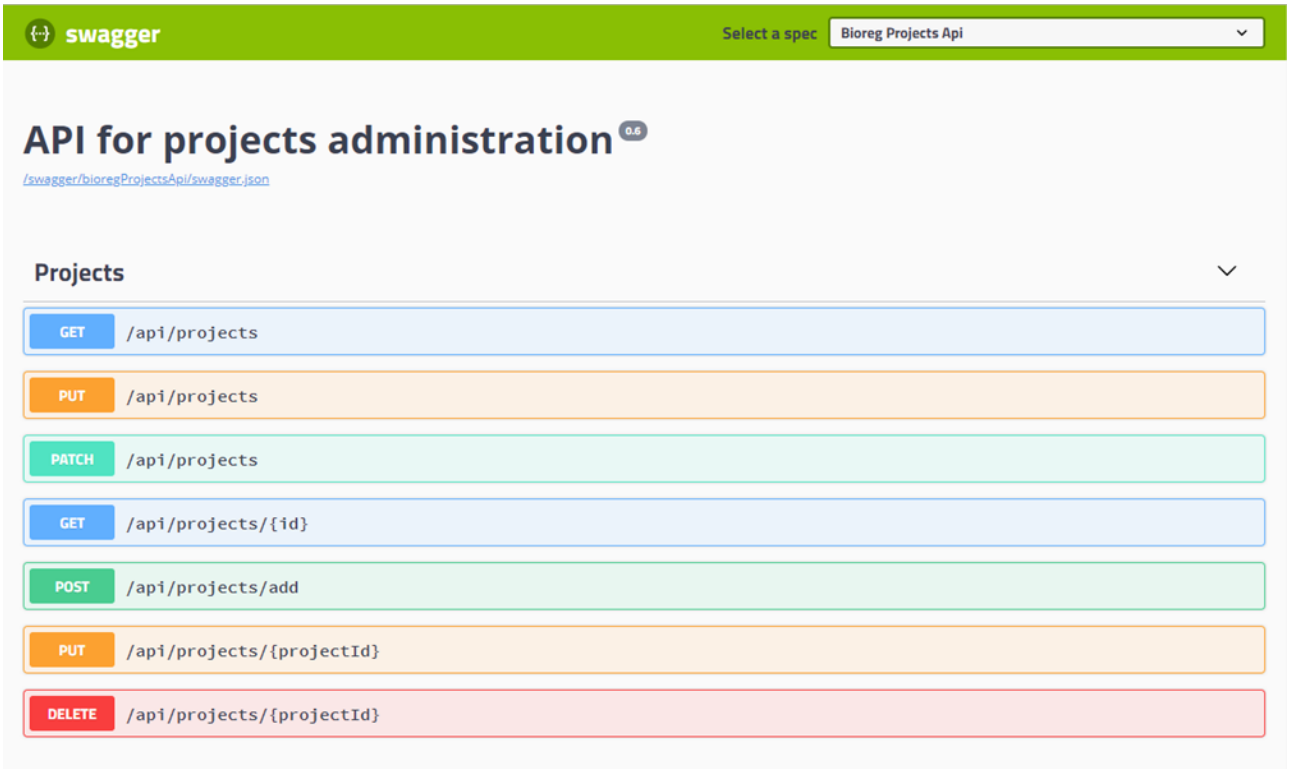


Рисунок 2.7. Приклад клієнту, створеного за допомогою Swagger

Висновки до розділу

В цьому розділі я розглянув, проаналізував та вибрав технології та інструменти створення системи безпечного доступу до даних навчального закладу, навів їх опис та характеристики, вказав на очевидні переваги та неочевидні недоліки.

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Архітектура програмного забезпечення системи зображує організацію або структуру системи та дає пояснення, як вона поводиться. Система представляє сукупність компонентів, які виконують певну функцію або набір функцій. Іншими словами, архітектура програмного забезпечення забезпечує міцну основу, на якій можна будувати програмне забезпечення.

Серія архітектурних рішень та компромісів впливають на якість, продуктивність, ремонтопридатність та загальний успіх системи. Неврахування загальних проблем та довгострокових наслідків може поставити під загрозу вашу систему.

Існує кілька моделей та принципів архітектури високого рівня, які зазвичай використовуються в сучасних системах. Їх часто називають архітектурними стилями. Архітектура програмної системи рідко обмежується одним архітектурним стилем. Натомість поєднання стилів часто складають повну систему.

Шаруватий шаблон – це n-ярусна архітектура, де компоненти організовані в горизонтальні шари, це можна побачити на рис. 3.1. Це традиційний метод розробки більшості програмного забезпечення, де всі компоненти взаємопов’язані, але не залежать один від одного.

У цій архітектурі є чотири шари, де кожен шар має зв'язок між модулями та компонентами всередині них. Далі перераховані ключові компоненти:

- презентаційний шар, він містить усі категорії, пов'язані з візуальним представленням даних;
- діловий шар, він містить логіку поведінки та процесів;
- шар стійкості використовується для обробки функцій, таких як об'єктно-реляційне відображення;
- шар бази даних, тут зберігаються всі дані.

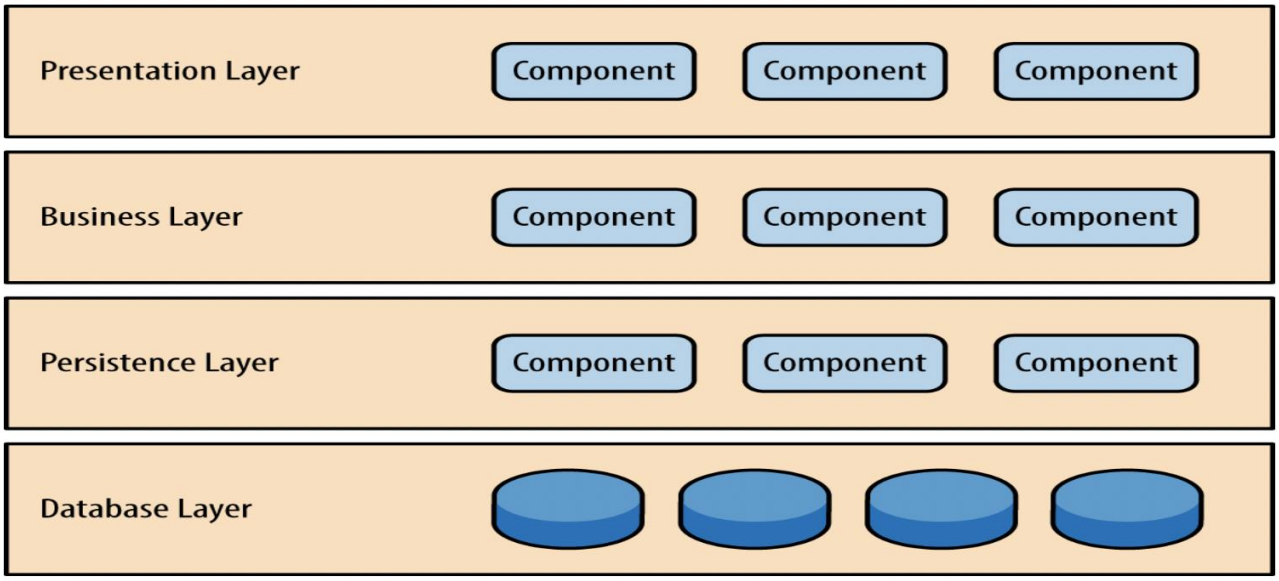


Рисунок 3.1. Приклад багатошарової архітектури

У нашому випадку шари будуть закриті, тобто запит повинен проходити всі шари зверху вниз. Для цього є дві причини, одна з яких полягає в тому, що всі «подібні» компоненти повинні бути разом, а інша причина полягає в тому, що вона забезпечує шари ізоляції.

Докладніше, поєднання "подібних" компонентів разом означає, що все, що має відношення до певного шару, залишається в одному шарі. Це дозволяє чітко розділити між типами компонентів, а також допомагає збирати аналогічний код програмування в одному місці. Ізолюючи шари, вони стають незалежними один від одного. Таким чином, якщо, наприклад, ми хочемо змінити базу даних з сервера Oracle на сервер SQL, це спричинить великий вплив на рівень бази даних, але це не вплине на інші шари. Припустімо також, що у вас є власний написаний бізнес-шар і хочете змінити його на механізмі бізнес-правил. Зміна не вплине на інші шари, якщо у нас є чітко визначена багатошарова архітектура.

Шаруватий шаблон архітектури може бути змінений, щоб мати додаткові шари окрім згаданих. Це відомо як гібридна багатошарова архітектура. Наприклад, між бізнес-шаром і шаром збереження даних може бути службовий рівень. Однак це не ідеальний сценарій, оскільки тепер бізнес-шар повинен пройти службовий рівень, щоб дійти до рівня збереження даних. Цей запит не отримує жодної цінності, пройшовши сервісний рівень. Це називається

архітектурним антипатерном. Запити проходять через шари з малою або ніякою логікою, виконаною у кожному шарі. Тому треба бути обачливим та не створювати непотрібні шари.

3.2 Опис кінцевих точок прикладного програмного інтерфейсу

Далі буде приведено опис кінцевих точок, які реалізовано для відповідності до прописаних раніше вимог.

3.2.1 Запит для отримання оцінок студента за семестр

Цей запит використовується для отримання результатів сесійного контролю та результатів атестації певної дисципліни у певному семестрі. Для цього потрібно вказати унікальний ідентифікатор студента та номер семестру. Якщо інформація існує в сховищі даних, то ми її отримаємо у відповіді запиту.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/marks

Параметри запиту перераховані в табл. 3.1.

Структура відповіді у форматі JSON:

```
[
  {
    "disciplineName": "string",
    "studyTypeName": "string",
    "mark": 5,
    "firstAttestation": null,
    "secondAttestation": null,
    "additionalPoints": null
  },
  {...}
]
```

Таблиця 3.1. Опис параметрів запиту для отримання оцінок

Назва	Тип	Опис	Приклад
studentId	string	Унікальний ідентифікатор студента	5YCHRMMNX73Y
semesterId	number	Номер семестру навчання	1

3.2.2 Запит для отримання довідкової інформації про студента

Цей запит використовується для отримання довідкової інформації про студента, а саме назву факультету, номер телефону, кафедру, номер наказу про зарахування. Це потрібно для формування довідок про навчання в автоматичному режимі.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/certificates

Параметри запиту перераховані в табл. 3.2.

Структура відповіді у форматі JSON:

```
{
  "facultyAddressEng": "string",
  "tel": "string",
  "facultyUpper": "string",
  "pib": "string",
  "vin": "string",
  "zak": "string",
  "vud": "string",
  "vud1": "string"
}
```

Таблиця 3.2. Опис параметрів запиту для отримання довідкової інформації

Назва	Тип	Опис	Приклад
studentId	string	Унікальний ідентифікатор студента	5YCHRMMNX73Y

3.2.3 Запит для отримання оцінок групи за певний предмет

Цей запит використовується для отримання всіх оцінок студентів певної групи за вказаний семестр для вказаного предмету. У відповіді можна знайти ідентифікатор студента, прізвище, ім’я, по-батькові, результат оцінювання та результат перездачі предмету. Цю інформацію можна використовувати для генерації відомостей для певної дисципліни.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/student-marks

Параметри запиту перераховані в табл. 3.3.

Структура відповіді у форматі JSON:

```
[
  {
    "testResultId": "string",
    "studentId": "string",
    "firstName": "string",
    "middleName": "string",
    "lastName": "string",
    "rateValue": -1,
    "retakeRateValue": 5
  },
  {...}
]
```

Таблиця 3.3. Опис параметрів запиту для отримання оцінок групи

Назва	Тип	Опис	Приклад
sessionRegisterId	string	Унікальний ідентифікатор предмету	P4YNP4S1OONN

3.2.4 Запит для отримання предметів на сесії для викладача

Цей запит використовується для отримання дисциплін, котрі потрібно перевірити під час сесії, або переглянути інформацію про сесію, котра вже минула. Це може знадобитися для розуміння який тип контролю розрахований для певного предмету. У відповіді ви отримуєте список предметів для певної сесії у викладача.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/teacher-disciplines

Параметри запиту перераховані в табл. 3.4.

Структура відповіді у форматі JSON:

```
[
  {
    "sessionRegisterId": "string",
    "disciplineName": "string",
    "studyTypeName": "string",
    "studyTypeId": 9,
    "studyGroupName": "string"
  },
  {...}
]
```

Таблиця 3.4. Опис параметрів запиту для отримання предметів певної сесії

Назва	Тип	Опис	Приклад
employeeId	string	Унікальний ідентифікатор викладача	0ZLWYMRV445R

3.2.5 Запит для отримання ПІБ студента

Цей запит використовується для отримання прізвища, імені, по-батькові, поточний семестр, назву факультету та назву групи для певного студента. Це довідкова інформація, котра може бути використана для відображення в інтерфейсі користувача.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/full-names/student

Параметри запиту перераховані в табл. 3.5.

