КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій
(факультет)

інформаційних технологій проєктування та прикладної математики
(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка вебзастосунку для вивчення іноземної мови»

ІМАНОВ ГУСЕЙН
(прізвище, ім’я та по батькові студента повністю)

Київ 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій проєктування та прикладної математики

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТППМ
д.т.н., професор Бородавка Є.В.

_____
„___” _____2026 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка вебзастосунку для вивчення іноземної мови»

Виконав: студент 4-го курсу, групи ІСТ-22

Спеціальності: 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

Іманов Гусейн
(прізвище та ініціали)

Керівник д.т.н., проф. Івахненко І.С.
(прізвище та ініціали)

Рецензент к.т.н., доц. Шабала Є.Є.
(прізвище та ініціали)

Київ, 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Кафедра: інформаційних технологій проєктування та ПМ

Освітній рівень: «бакалавр за ОП»

Спеціальність: 126 «Інформаційні системи та технології»

Освітня програма: «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТППМ
д.т.н., професор Бородавка Є.В.

_____
„___” _____2026 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Іманов Гусейн

1. Тема роботи: Розробка Web-сервісу для впровадження в систему заселення до гуртожитку
затверджена наказом ректора КНУБА № 1857/23.6/25 від «03» 11 2025 р.
2. Керівник роботи: Івахненко Ірина Сергіївна, д.т.н., професор кафедри ІТППМ
3. Строк подання студентом роботи до захисту: червень 2026 року
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Розробка інформаційного забезпечення
Р.3. Розробка програмного забезпечення
Р.4. Ергономіка інформаційних технологій
5. Інформаційні слайди:
С.1. Подкастинг
С.2. Технології та забезпечення
С.3. Трирівнева архітектура
С.4. Діаграма організації БД розроблюваного модуля
С.5. Архітектура. DAL – рівень

С.6. Архітектура. BLL – рівень
С.7. Архітектура. Presentation – рівень
С.9-10. Тестування інтерфейсу + Angular

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2026 р.
Р. 2. Розробка інформаційного забезпечення	Лютий 2026 р.
Р. 3. Розробка програмного забезпечення	Березень 2026 р.
Тестовий приклад програми	Квітень 2026 р.
Р. 4. Ергономіка інформаційних технологій	Травень 2026 р.
Остаточне оформлення роботи	Травень 2026 р.
Направлення роботи на рецензування	Червень 2026 р.
Попередній захист роботи на кафедрі	Червень 2026 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	д.т.н. проф. Терентьєв О.О.		
Прийом програмного продукту	к.т.н., доц. Шабала Є.Є.		

8. Дата видачі завдання: 12 січня 2026 р.

Керівник _____ Івахненко І.С.
(підпис) (прізвище та ініціали)

Бакалавр _____ Іманов Гусейн
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка WEB-застосування для вивчення іноземної мови.

Кваліфікаційна робота бакалавра за спеціальністю: 126 «Інформаційні системи та технології», освітня програма: «Інформаційні системи та технології». – Київський національний університет будівництва та архітектури. – Київ, 2026.

Робота присвячена забезпеченню можливості опанування іноземної мови шляхом застосування системи для відеоподкастингу та подальшого тестування (контроль отриманих знань та корекція подальшого плану навчання).

В результаті роботи було розроблено частину веб-додатку (модуль тестування), створені користувацькі групи, створені користувачі і забезпечена система тестування отриманих знань.

Ключові слова: веб-застосування, інформаційне забезпечення, дистанційне навчання, контроль знань, подкасти, відоподкасти, тести.

SUMMARY

WEB application development for a foreign language learning.

Bachelor's qualification work in the specialty: 126 "Information systems and technologies", educational program: "Information systems and technologies". - Kyiv National University of Construction and Architecture. - Kyiv, 2026.

The work is devoted to provide the opportunity to master a foreign language by applying a system for video podcasting and further testing (control of acquired knowledge and correction of further training plan).

As a result, a part of web application was developed, user groups were created, users were created, and a knowledge testing system was provided.

Keywords: web application, information support, distance learning, knowledge control, podcasts, sub-podcasts, tests.

ЗМІСТ

Вступ 8

Перелік умовних позначень 10

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 11

1.1 Опис предметної області 11

1.2 Аналіз задачі 13

1.3 Постановка задачі 18

1.4 Аналіз варіантів реалізації системи 19

1.5 Вибір технології реалізації 20

1.5.1 Забезпечення бізнес-логіки 20

1.5.2 Забезпечення рівня робити з даними 24

1.5.3. Забезпечення користувацького інтерфейсу 26

2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ СИСТЕМИ. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 29

2.1 Діаграма компонентів 29

2.2 Діаграми варіантів використання 33

2.2.1 Діаграма варіантів використання для гостя 35

2.2.2 Діаграма варіантів використання для викладача 36

2.2.3 Діаграма варіантів використання для студента 37

2.3 Інфологічна модель бази даних 38

2.4 Інформаційне забезпечення 40

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТА ТЕСТОВИЙ ПРИКЛАД ПРОГРАМИ 44

3.1 Архітектурні рівні системи 44

3.2 Інтерфейс комплексу 51

4. ЕРГОНОНОМІКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ 55

4.1 Розрахунок часу евакуації людей при пожежі в приміщенні..... 55

4.2 Ергономічні вимоги до організації робочих місць з комп`ютерною технікою 63

ВИСНОВКИ..... 67

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 68

Вступ

Подкастинг - процес створення і поширення звукових або відео-передач у мережі інтернет. Спочатку цей жанр мав на увазі тільки аудіозаписи в RSS-потоці, подкасти нагадували "звукові живі журнали". Але з часом термін "подкаст" розширився і знайшов три вистави: аудіокаст, відеокаст і скрінкасти, притому скрінкастинг можна вважати частиною відеоподкастингу, оскільки інформація передається за допомогою відео- та аудіо-потоку, з тією лише різницею, що зображення записано не на відеокамеру, а безпосередньо з екрану комп'ютера. [1]

В освіту подкастинг прийшов не відразу. В мережі Інтернет почали з'являтися відеозаписи виступів відомих лекторів європейських і американських університетів. Деякі університети почали викладати цикли таких відеолекцій у відкритий доступ, де кожен користувач мережі Інтернет у разі наявності відповідної швидкості доступу і знання англійської мови міг навчатися дистанційно у кращих лекторів світу. Паралельно з цим розвивався жанр скрінкастинга, де користувачі мережі Інтернет записували для інших користувачів посібники на різні теми, наприклад "Основи використання поштового клієнта Mozilla Thunderbird" і ін. Всі ці речі не могли не стати в нагоді як в дистанційному, так і очному освітньому процесі, і в даний час відеокасти набувають все більшої популярності.

Всі ці процеси можна простежити, якщо подивитися на провідні університети світу. Багато з них мають або власні майданчики для розміщення відео- та аудіолекцій, інші ж користуються можливостями всесвітньо відомих місць для розміщення відеофайлів, наприклад YouTube.EDU, Vimeo і іншими. Безсумнівно, однією з головних проблем при роботі такого розділу виступає установка прав доступу, оскільки багато матеріалів курсів і методики є унікальними. Також досить великою проблемою є категоризація курсів, яка на відеохостингу організована тільки у вигляді міток і плейлистів. Взаємодія з відеохостингу можливо тільки через надане ними API (якщо воно існує), що само по собі накладає обмеження на авторів курсів.

Саме через ці причини багато університетів або шкіл світу використовують власні майданчики для відеокурсів. У той же час багато університетів використовують майданчики для розміщення відеофайлів для зберігання і конвертації в популярні формати своїх курсів. Серед найвідоміших відео- і аудіолекцій - матеріали Массачусетського технологічного інституту, Гарварду, Стенфорду, Кембриджу, Берклі, Єльського університету, Оксфорда.

Серед українських університетів даний напрямок в освіті тільки починає розвиватися. У школі SpeakNow напрямок онлайн освіти має зародковий вигляд. У той же час наявність даного розділу тільки популяризувала б школу як місце для зручного здобуття освіти.