Структура відповіді у форматі JSON:

```
{
  "firstName": "string",
  "middleName": "string",
  "lastName": "string",
  "lastSemester": 2,
  "facultyName": "string",
  "groupName": "string"
}
```

Таблиця 3.5. Опис параметрів запиту для отримання ПІБ студента

Назва	Тип	Опис	Приклад
studentId	string	Унікальний ідентифікатор студента	5YCHRMMNX73Y

3.2.6 Запит для отримання ПІБ викладача

Цей запит використовується для отримання прізвища, імені, по-батькові, назву посади та вчений ступінь певного викладача. Це довідкова інформація, котра може бути використана для відображення в інтерфейсі користувача.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/full-names/teacher

Параметри запиту перераховані в табл. 3.6.

Структура відповіді у форматі JSON:

```
{
  "firstName": "string",
  "middleName": "string",
  "lastName": "string" ,
  "divisionName": "string",
  "positionName": "string"
}
```

Таблиця 3.6. Опис параметрів запиту для отримання ПІБ викладача

Назва	Тип	Опис	Приклад
employeeId	string	Унікальний ідентифікатор викладача	0ZLWYMRV445R

3.2.7 Запит для отримання поточного семестру для студента

Це службовий запит, котрий використовується для отримання номеру поточного семестру для певного студента.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/information/current-semester

Параметри запиту перераховані в табл. 3.7.

Структура відповіді у форматі JSON:

```
{
  "currentSemester": "3"
}
```

Таблиця 3.7. Опис параметрів запиту для отримання поточного семестру

Назва	Тип	Опис	Приклад
studentId	string	Унікальний ідентифікатор студента	5YCHRMMNX73Y

3.2.8 Запит для отримання середнього балу за семестр

Цей запит використовується для отримання середнього балу певного студента за певний навчальний семестр. В результаті можна будувати графіки успішності студентів певної групи, або графіки досягнень студента за весь період навчання.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/marks/average

Параметри запиту перераховані в табл. 3.8.

Структура відповіді у форматі JSON:

```
{
  "averageMark": 94.55
}
```

Таблиця 3.8. Опис параметрів запиту для отримання середнього балу

Назва	Тип	Опис	Приклад
studentId	string	Унікальний ідентифікатор студента	5YCHRMMNX73Y
semesterId	number	Номер семестру навчання	1

3.2.9 Запит для пошуку студента

Цей запит використовується для пошуку певного студента за його ПІБ та факультетом. В результаті ми отримуємо список студентів, що підійшли під дані запиту. Кількість результатів у відповіді обмежена зі сторони серверу, з міркувань безпеки.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/search/student

Параметри запиту перераховані в табл. 3.9.

Структура відповіді у форматі JSON:

```
[
  {
    "studentId": "string",
    "facultyName": "string",
    "groupName": "string",
    "firstName": "string",
    "middleName": "string",
    "lastName": "string"
  },
  {...}
]
```

Таблиця 3.9. Опис параметрів запиту для пошуку студента

Назва	Тип	Опис	Приклад
firstName	string	Ім’я повністю або частина	Іван
lastName	string	Прізвище повністю або частина	Іванов
middleName	string	По-батькові повністю або частина	Іванович
facultyId	string	Ідентифікатор факультету (обов’язковий параметр)	20

3.2.10 Запит для пошуку співробітника

Цей запит використовується для пошуку певного співробітника за його ПІБ та факультетом, за яким він закріплений. В результаті ми отримуємо список співробітників, що підійшли під дані запиту. Кількість результатів у відповіді обмежена зі сторони серверу, з міркувань безпеки.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/search/employee

Параметри запиту перераховані в табл. 3.10.

Структура відповіді у форматі JSON:

```
[
  {
    "employeeId": "string",
    "divisionName": "string",
    "positionName": "string",
    "scientificRank": "string",
    "asscientificDegree": "string",
    "firstName": "string",
    "middleName": "string",
    "lastName": "string",
  },
  {...}
]
```

Таблиця 3.10. Опис параметрів запиту для пошуку співробітника

Назва	Тип	Опис	Приклад
firstName	string	Ім’я повністю або частина	Іван
lastName	string	Прізвище повністю або частина	Іванов
middleName	string	По-батькові повністю або частина	Іванович
facultyId	string	Ідентифікатор факультету (обов’язковий параметр)	20

3.2.11 Запит для отримання списку факультетів

Це службовий запит, який використовується для отримання списку факультетів. Цей список корисний при пошуку студента чи співробітника. Оскільки запит працює з інформацією, котра не часто оновлюється, то для нього

додано кешування, щоб збільшити швидкість відповіді та зменшити навантаження на базу даних.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/information/faculty-list

Запит працює без вхідних параметрів.

Структура відповіді у форматі JSON:

```
[
  {
    "id": "string",
    "abbreviation": "string",
    "fullName": "string",
   },
   {...}
]
```

3.2.12 Запит для виставлення оцінки за предмет

Цей запит використовується для виставлення оцінки за певну дисципліну під час певної сесії. Може бути корисним викладачам для зручного вводу інформації в базу даних навчального закладу.

Тип HTTP запита: PUT

Адрес кінцевої точки: http://apis.kpi.ua/api/tests/take

Тип контенту, що передається в тілі запиту: application/json

Структура запиту у форматі JSON:

```
{
  "testResultId": "string",
  "studentId":"string",
  "rateValue":100,
  "studyTypeId":9
}
```

Структура відповіді у форматі JSON:

```
{
  "success": true
}
```

3.2.13 Запит для виставлення оцінки за перескладання предмету

Цей запит використовується для виставлення оцінки за перескладання дисципліни під час додаткової сесії. Може бути корисним викладачам для зручного вводу інформації в базу даних навчального закладу.

Тип HTTP запита: PUT

Адрес кінцевої точки: https://localhost:44389/api/tests/retake

Тип контенту, що передається в тілі запиту: application/json

Структура запиту у форматі JSON:

```
{
  "testResultId": "string",
  "studentId":"string",
  "rateValue":100,
  "studyTypeId":9
}
```

Структура відповіді у форматі JSON:

```
{
  "success": true
}
```

3.2.14 Запит для генерації та відправки тимчасового коду

Це запит для генерування тимчасового коду та відправки смс повідомлення користувачу, що запитує доступ до системи. Зі сторони серверу обмежено частоту відправки нових кодів для уникнення великої кількості запитів. Це допоміжний запит для автентифікації користувача в системі.

Тип HTTP запита: POST

Адрес кінцевої точки: http://apis.kpi.ua/api/identities/secret

Параметри запиту перераховані в табл. 3.11.

В якості відповіді, сервер відправляє HTTP код.

Таблиця 3.11. Опис параметрів запиту для відправки смс

Назва	Тип	Опис	Приклад
phone	string	Номер телефону	0XXYYYYYYY

3.2.15 Запит для отримання ідентифікатора студента

Це запит для отримання ідентифікатора користувача. Якщо введений код відповідає тому, що був раніше згенерований та відправлений користувачу, то у відповідь він отримує ідентифікатор, який може використовувати для подальшої роботи з системою.

Тип HTTP запита: GET

Адрес кінцевої точки: http://apis.kpi.ua/api/identities

Параметри запиту перераховані в табл. 3.12.

Структура відповіді у форматі JSON:

```
{
  "personId": "string",
  "isStudent": true
}
```

Таблиця 3.12. Опис параметрів запиту для отримання ідентифікатора

Назва	Тип	Опис	Приклад
phone	string	Номер телефону	0XXYYYYYYY
secret	string	Тимчасовий код	1898

3.3 Алгоритм автентифікації користувача

Важко посперечатися, що сьогодні кожен студент чи викладач має, якщо не смартфон, то хоча б мобільний телефон. Тому основним елементом процесу автентифікації користувача є генерація тимчасового коду та відправка його на номер телефону.

Давайте більш детально розглянемо сам алгоритм автентифікації користувача. Він складається з двох запитів, котрі виконуються послідовно. Перший – для генерації тимчасового коду, другий – для перевірки тимчасового коду та, у випадку співпадіння, відправки ідентифікатора користувача. Далі покроково наведено алгоритм генерації тимчасового коду:

– отримати запит для генерації тимчасового коду з параметром, я кому міститься номер телефону для автентифікації;

- якщо номер телефону відсутній, чи не відповідає визначеному формату, то згенерувати відповідь зі статусним кодом 400 та текстом, що повідомляє про помилку у вхідних даних;
- вирахувати значення, за допомогою хеш функції, для отриманого номеру телефону;
- перевірити чи існує в кешу попередньо згенерований тимчасовий код, для хешу отриманого номеру телефону;
- якщо код уже існує, та він був згенерований менше ніж 60 секунд тому, то згенерувати відповідь зі статусним кодом 400 та текстом, що повідомляє про обмеження частоти генерації кодів;
- відправити запит до БД для перевірки чи існує студент чи викладач із таким номером телефону;
- якщо користувач з таким номером телефону не існує, то згенерувати відповідь зі статусним кодом 404, та текстом, що повідомляє про відсутність користувача із такими даними;
- згенерувати тимчасовий чотиризначний код;
- вирахувати значення, за допомогою хеш функції, для тимчасового коду;
- відправити тимчасовий код на номер телефону, що був отриманий в запиті;
- записати в кеш, використовуючи як ключ, хеш номеру телефону, та, як значення, хеш тимчасового коду та дату генерації коду, виставивши час життя запису в 10 хв;
- згенерувати відповідь зі статусним кодом 200, що означає успіх операції;
- якщо під час виконання запиту сталася помилка, то згенерувати відповідь зі статусним кодом 500 та текстом, що містить пояснення помилки.

Далі покроково наведено алгоритм перевірки тимчасового коду та відправки токену користувачу:

- отримати запит для отримання токену, з параметрами, в яких містяться номер телефону та тимчасовий код;
- якщо номер телефону відсутній, чи не відповідає визначеному формату, то згенерувати відповідь зі статусним кодом 400 та текстом, що повідомляє про помилку у параметрі з номером телефону;
- якщо тимчасовий код відсутній, чи не відповідає визначеному формату, то згенерувати відповідь зі статусним кодом 400 та текстом, що повідомляє про помилку у параметрі з тимчасовим кодом;
- вирахувати значення, за допомогою хеш функції, для тимчасового коду;
- вирахувати значення, за допомогою хеш функції, для номеру телефону;
- перевірити чи існує в кешу попередньо згенерований тимчасовий код, для хешу отриманого номеру телефону;
- якщо значення в кешу відсутнє, то згенерувати відповідь зі статусним кодом 404 та текстом, що повідомляє про відсутність потрібних даних та необхідності їх генерації;
- якщо значення в кешу не відповідає хешу, отриманого значення, то згенерувати відповідь зі статусним кодом 400 та текстом, що повідомляє про невідповідність отриманих даних;
- відправити запит до БД для отримання ідентифікатора користувача та його типу;
- видалити з кешу запис з хешем, що вирахуваний для номеру телефону, в якості ключа;
- згенерувати відповідь зі статусним кодом 200 та ідентифікатором користувача та його роллю.
- якщо під час виконання запиту сталася помилка, то згенерувати відповідь зі статусним кодом 500 та текстом, що містить пояснення помилки.

З огляду на інформацію наведену вище, ми маємо алгоритм, що складається з двох частин, для автентифікації користувача з великою точністю та складністю до підробки даних. Варто зазначити, що в алгоритмі генерації коду

здійснено захист від спаму, за допомогою обмеження частоти генерації нових повідомлень. Також обмежено час актуальності коду, за рахунок встановлення часу життя запису в кешу. Що також знімає необхідність ручного видалення застарілих даних з сховища, для уникнення його переповнення та збереження працездатності системи. На рис. 3.2 можна побачити приклад смс повідомлень із тимчасовими кодами. Можна помітити, що генеровані коди не повторюються, є випадковими та не можливо підібрати алгоритм для їх визначення. Що, в свою чергу, є ознакою високої надійності системи.

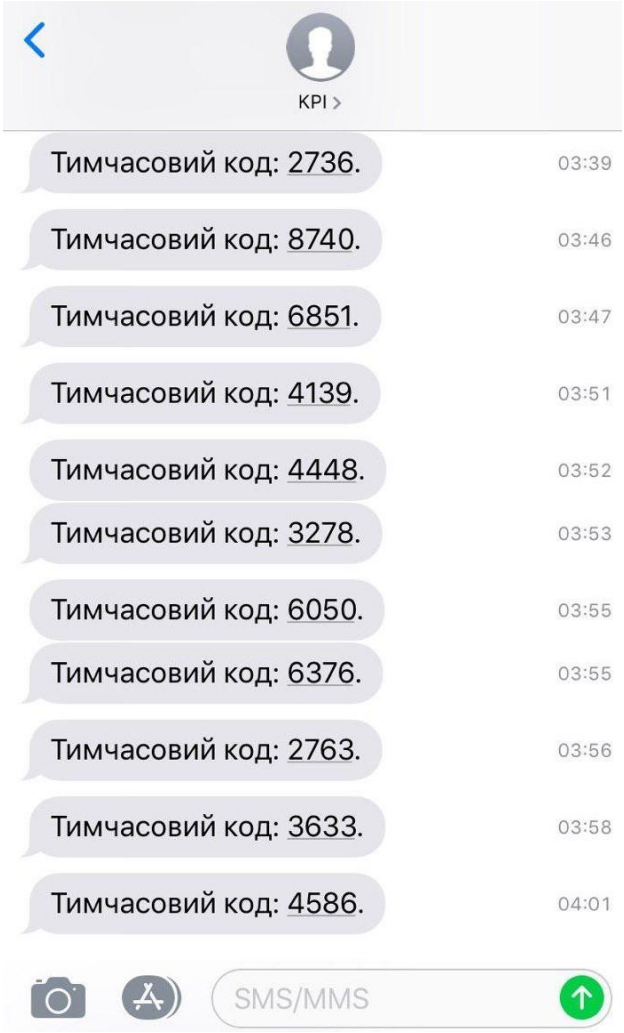


Рисунок 3.2. Приклад смс із тимчасовими кодами

3.4 Опис структури тестової моделі

Забезпечення якості (QA) – це будь-який систематичний процес визначення, чи відповідає продукт чи послуга визначеним вимогам.

QA встановлює та підтримує вимоги щодо розробки або виготовлення надійних виробів. Система забезпечення якості покликана підвищити довіру клієнтів та надійність компанії, а також покращити робочі процеси та ефективність, а також дає змогу компанії краще конкурувати з іншими.

ISO (Міжнародна організація зі стандартизації) є рушійною силою практики QA та відображення процесів, що використовуються для впровадження QA. QA часто поєднується з міжнародним стандартом ISO 9000. Багато компаній використовують ISO 9000, щоб забезпечити свою систему забезпечення якості та ефективність.

Концепція забезпечення якості як формалізована практика розпочалася у виробничій галузі, і з тих пір вона поширилася на більшість галузей, включаючи розробку програмного забезпечення.

Забезпечення якості допомагає компанії створювати продукти та послуги, що відповідають потребам, очікуванням та вимогам клієнтів. Це дає високоякісні пропозиції товарів, які формують довіру та вірність клієнтів. Стандарти та процедури, визначені програмою забезпечення якості, допомагають запобігти дефектам продукції до їх виникнення.

Для забезпечення якості використовується один з трьох методів:

– тест на відмову – постійно тестує виріб, щоб визначити, чи він зламається чи вийде з ладу. Для фізичних продуктів, яким потрібно витримати стрес, це може включати випробування виробу на тепло, тиск або вібрацію. Для програмних продуктів тестування несправності може включати розміщення програмного забезпечення в умовах високого використання або завантаження.
– статистичний контроль процесів – методологія, заснована на об'єктивних даних та аналізі. Ця методологія використовує статистичні методи управління та контролю виробництва продукції.
– тотальне управління якістю – застосовує кількісні методи як основу для постійного вдосконалення. Даний вид спирається на факти, дані та аналіз для підтримки планування продукту та огляду ефективності.

Забезпечення якості програмного забезпечення (SQA) систематично знаходить закономірності та дії, необхідні для покращення циклів розвитку. Пошук та виправлення помилок кодування може мати непередбачувані наслідки: можна виправити одне, але порушити інші функції та функціонал одночасно.

SQA стала важливою для розробників як засіб уникнення помилок до їх виникнення, економивши час і витрати на розробку. Навіть із встановленими процесами SQA, оновлення програмного забезпечення може порушити інші функції та викликати дефекти - загальновідомі як помилки.

Існували численні стратегії SQA. Наприклад, Capability Maturity Model Integration (CMMI) є моделлю SQA, орієнтованою на підвищення продуктивності. CMMI працює шляхом ранжування рівнів зрілості у межах організації та визначає оптимізації, які можуть бути використані для вдосконалення. Рівні рангів коливаються від неорганізованого до максимально оптимального.

З часом розвивалися методології розробки програмного забезпечення, які покладаються на SQA, такі як Waterfall, Agile та Scrum. Кожен процес розробки прагне оптимізувати ефективність роботи.

Waterfall – традиційний лінійний підхід до розробки програмного забезпечення. Це покроковий процес, який зазвичай включає збір вимог, формалізацію дизайну, впровадження коду, тестування коду та виправлення та випуск. Його часто сприймають як надто повільний, саме тому були побудовані альтернативні методи розвитку.

Agile – це орієнтована на команду методологія розробки програмного забезпечення, де кожен крок робочого процесу підходить як спринт. Розробка програмного забезпечення Agile вкрай адаптивна, але менш прогнозована, оскільки сфера проекту може легко змінитися.

Scrum – це поєднання обох процесів, де розробники розбиваються на команди для вирішення конкретних завдань, і кожне завдання розділяється на кілька спринтів.

Отже, враховуючи вище сказане, тестування є невід’ємним кроком в розробці якісного програмного забезпечення. Враховуючи один із ключових недоліків поточної системи «Кампус», а саме відсутність можливості працювати під навантаженням, одним із найбільших акцентів при тестування спрямовано на тестування під навантаженням. Потрібно бути впевненим, що система адекватно реагує на пікові навантаження, оброблює всі запити, та здатна до самостійного відновлення, у випадку збою під час роботи. Окрім того, під час такого тестування потрібно отримувати інформацію про внутрішній стан системи. Це потрібно для вчасного реагування на виключні ситуації, що виникають під час обробки запиту, аналізу та пошуку слабких місць системи (база даних, кеш).

Для аналізу поведінки системи використовується сервіс Datadog. Datadog – це служба моніторингу хмарних додатків, що забезпечує моніторинг серверів, баз даних, інструментів та служб через платформу аналітики даних на основі SaaS.

Створити достатнє навантаження при такому тестуванні за допомогою власних ресурсів доволі важко, тому використано сервіс, який допомагає у вирішення цього питання. На рис. 3.3 можна побачити результат імітації роботи 50 користувачів, котрі протягом 20 хвили постійно генерують запити до системи.

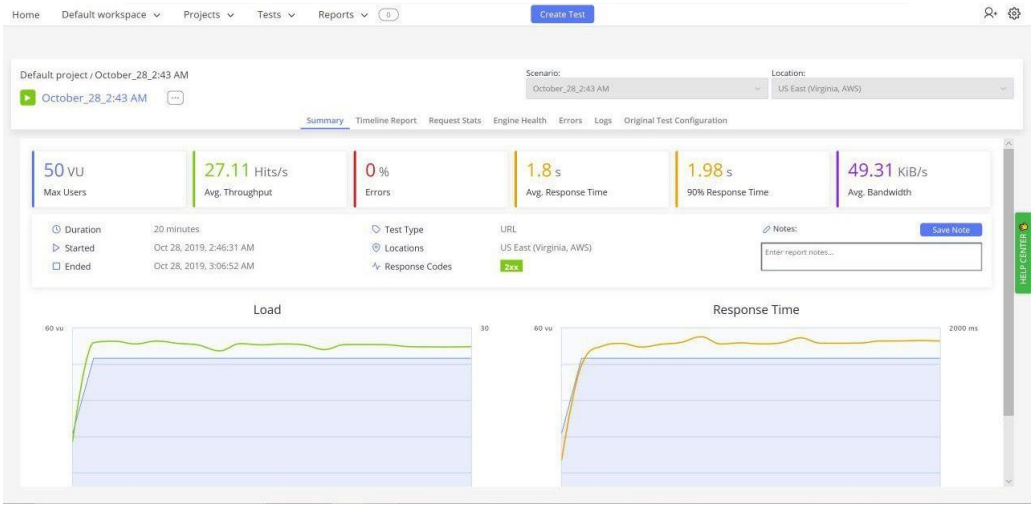


Рисунок 3.3. Сервіс для тестування під навантаженням

Як і очікувалось, система є стабільною та легко витримує такий потік запитів. На рис. 3.4 можна побачити, що за цей час було відправлено більше 70 000 запитів, і не отримано жодної помилки, що доводить високу надійність

додатку. На рис. 3.5 можна побачити яке було навантаження на систему, скільки витрачено ресурсів при обробці вхідної інформації. Оскільки, навантаження на CPU не перевищувало 20%, то можна зробити припущення, що на поточному апаратному забезпеченні система може витримувати до 300 000 запитів. Це є доволі хорошим показником, зважаючи на те, що поточна кількість студентів – 30 000.

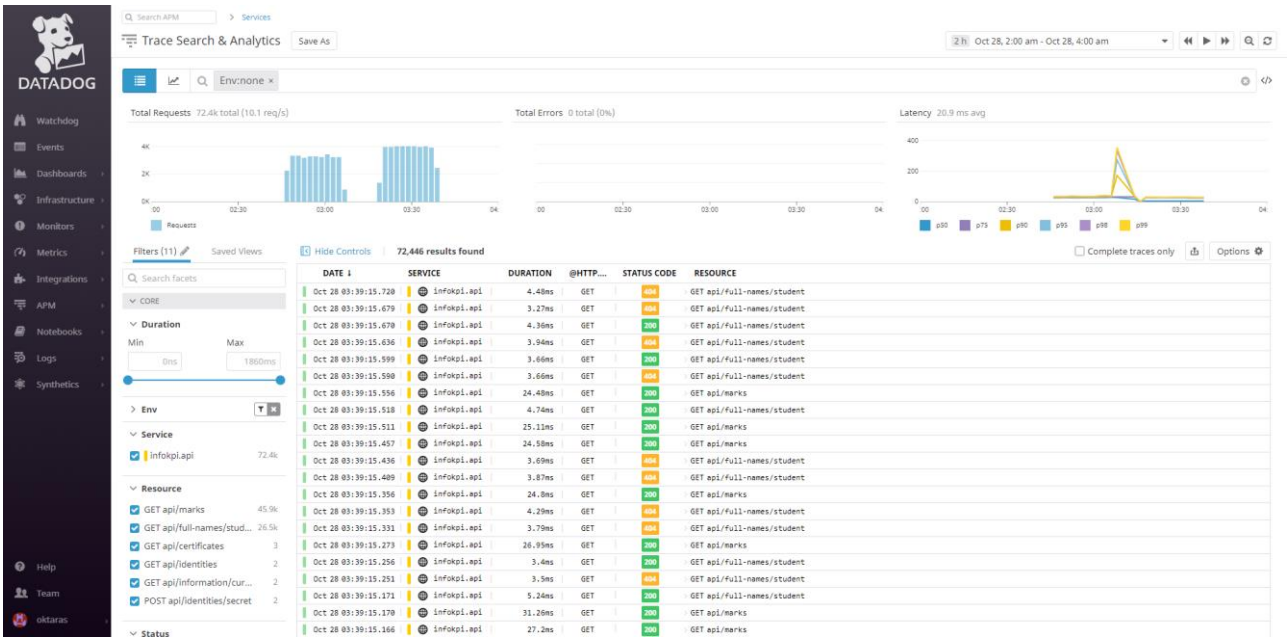


Рисунок 3.4. Результати тестування системи

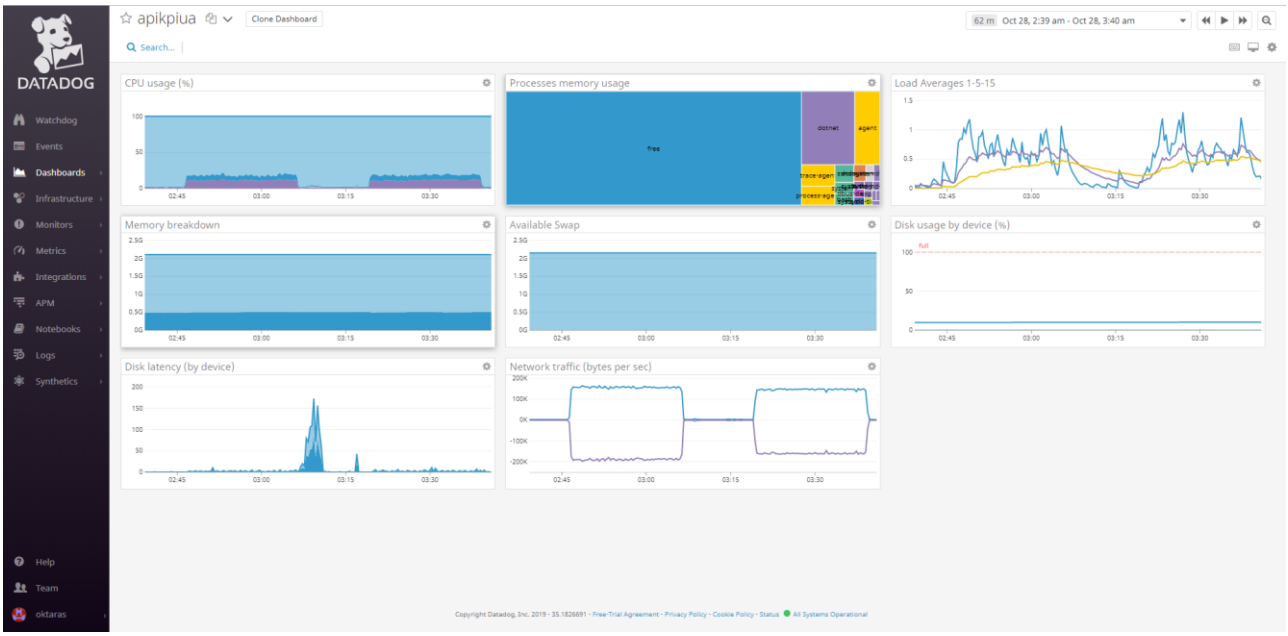


Рисунок 3.5. Моніторинг завантаженості серверу під час тестування

3.5 Вимоги до апаратного та програмного забезпечення

Для роботи системи потрібно мати середовище, де буде виконуватися запуск вихідного коду. В свою чергу, для запуску програмного забезпечення потрібно апаратне забезпечення. Тому, далі я наведу вимоги до апаратного та програмного забезпечення, котрі потрібні для адекватної та стабільної роботи системи.

3.5.1 Вимоги до апаратного забезпечення

Для запуску мінімальної операційної системи, необхідної для роботи системи потрібно таке апаратне забезпечення:

- процесор 1.6 Ггц, 1 ядро ЦП або краще;
- оперативна пам’ять не менше ніж 2 Гб, DDR4;
- ПЗУ не менше ніж 32 Гб, SSD;
- доступ до мережі Internet.

3.5.2 Вимоги до програмного забезпечення

Для запуску вихідного програмного коду системи потрібно мати таке програмне забезпечення:

- БД Oracle RDBMS 11g;
- середовище виконання вихідного коду ASP.NET Core Runtime 3.0.0;
- кеш-кластер Redis 4.0.0;
- операційна система:
    - Windows 10;
    - Mac OS X 10.10 або вище;
    - Ubuntu 14.04 або інша Linux-подібна.

3.6 Керівництво користувача

Оскільки, першочерговим завданням стояло побудувати надійну та стабільну систему, здатну швидко обробляти запити та витримувати великі

навантаження, то розробка візуального клієнту відійшла на другий план. Але для тестування та відладки системи все ж потрібно мати хоча б якийсь клієнт. Завдяки дотриманню специфікації OpenAPI, за допомогою інструменту Swashbuckle для ASP.NET Core згенеровано візуальний клієнт для використання спроектованого прикладного програмного інтерфейсу. Подальше керівництво користувача буде описано на основі цього клієнту.

В системі є кілька кінцевих точок, котрі можна використовувати без авторизації. Це точки, котрі не мають ніякої конфіденційної інформації і можуть використовуватися для отримання довідкової інформації. Тому вони відразу готові до використання і не потребують ніяких додаткових дій. Такі точки не мають позначення з піктограмою замку. На рис. 3.6 можна побачити одну із таких точок, котра призначена для отримання списку факультетів та інститутів. Цю інформацію можна використовувати для наповнення даними випадаючих списків при пошуку студента чи викладача.

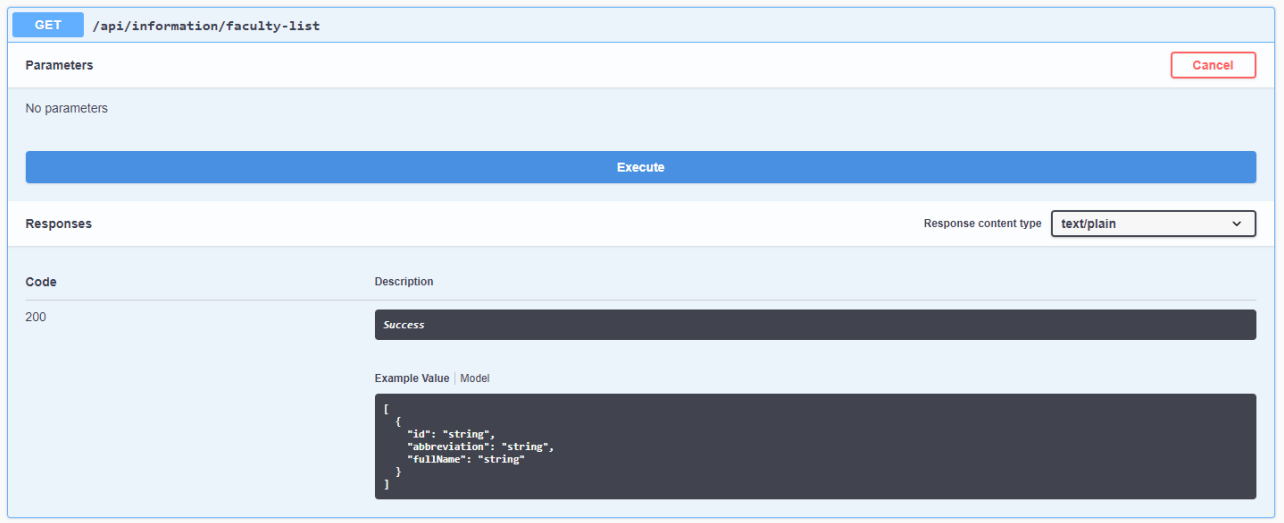


Рисунок 3.6. Кінцева точка для отримання списку факультетів

Для відправки запитів, котрі містять конфіденційну інформацію чи надають доступ до спеціального функціоналу, як то виставлення оцінок за предмет, потрібно пройти авторизацію. Це гарантує, що користувач зможе отримати доступ тільки до власної інформації і зможе виконувати тільки ті дії, котрі передбачені для його ролі.

Давайте детальніше розглянемо процес авторизації та отримання токену, котрий буде використовуватися для підтвердження особи користувача. Спочатку потрібно отримати тимчасовий код на власний номер телефону, для цього потрібно зробити запит на генерацію цього тимчасового коду. На рис. 3.7 можна побачити приклад відправки цього запиту та отримані результати.

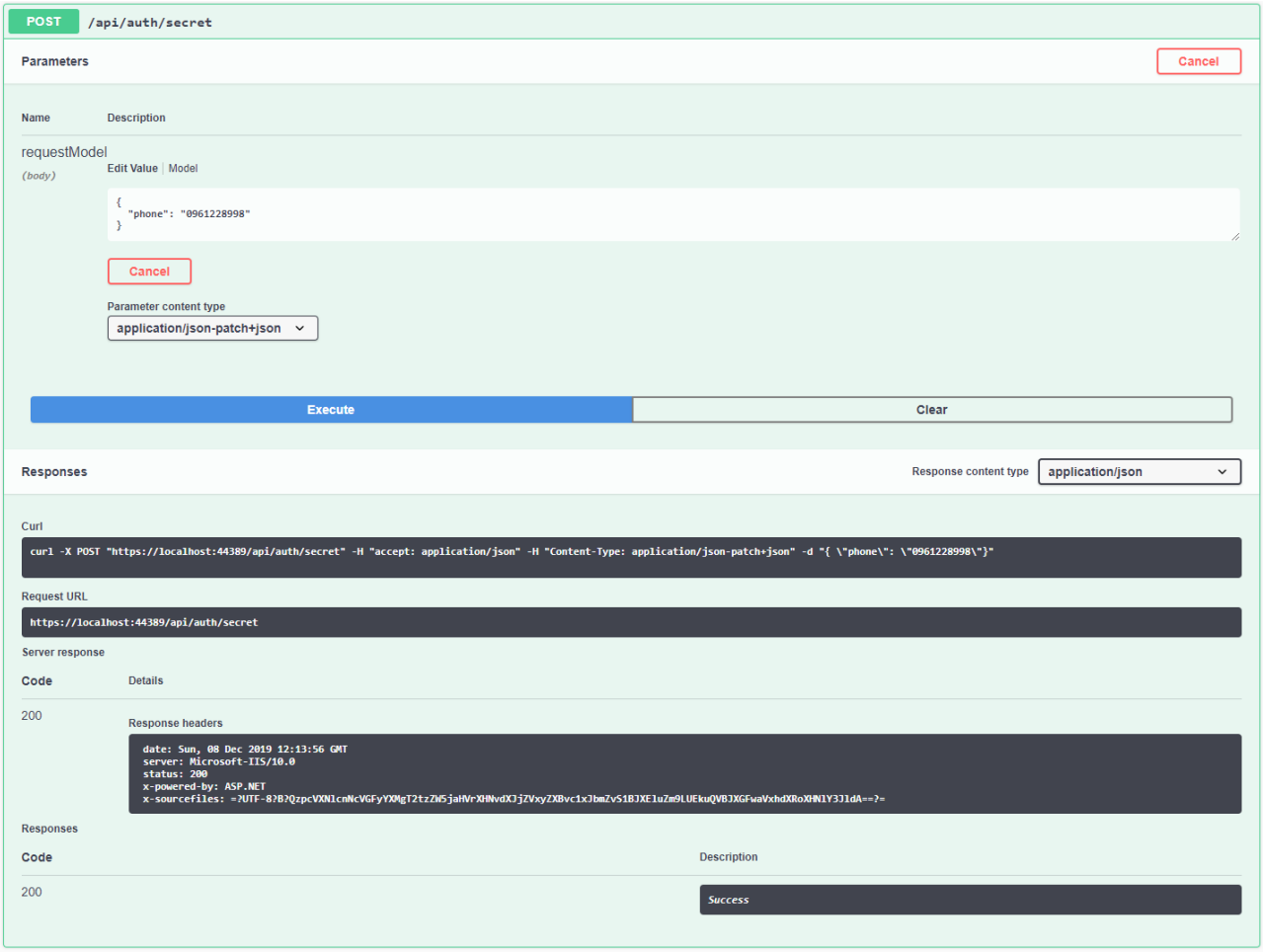


Рисунок 3.7. Кінцева точка для генерації тимчасового коду

Після обробки запиту сервером на вказаний номер телефону приходить смс повідомлення з тимчасовим кодом. Тепер цей код потрібно застосувати для отримання пари JWT токенів. Токен доступу потрібно використовувати при відправці запитів до захищених кінцевих точок, додаючи його в якості заголовку. Завдяки токену перезавантаження, після закінчення часу актуальності токену доступу, можна обміняти цю пару токенів на нову. Таким чином, не потрібно кожен раз генерувати тимчасовий код. На рис. 3.8 можна побачити приклад відправки запиту для отримання пари токенів та результат, котрий згенеровано

сервером. Тепер, маючи токен доступу, можна записати його в клієнт. І цей токен буде автоматично додаватися в заголовок при відправці запиту до захищеної кінцевої точки. На рис. 3.9 можна побачити форму для вводу цього токену.

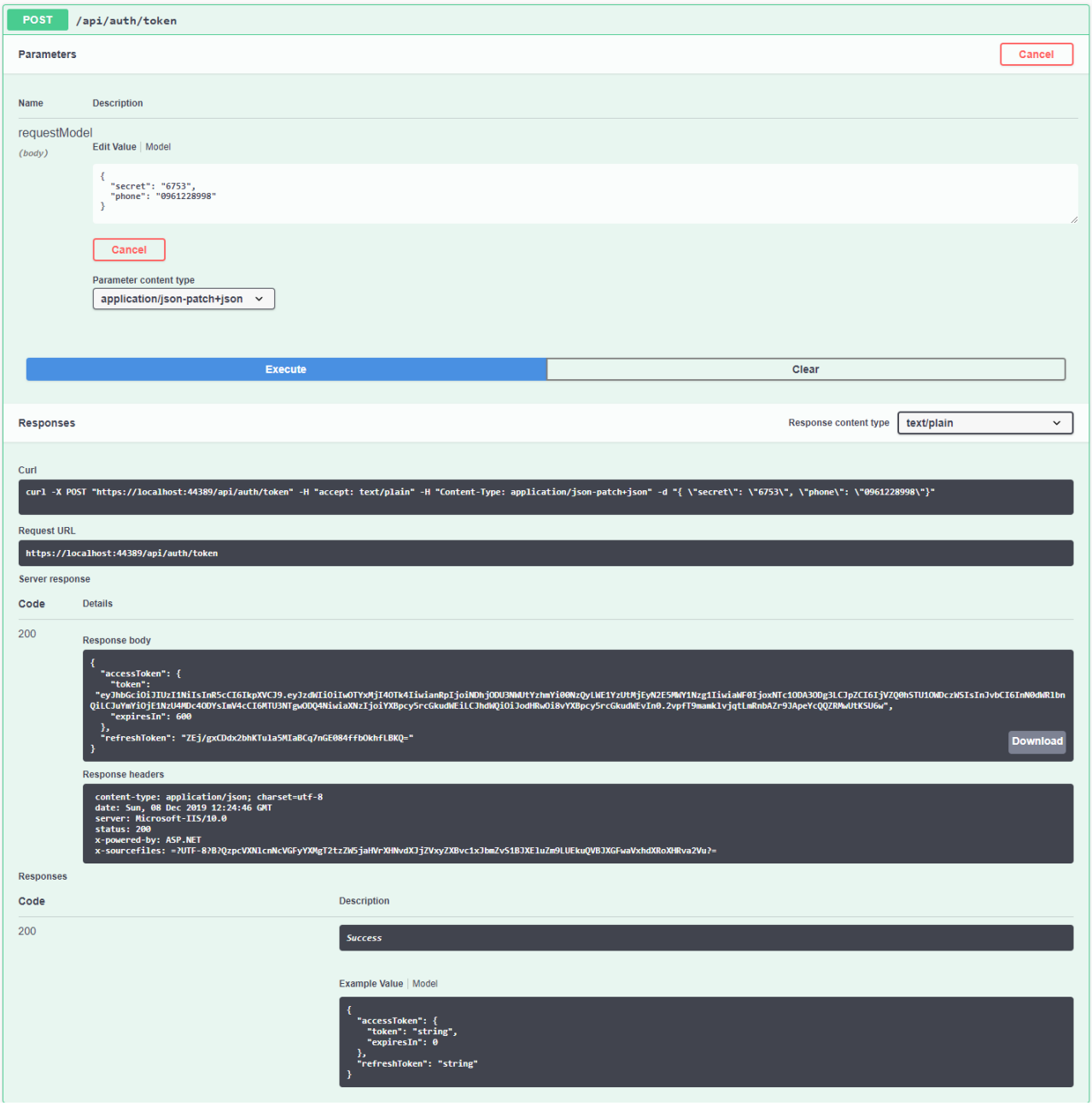


Рисунок 3.8. Кінцева точка для отримання пари токенів

Таким чином, зробивши кілька простих кроків ми отримали доступ до захищених кінцевих точок. Але якщо ми отримали доступ як студент, то ми все одно не можемо здійснювати запити, котрі призначені для викладача. На рис. 3.10 можна побачити запит на виставлення оцінки, та генеровану сервером відповідь, що даний функціонал не доступний для поточного користувача.

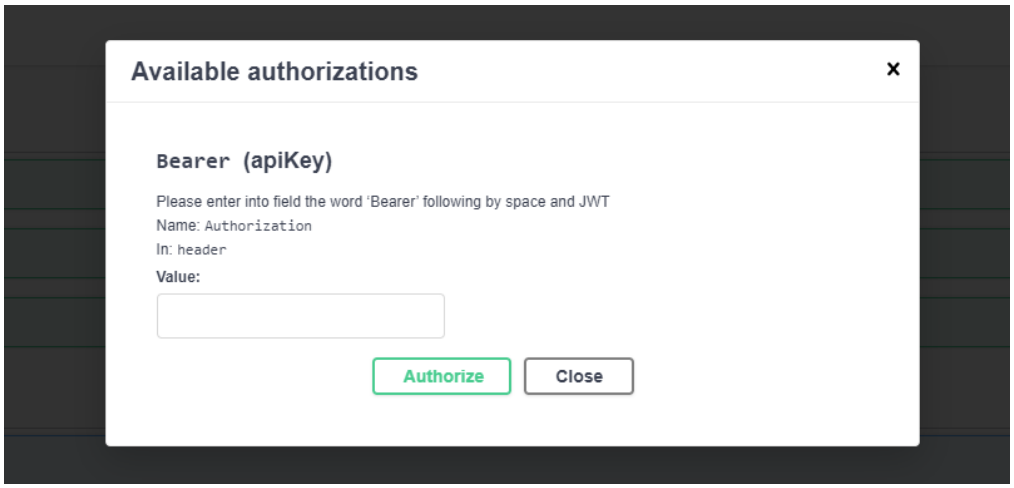


Рисунок 3.9. Форма для вводу токену доступу

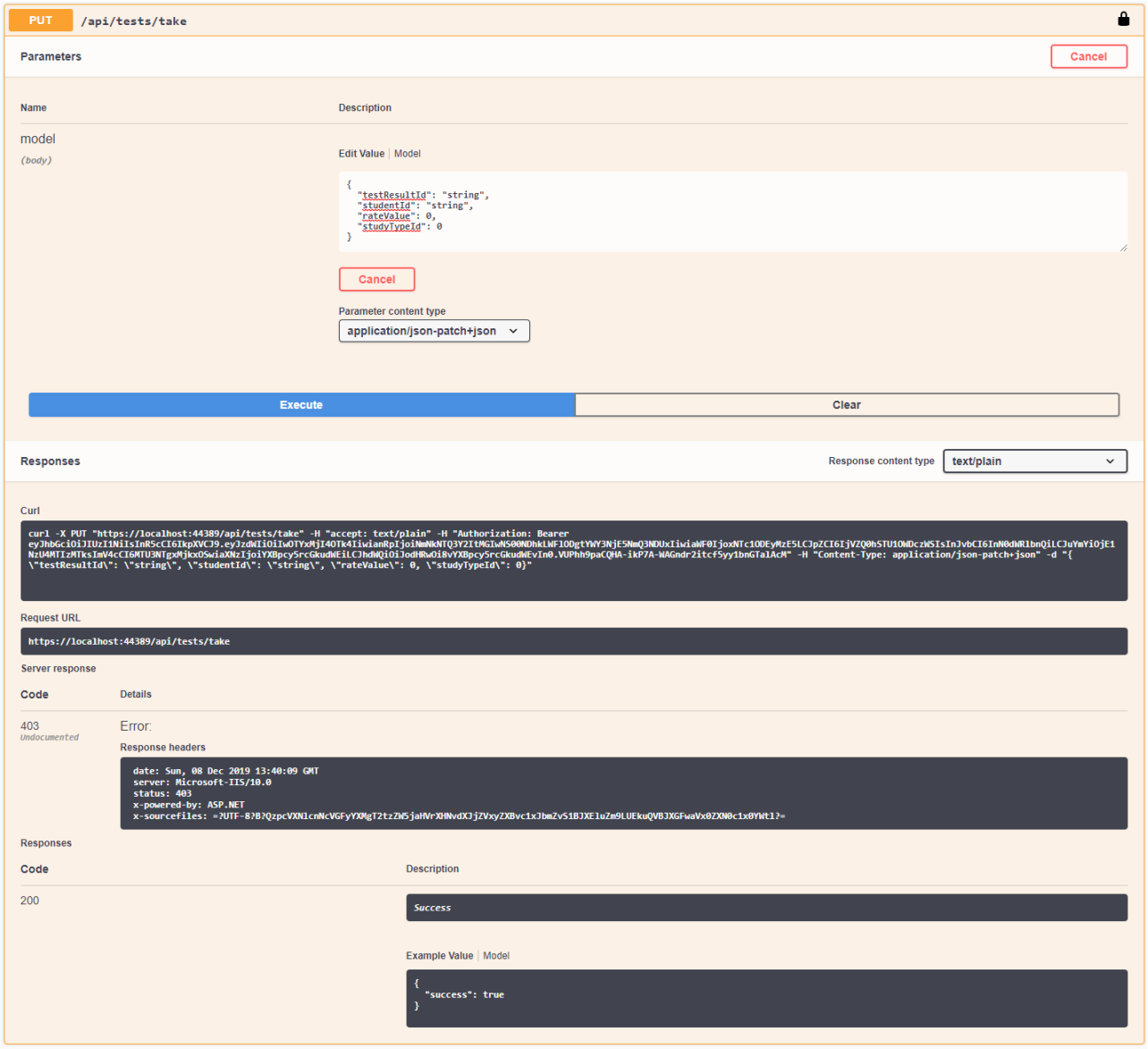


Рисунок 3.10. Кінцева точка для виставлення оцінок

Висновки до розділу

В цьому розділі описано етап розробки алгоритмічного та програмного забезпечення системи безпечного доступу до інформації навчального закладу. Детально описано архітектуру програмного продукту, алгоритм автентифікації користувача, кожну кінцеву точку прикладного програмного інтерфейсу, структуру тестової моделі та технічні вимоги, що необхідні для роботи системи. Також складено керівництво користувача для ознайомлення з основними принципами користування системою.

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Ідея проекту полягає в створенні системи, що буде надавати безпечний доступ до даних навчального закладу. Розглянемо зміст ідеї , можливі напрямки застосування, основні переваги, які зможе отримати користувач представлено у табл. 4.1.

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Безпечний доступ до даних навального закладу	Інтеграція зі школами для відображення успішності учнів	Можливість отримати онлайн-доступ до даних про власну успішність
	Інтеграція з університетами для відображення успішності студентів	Можливість отримати онлайн-доступ до даних про атестації та семестровий контроль

Даний проект відрізняється тим, що надає безпечний доступ до даних навчального закладу, та відділяє інформацію та функціонал залежно від того, з якою роллю користувач знаходиться в системі.

Далі проведено аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників). Визначено перелік техніко-економічних властивостей та характеристик ідеї.

На ринку є два конкуренти, котрі надають схожі можливості, але завжди задовольняють потреби чи обмеження навчальних закладів.

Проведено порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (табл. 4.2).

Таблиця 4.2. Визначення характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Продукція конкурентів			*W* (слабка сторона)	*N* (нейтральна сторона)	*S* (сильна сторона)
		Мій проект	«Електронний Кампус»	«Google Classroom»			
1	Швидкість роботи	Висока	Низька	Середня	Потрібно мати вільний простір в оперативній пам'яті для кешування інформації	Простота реалізації	Кешування даних для швидкого доступу
2	Зручність використання	Середня	Низька	Висока	Необхідність реалізації певних процедур та функцій	Відкритий прикладний програмний інтерфейс	Дотримання основних принципів RESTful
3	Вимоги до забезпечення	Низькі	Високі	Високі	Необхідність у використанні платної системи управління базами даних	Підтримання тільки кількох останніх версій	Безкоштовна система для кешування даних
4	Кросплатформ-ність	Наявна	Наявна	Наявна	Не всі потрібні компоненти можуть працювати на всіх платформах, що підтри-муються	Підтримка всіх основних операційних систем	Можливість запуску на різних платформах

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту наведено в табл. 4.3

Таблиця 4.3. Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Доступ до даних навчального закладу	Запити до сховища даних за допомогою структурованої системи запитів	База даних Oracle та мова SQL	Повністю доступна для користування
2	Кешування інформації, що рідко змінюється	Запис даних у сховище, що має велику швидкість відгуку	Кеш-кластер Redis	Має відкритий сирцевий код та вільну систему розповсюдження
3	Обмеження доступу до ресурсів та функціоналу за допомогою автентифікації	Автентифікація користувача на основі токенів доступу та перезавантаження	Відкритий JWT стандарт токену доступу	Відкритий стандарт, що має бібліотеки для роботи з токеном для найпопулярніших інструментів побудови веб-додатків
4	Система для безпечного доступу до даних навчального закладу	Використання всіх технологій, що перераховані вище	База даних Oracle та мова SQL. Кеш-кластер Redis. Відкритий JWT стандарт токену доступу	Всі технологій є відкритими та вільними для використання
Обрана технологія реалізації ідеї проекту: 4				

Отже, зважаючи на інформацію в попередній таблиці, технологічна реалізація продукту – можлива. Всі вибрані технології, які можуть нам допомогти розробити якісний продукт, є відкритими, доступними та не мають обмежень у використанні. Техніко-економічні характеристики повинні бути такими, щоб система могла конкурувати із поточними конкурентами.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку, що можна побачити в табл. 4.4.

Таблиця 4.4. Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн./ум.од	Не відомо
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	70

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим та надає змогу реалізувати продукцію на національному ринку, оскільки економічний розвиток надає необхідний результат для вдалої економічної реалізації проекту.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5. Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в доступі до інформації про власні досягнення у навчальному процесі	Учні шкіл	Передбачається, що будуть щоденні оновлення інформації, оскільки учні кожного дня мають змогу отримати оцінку за певний предмет	Зручність у використанні. Швидка робота системи. Спроможність швидко освоїти як користуватися системою. Можливість впроваджувати користувачам свій функціонал у систему для більшого розвитку проекту серед інших цільових аудиторій
2	Потреба в контролі своїх успіхів та вчасне реагування на прогалини у знаннях	Студенти	Передбачається, що можуть виникати пікові навантаження в періоди проведення міжсесійного та сесійного контролю	

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту (табл. 4.6), та факторів, що йому перешкоджають (табл. 4.7). Фактори в таблиці подаються в порядку зменшення значущості.

Таблиця 4.6. Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Зменшення ціни на поставлену послугу. Розробка унікальних характеристик товару. Надання ліцензій на обслуговування

Закінчення таблиці 4.6

2	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення додаткових інвесторів, мотивація роботи на перспективу. Ітеративна розробка продукту задля покрокового виведення продукту на ринок та отримання відповіді користувачів
3	Вихід аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	Вихід товару на ринок в коротші строки з не повною, але достатньою, функціональністю для зацікавлення усіх цільових аудиторій. Проведення рекламної компанії

Таблиця 4.7. Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Вихід на ринок. Зменшення монополії. Надання нових рішень у сфері	Розробка нової функціональності. Вихід нової продукції на ринок. Надання різноманітних типів ліцензій в залежності від потреб користувача/замовника
2	Вихід аналогу	Надати продукт з певними характеристиками та можливостями що відсутні у компаній конкурентів	Аналіз ринку та користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки за ціну, котра є дешевшою ніж у продуктів-замінників
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів системи кешування даних
4	Грошова винагорода за рекламу	При достатньому попиту на систему кешування даних можлива комерціалізація продукту на основі реклами задля отримання грошової винагороди для подальшого розвиту продукту та оплати заробітної плати працівникам	Точкова комерціалізація продукту. Введення реклами. Ведення додаткових коштів у проект задля його подальшого розвитку

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (табл. 4.8).

Таблиця 4.8. Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар від кожної компанії на ринку, являється недосконалим замінником товару, реалізованого іншими фірмами. На ринку є умови для входу та виходу. Ціна корелює між суперниками	Розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники. Кореляція цін у відповідності до товарів замінників. Різни типи ліцензій
2	Рівень конкурентної боротьби: світовий	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Вихід на ринок збуту продукту з клієнто-необхідною функціональністю. Налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів. Надання бета-версій продукту
3	Галузева ознака: внутрішньогалузева	Даний тип продукту може використовуватися тільки у сфері розробки IT додатків/продуктів	Надання зручного, інтуїтивно зрозумілого інтерфейсу. Підтримка всім відомих методів взаємодії з середовищем розробки. Наявність документації та он-лайн підтримки
4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція – конкуренція між товарами одного виду	Впровадження функціональності яка відсутня у товарів-замінників. Спрощення інтерфейсів. Надання підтримки
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація. Не цінова – надання функціональності, що відсутня у товарах-замінниках	Надання платних ліцензій лише на критично важливу функціональність для клієнта з певним строком підтримки, що зазначена у відповідній ліцензії. Впровадження унікальної функціональності
6	За інтенсивністю: марочна	Наявність унікального знаку що відрізняє даний продукт від продуктів-замінників	Впровадження власної назви та власного знаку

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (табл. 4.9).

Таблиця 4.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Інші компанії, що розробляють програмне забезпечення	Нові компанії-розробники програмного забезпечення	Відсутні	Навчальні заклади	Можна замінити виключно системами з подібним функціоналом
Висновки	Попри затребуваність, інтенсивність конкуренції не надто висока	Наявні можливості виходу на ринок і нових компаній-розробників програмного забезпечення	-	Потенційні споживачі можуть висувати власні вимоги до функціоналу розроблюваної системи	Необхідний постійний й усебічний моніторинг ринку для своєчасного виявлення конкуруючих систем

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею.

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

На основі аналізу конкуренції, проведеного в табл. 4.9, а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6-4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 4.10.

Таблиця 4.10. Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Прагматичність	Через запуск стартапу система буде не дуже складно з точки зору архітектури перший час. Через певний період із додаванням функціоналу та оптимізації алгоритмів роботи програмний код буде все складнішим. Такий етап наступить не раніше одного року постійної роботи над проектом
2	Зручність	Оскільки стартап розробляється на багатьох платформах з різною шириною екранів, то зручність використання системи на різних пристроях буде відігравати не малу роль у спроможності конкурувати з іншими гравцями ринку
3	Швидкість роботи	Швидкість роботи відіграє велику роль для користувачів, оскільки вони не будуть готові чекати декілька хвилин на виведення результату роботи додатку
4	Оптимізація	Якщо додаток буде дуже часто видавати помилки при роботі, то користувачі не будуть вважати додаток надійним
5	Налаштування під користувача	Різні люди мають різні звички, які вони використовують, наприклад, якщо є люди, які люблять працювати за додатком де є темні кольори, а є такі люди, які люблять світлі кольори. Можливість редагувати зовнішній вигляд додатку надає значну перевагу серед конкурентів
6	Відкритість вихідного коду	При наявності вихідного коду будь-який продукт має перспективи розвиватися у багатьох напрямках, особливо таких, які можуть бути неочевидні на перший погляд
7	Приватність	В останні роки приватність людей та інформація щодо них все частіше зловживається шахраями або великими корпораціями, які потребують погодження з умовами доступу до приватної інформації та її обробки
8	Технічна підтримка	Якщо технічна підтримка компанії буде працювати своєчасно та швидко, то це допоможе зберегти репутацію компанії на відміну від конкурентів, де їй не приділяють увагу

Закінчення таблиці 4.10

9	Документація	Будь-який додаток, особливо якщо він має новий функціонал, повинен бути добре роз'яснений своїм користувачам та як його можна використовувати, щоб не мати проблем при подальшій роботі з ним
10	Ціна	Чим дешевше товар, тим більше шансів що його можуть купити, особливо якщо він працює краще ніж конкурентний товар

Підсумовуючи дані, викладені в попередній таблиці, можна зробити висновок про те, що для забезпечення конкурентоспроможності розроблюваного програмного забезпечення необхідно постійно тримати руку на пульсі відповідного ринку та його новинок.

На основі здійсненого аналізу конкуренції, конкурентного середовища на ринку, з урахуванням усіх загроз та можливостей, що надаються розроблюваним програмним забезпеченням, можна виділити фактори конкурентоспроможності, що впливатимуть на затребуваність та популярність продукту, та здійснити більш детальний їх аналіз, що й буде зроблено в табл. 4.11 (С.П. – стартап проект, К.1 – Конкурент 1: «Електронний Кампус», К.2 – Конкурент 2: «Google Classroom»).

Таблиця 4.11. Порівняльний аналіз сильних та слабких сторін системи

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Прагматичність	15	К.1			К.2			С.П.
2	Зручність	8		К.1				С.П.	К.2
3	Швидкість роботи	5	К.1					С.П.	К.2
4	Оптимізація	5	К.1					К.2	С.П.
5	Налаштування під користувача	10	К.1			К.2			С.П.
6	Відкритість вихідного коду	20	К.1	К.2		С.П.			
7	Приватність	20	К.1					К.2	С.П.
8	Технічна підтримка	15		К.1	С.П.				К.2

Закінчення таблиці 4.11

9	Документація	15	К.1				С.П.		К.2
10	Ціна	15	К.2			С.П.			К.1

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 4.11).

Таблиця 4.12. SWOT аналіз стартап-проекту

Сильні сторони (*S*): – Прагматичність системи через її легкість роботи; – Простота у використанні; – Наявність відкритого вихідного коду; – Збереження приватності інформації користувача.	Слабкі сторони (*W*): – Мала кількість інтеграцій; – Необхідність вносити зміни в існуючу систему.
Можливості (*O*): – Зворотній зв’язок з клієнтами компанії для спроможності розвивати проект в інші напрямки.	Загрози (*T*): – Складність роботи алгоритму при невиявлених випадках використання додатку.

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 4.13).

Таблиця 4.13. Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс - наявний	2-3 місяці
2	Реклама	Залучення власних коштів для реклами товару	1-2 місяці

Закінчення таблиці 4.13

3	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс - наявний	2-3 тижні
4	Презентація товару на хакатонах й інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці

Після аналізу необхідно зазначити обрану альтернативу.

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 4.14).

Таблиця 4.14. Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Учні шкіл	Присутня	Високий	Майже відсутня	Легка
2	Студенти	Присутня	Середній	Майже відсутня	Середня
Які цільові групи обрано: 1, 2					

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є учні, студенти та викладачі. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу, оскільки для масової аудиторії в цілому надається стандартизований продукт з можливістю розширення функціональності за домовленістю (відповідно до ліцензії).

Таблиця 4.15. Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами. Формування лояльності і прихильності споживачів	Зниження ступеню замінності товару. Прихильність клієнтів. Відмітні властивості товару. Відмітні характеристики товару	Стратегія диференціації

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16).

Таблиця 4.16. Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника та до продукту в табл. 4.14, а також в залежності від обраної базової стратегії розвитку в табл. 4.15 та стратегії конкурентної поведінки в табл. 4.16, розробляється стратегія позиціонування (табл. 4.17). Що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17. Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Зручність	Диференціація	Спроможність економити час на вивченні навчального матеріалу	Спроможність легко дізнатися свої оцінки
2	Відкритість вихідного коду	Диференціація	Перспектива розвитку проекту	Розвиток в сфері освіти
3	Документація	Заняття конкурентної ніші	Можливість краще орієнтуватися у власних успіхах	Швидкий доступ

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18. Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Доступ до інформації про власні досягнення у навчанні	Готове рішення для онлайн-доступу до даних	Висока швидкість
2	Автентифікація користувача системи	Легка та точна автентифікація користувача	Безпечний доступ до даних

Надалі розробляється трирівнева маркетингова модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 4.19).

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
1. Товар за задумом	Система для безпечного доступу до даних навчального закладу		
2. Товар у реальному виконанні	Властивості/характеристики	*М/Нм*	*Вр/Тх/Тл/Е/Ор*
	Зручність	*Нм*	*Е*
	Швидкість роботи	*Нм*	*Тх*
	Оптимізація	*Нм*	*Тх*
	Ціна	*Нм*	*Е*
	Документація	*Нм*	*Тл*
	Технічна підтримка	*Нм*	*Тх*
	Приватність	*Нм*	*Тх*
	Налаштування під користувача	*Нм*	*Ор*
3. Товар із підкріпленням	До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товару		
	Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, патент			

Таблиця 4.20. Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
50 тис. грн	75 тис. грн	300 тис. грн	20 - 70 тис. грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21):

– проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

– вибір та обґрунтування оптимальної глибини каналу збуту;
– вибір та обґрунтування виду посередників.

Таблиця 4.21. Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Вибір послуг на сайті, оплата, постачання послуг	-	Виробник - споживач	Веб-сайт

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22. Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Знають, які саме послуги треба для вирішення задач	Веб-сайт, телефон, зустрічі	Підтримка клієнтів, індивідуальний підхід	Донесення переваг до клієнтів	Допомога у виборі бібліотеки машинного навчання

Як результат створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

В четвертому розділі описано стратегії та підходи з розроблення стартап-проекту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проекту.

Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проекту було встановлено що проект є перспективним.

Розглянуто та вибрано альтернативу впровадження стартап-проекту та доведено доцільність подальшої імплементації проекту.

ВИСНОВКИ

Під час написання магістерської дисертації зроблено такі кроки:

1. Проведено аналіз предметної області, під час якого визначено об’єкт та предмет дослідження. Розглянуто найпопулярніші системи для моніторингу спішності та супроводження навчального процесу. Детально розглянуто їх функціонал та основні можливості. Проведено порівняння швидкості роботи та головних функцій. Сформовано вимоги до розроблюваної системи, визначено ролі користувачів та список можливостей, доступних для кожної з них. В результаті виконання першого розділу визначено список задач, котрі потрібно вирішити для успішної реалізації системи.
2. Наступним кроком було проведення аналізу інформаційного забезпечення. Проаналізовано основні засоби збереження даних, визначено їх переваги та недоліки, для збереження постійних даних обрано СУБД Oracle Database. Визначено, що для підвищення швидкості системи потрібно додати кеш-кластер Redis, в якому буде зберігатися інформація, що часто запитується та рідко змінюється.
3. В якості платформи для побудови системи обрано .NET Core 3, оскільки вона зможе повністю задовольнити попередньо сформовані вимоги. Потім вибрано мову програмування C#, котра має всі необхідні інструменти для написання швидкого коду та повністю підтримується обраною платформою. Та визначено інструмент об’єктно-реляційного відображення Dapper, для зручної та швидкої роботи з базою даних безпосередньо з коду програми.
4. Також визначено основний архітектурний стиль REST для побудови прикладного програмного інтерфейсу, котрий використовується для взаємодії інших додатків із системою.
5. Після цього розроблено алгоритмічне та програмне забезпечення. Для цього підібрано шарувату архітектуру системи, котра повністю

задовільнила визначені раніше вимоги. На основі цієї архітектури визначено та детально описано список кінцевих точок прикладного програмного інтерфейсу, котрі необхідно реалізувати для повноти системи.

6. Далі визначено та детально описано алгоритм автентифікації користувача системи, який складається з двох крокі. Першим кроком є послідовність кроків, необхідних для генерації тимчасового коду та відправки його на телефон користувача, у вигляді смс повідомлення. Другим кроком є перевірка відповідності отриманого та згенерованого коду, та генерація пари токенів для доступу та перезавантаження.
7. Також детально описано структуру тестової моделі, при цьому, сфокусовано особливу увагу на тестуванні під навантаженням, бо це дозволяє визначити наскільки система є працездатною та стабільною.
8. Визначено вимоги до апаратного та програмного забезпечення, необхідних для коректності роботи програмного коду.
9. Для більшої зручності сформовано керівництво користувача, де описано базові принципи роботи з прикладним програмним інтерфейсом та детально прописано кроки, необхідні для автентифікації користувача в системі.
10. Проведено маркетинговий аналіз стартап-проекту «Система для безпечного доступу до даних навчального закладу». В процесі якого, описано ідею проекту, здійснено технологічного аудиту ідеї проекту, проаналізовано ранкові можливості запуску стартап-проект, розроблено ринкову стратегію та маркетингову програму стартап-проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications: science book / Nicolaos Karayiannis, Anastasios N. Venetsanopoulos — Springer Science & Business Media, 2013 — 440 p.
2. Artificial neural networks: science book / B Yegnanarayana — New Delhi : Prentice-Hall of India, 1999 — 461 p.
3. Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives / P. S. Neelakanta, Dolores DeGroff — CRC Press, 1994 — 256 p.
4. Neural Networks: An Introduction / Berndt Müller, Joachim Reinhardt, Michael T. Strickland — Springer Science & Business Media, 1995 — 331 p.
5. Neural Network Computing for the Electric Power Industry: article / Dejan J. Sobajic — Psychology Press, 2013 — 240 p.
6. Principles of Artificial Neural Networks: science book / Daniel Graupe — World Scientific, 2007 — 320 p.
7. Automatic Generation of Neural Network Architecture Using Evolutionary Computation: science book / E. Vonk, L. C. Jain, Ray P. Johnson — World Scientific, 1997 — 182 p.
8. Adaptive Neural Network Control of Robotic Manipulators, book / Tong Heng Lee, Christopher John Harris — World Scientific, 1998 — 381 p.
9. Machine Learning: An Artificial Intelligence Approach: science book / Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell — Elsevier, 2014 — 572 p.
10. Machine Learning: An Algorithmic Perspective: book / Stephen Marsland — CRC Press, 2011 — 406 p.
11. Microsoft SQL Server 2016 Reporting Services: science book / Brian Larson — 5th edition — McGraw Hill Professional, 2016 — 793 p.
12. Professional Microsoft SQL Server 2016 Reporting Services and Mobile Reports: book / Paul Turley — John Wiley & Sons, 2017 — 816 p.
13. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: book / Mark J. Price — Packt Publishing Ltd, 2019 — 818 p.

14. Professional ASP.NET MVC 5: science book / Jon Galloway, Brad Wilson, K. Scott Allen, David Matson — John Wiley & Sons, 2014 — 624 p.
15. Hands on with ASP.NET MVC: Covering MVC 6: book / Rahul Sahay — Vij Books India Pvt Ltd, 2014 — 506 p.
16. IIS 8 Administration: The Personal Trainer for IIS 8.0 and IIS 8.5: book / William Stanek — Stanek & Associates, 2015 — 368 p.
17. An Introduction to the Bootstrap: book / Bradley Efron, R.J. Tibshirani — CRC Press, 1994 — 456 p.
18. Managing Startup Enterprises in Emerging Markets: Leadership Dynamics and Marketing Strategies: book / Ananya Rajagopal — Springer Nature, 2019 — 182 p.
19. 101 Startup Lessons: An Entrepreneur's Handbook: book / George Deeb, Red Rocket Ventures — BlogIntoBook.com, 2013 — 150 p.
20. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.

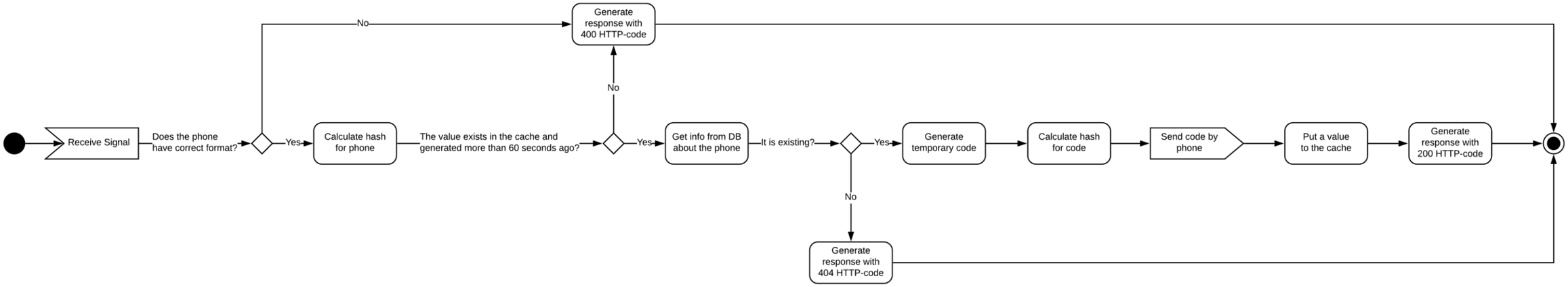
ДОДАТОК А

РЕЗУЛЬТАТИ ПЕРЕВІРКИ НА СПІВПАДІННЯ

ДОДАТОК Б

ГРАФІЧНІ МАТЕРІАЛИ

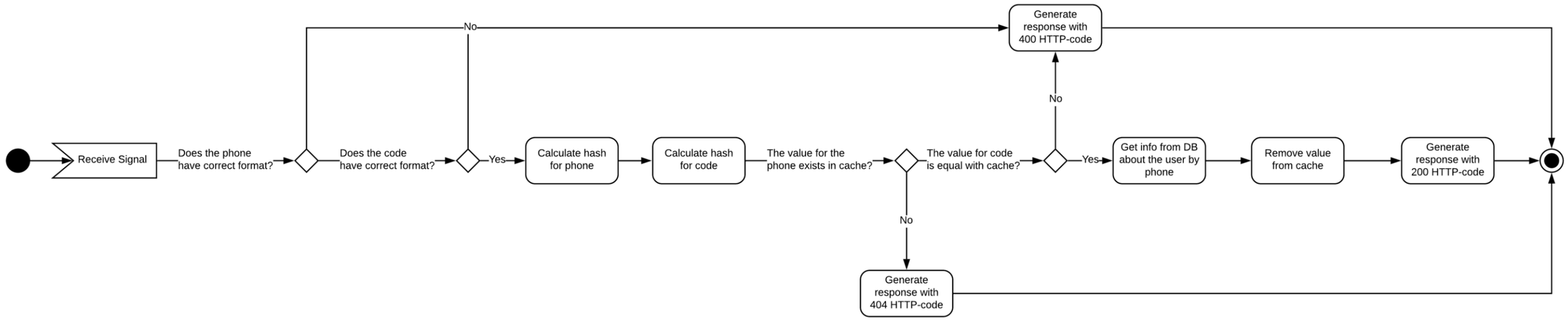
Алгоритм генерації тимчасового коду



Демонстраційний плакат №_1_
до дипломної роботи на тему
„Система для безпечного доступу до даних навчального закладу”

Розробив: __Оксенчук Т. В._
Прийняв: _____

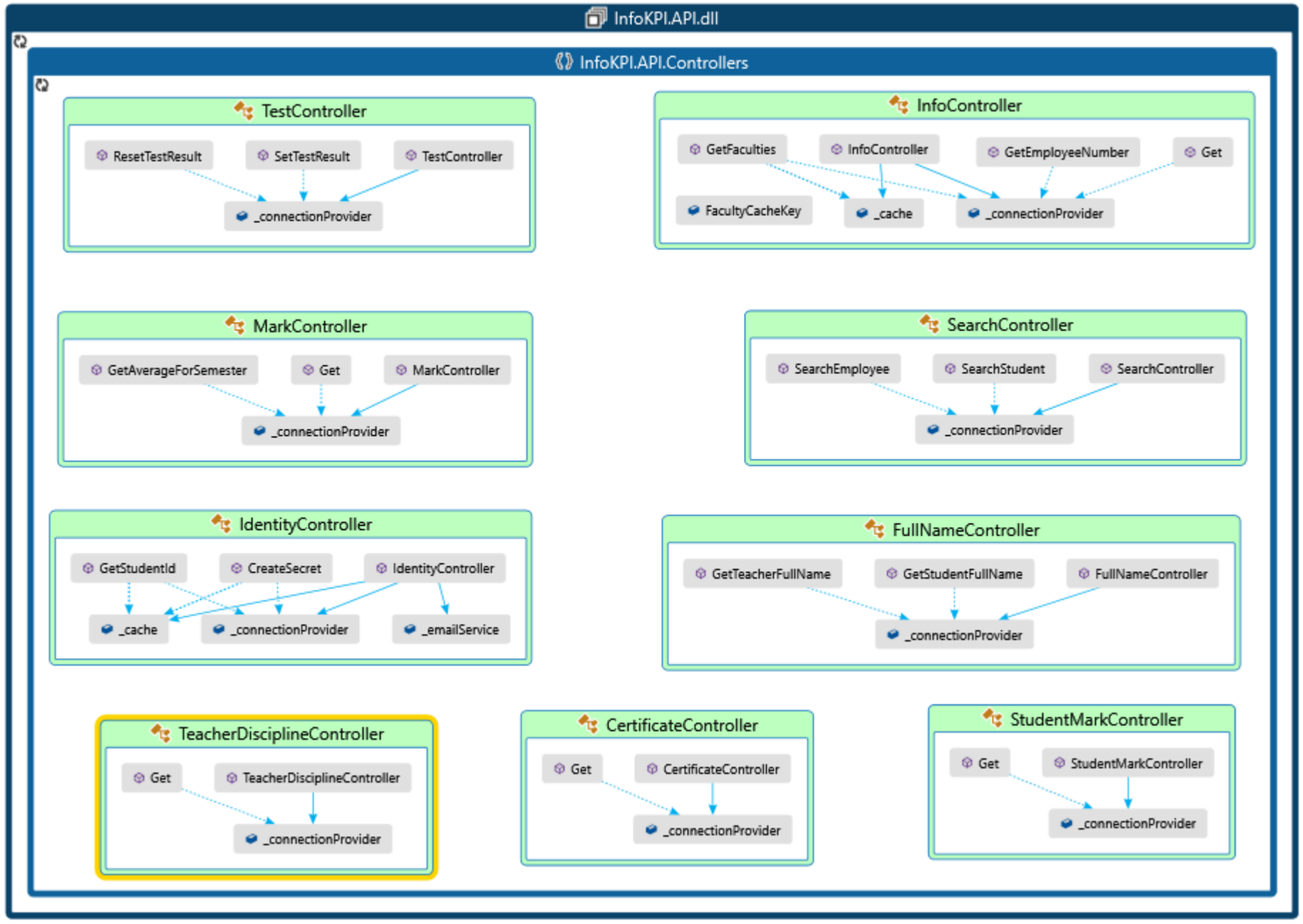
Алгоритм перевірки тимчасового коду та генерації токену доступу



Демонстраційний плакат №_2_
до дипломної роботи на тему
„Система для безпечного доступу до даних навчального закладу”

Розробив: __Оксенчук Т. В._
Прийняв: _____

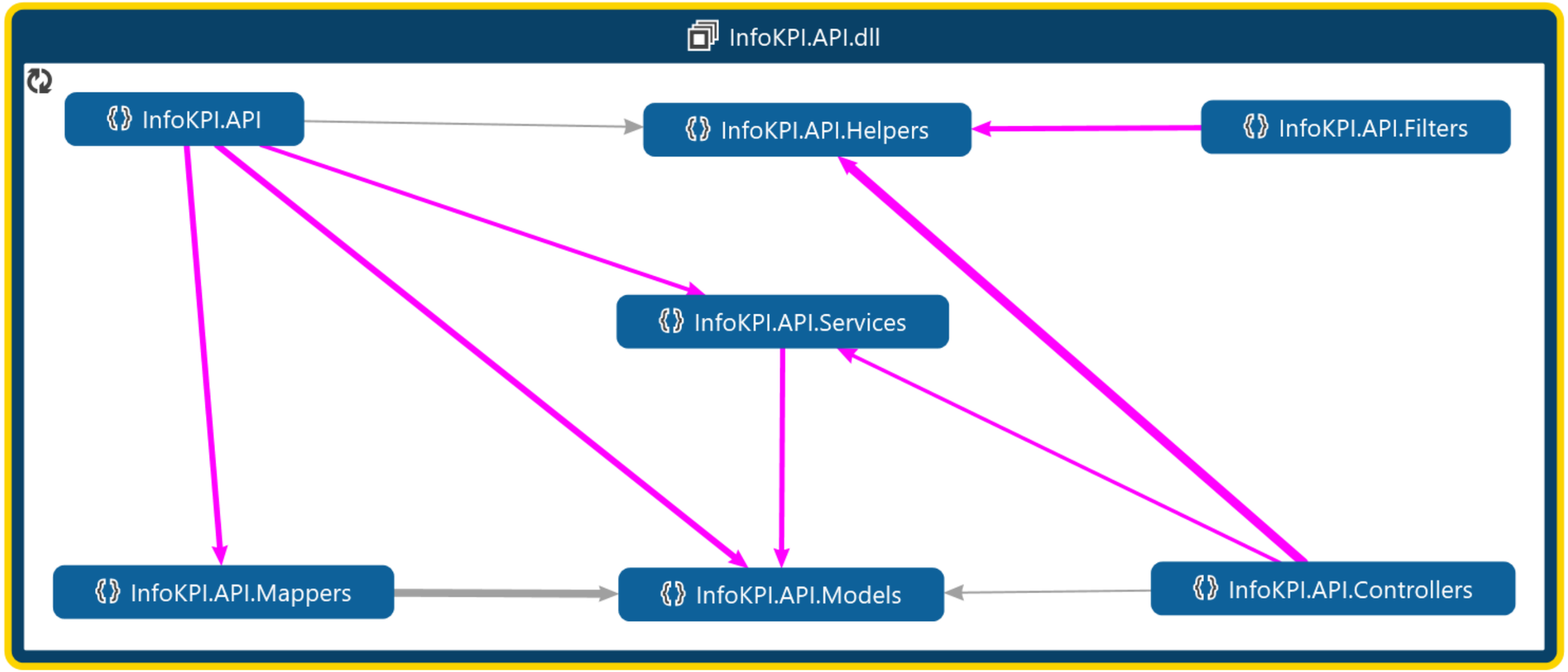
Структура контролерів системи



Демонстраційний плакат №_3_
до дипломної роботи на тему
„Система для безпечного доступу до даних навчального закладу”

Розробив: __Оксенчук Т. В._
Прийняв: _____

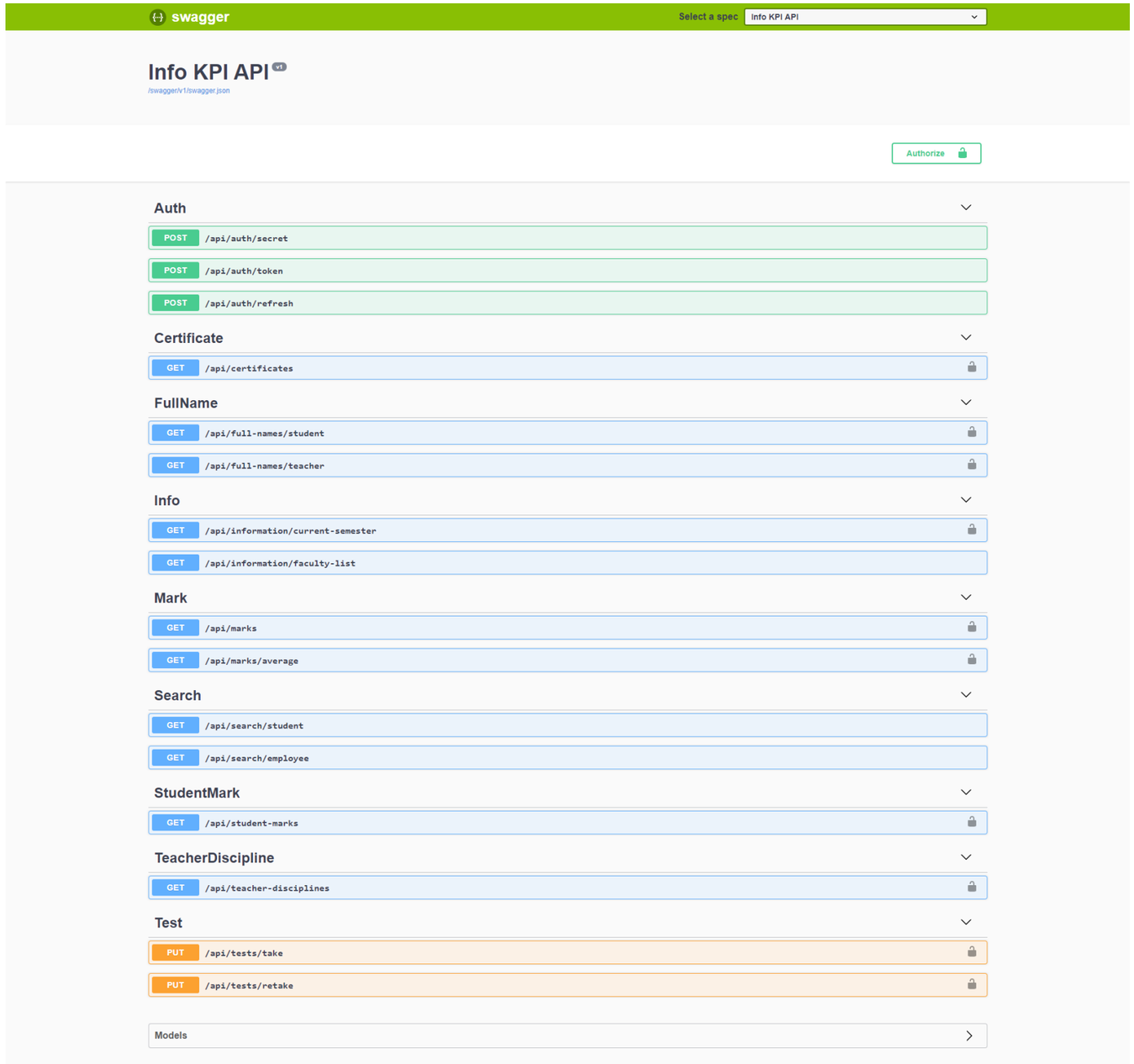
Основні компоненти системи



Демонстраційний плакат №_4_
до дипломної роботи на тему
„Система для безпечного доступу до даних навчального закладу”

Розробив: __Оксенчук Т. В._
Прийняв: _____

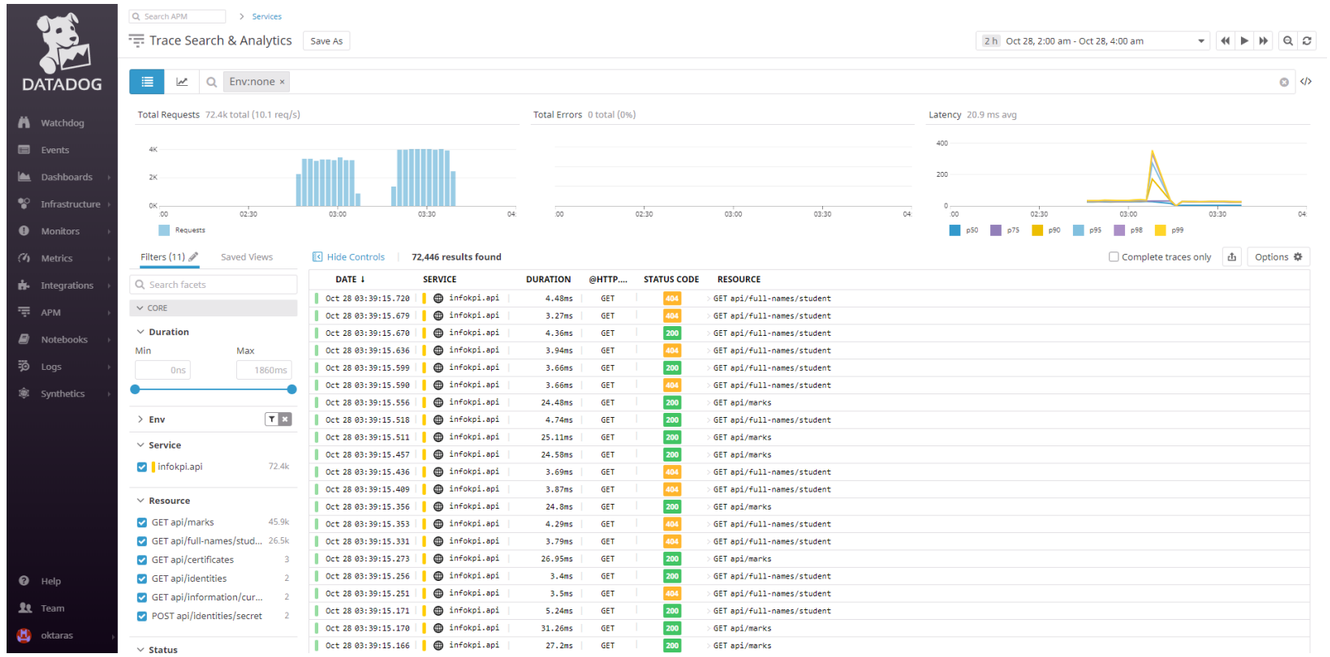
Загальний вигляд клієнтського додатку

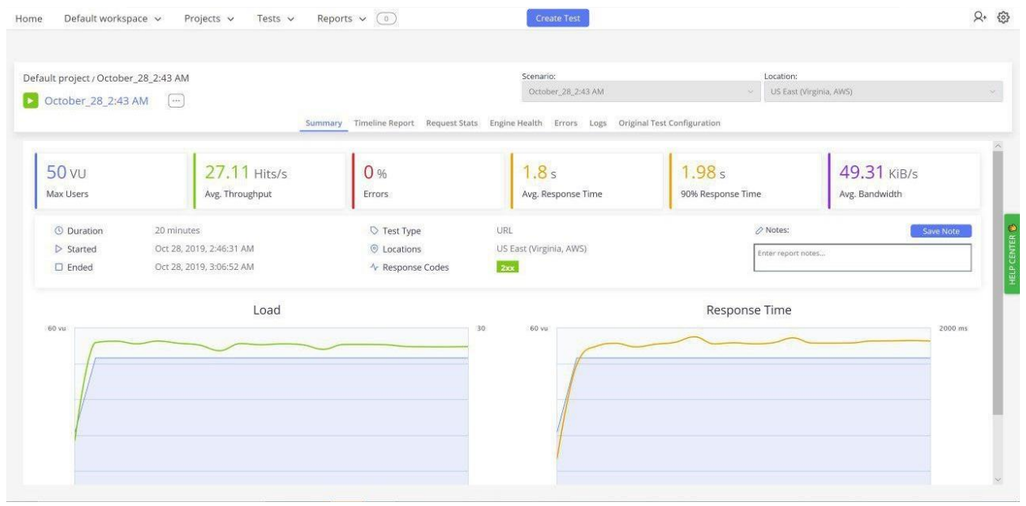


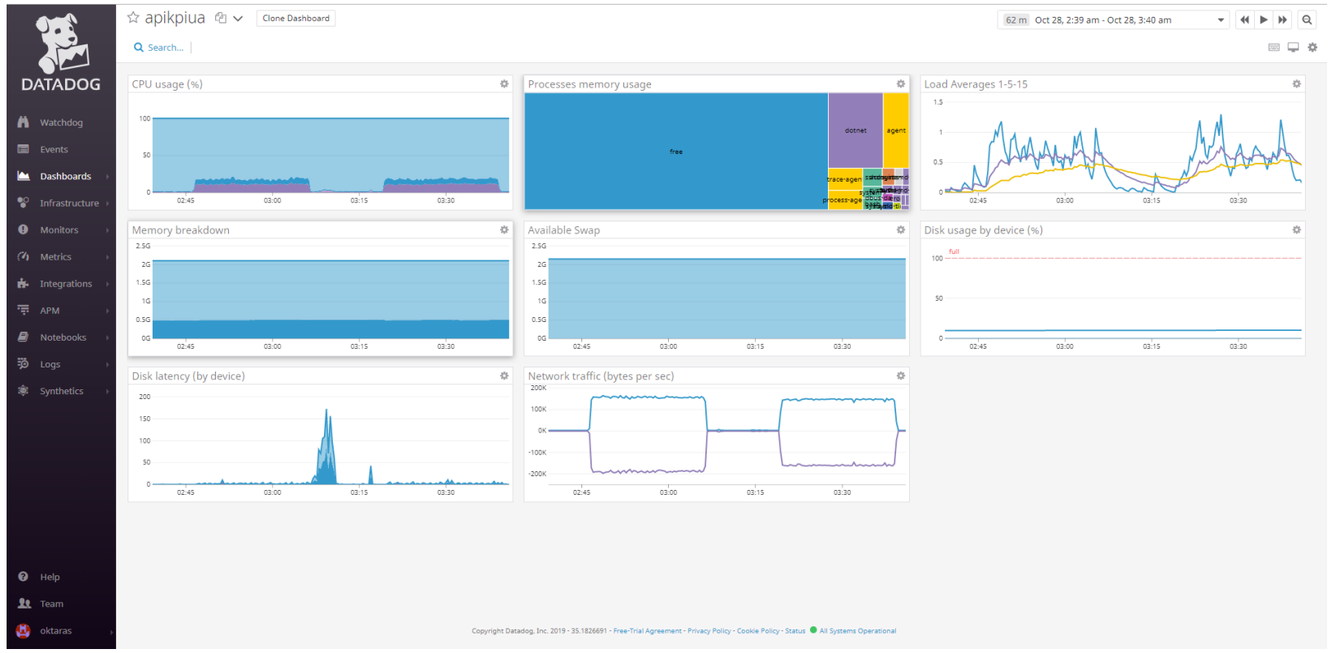
Демонстраційний плакат №_5_
до дипломної роботи на тему
„Система для безпечного доступу до даних навчального закладу”

Розробив: __Оксенчук Т. В._
Прийняв: _____

Результати тестування під навантаженням







Демонстраційний плакат №_6_
до дипломної роботи на тему
„Система для безпечного доступу до даних навчального закладу”

Розробив: _Оксенчук Т.В.__
Прийняв: _____