Перелік умовних позначень

БД – База (або бази) даних

ІТ – Інформаційні технології

СКБД – Система керування БД

API – Application Programming Interface – набір готових класів, функцій, структур та констант, що надаються додатком для його використання у зовнішніх програмних продуктах

CIL – Common Intermediate Language

CLR – Common Language Runtime

CRUD – Create Read Update Delete

DI – Dependency Injection

DTO – Data Transfer Object

EF CORE – Entity Framework Core

HTTP – HyperText Transfer Protocol

JS – JavaScript

MSSS – MS SQL SERVER – Microsoft SQL Server

T-SQL – Transact SQL – Transact Structured Query Language

VS – Microsoft Visual Studio

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Проблема, яку потрібно вирішити (з позиції системного аналізу): попередня система навчання студентів іноземним мовам, що була впроваджена на підприємстві морально і технічно застаріла. Вона погано адаптується до сучасних умов ведення бізнесу, їй бракує новітніх систем навчання, що призводить до досить складного алгоритму введення нових функцій та іноді унеможливлює швидку і адекватну співпрацю з іншими структурами. Внаслідок використання застарілих технологій, страждає якість послуг, які підприємство надає клієнтам.

Актуальність задачі: у сучасному світі погано розроблена система навчання іноземним мовам може призвести до значних фінансових витрат. Як відомо успішність будь-якого проекту оцінюється у тому наскільки швидко цей проект окупається і які прибутки ця задумка приносить. На всі ці характеристики впливає гнучкість програмного забезпечення і машинно-технічного забезпечення, що дозволяє швидко задовольняти потреби клієнтів і отримувати вищі прибутки порівняно з конкурентами. Якщо розглядати конкретну систему, яка буде спроектована у цій курсовій роботі саме візуальна подача матеріалу, яка найбільш впливає на сприйняття інформації користувачем, повинна бути розглянута до найменших деталей і максимально оптимізована. Тому найбільш важливим аспектом для сучасної організацій є простий і логічно організований додаток з легкою системою навігації та обов’язковим пунктом є його максимальна оптимізація для забезпечення найшвидшої взаємодії між користувачем і системою вивчення іноземних мов.

Для того, щоб провести конкретну постановку задачі і визначити цілі, які мають бути досягнені створенням певного програмного продукту, слід описати **предметну область**.

Предметна область – площина в якій виникла певна проблема, яку можна розв’язати створивши певний програмний продукт або алгоритм опрацювання проблеми задля покращення його ефективності.

Впровадження інформаційних технологій в освітній процес дозволяє значно полегшити організацію безлічі навчальних процесів, збір інформації про студентів, викладачів і другом персоналі, систематизацію та аналіз результатів навчання. Також впровадження ІТ дозволяє реалізовувати багато нових форм, наприклад дистанційне навчання. При цьому саме поняття дистанційного навчання містить в собі різні методи, наприклад навчання за допомогою віддаленого тестування або перегляду потокових скрінкастів.

У зв'язку з цим виникла необхідність створення середовища для організації підтримки і управління дистанційним і очним навчальним процесом через навчання за допомогою переглядів відеокастів. Організовувати весь навчальний процес - це окреме завдання, тому за основне завдання була прийнята розробка веб-додатку для інформаційного забезпечення знаннями учнів через перегляди відеокастів. Дана система повинна дозволяти студентам проходити навчання з будь-якої точки світу, де є доступ в мережу Інтернет. Викладачі та автор повинен мати можливість завантажувати відеолекції і призначати групи студентів до їх перегляду. Також повинен бути реалізований модуль, який відповідає за тестування отриманих учнями знань.

В даній роботі розглянута проблема системи навчання у школі «SpeakNow». Отже, предметна область даного проекту це організація ефективного навчального процесу у школі, надання користувачам додатку ясного і чіткого знання про теоретичні та практичні навички володіння іноземною мовою; сформувати навички самостійної, творчої роботи; вміння організовувати своє навчання, розвинути здібності розмовляти іноземною мовою; тестування користувачів школи SpeakNow, виявлення їх сильних та слабких сторін у вивченому матеріалі, корегування навчального плану.

1.2 Аналіз задачі

Дерево цілей - це графічне зображення взаємозв'язку і підпорядкованості цілей, що відображає розподіл місії і мети на цілі, підцілі, завдання та окремі дії. Дерево цілей можна визначити, як цільовий каркас організації, явища чи діяльності. [2]

Дерево цілей з кількісними показниками, що використовуються в якості одного із засобів при прийнятті рішень, і носить назву дерева рішень. Головна перевага дерева рішень перед іншими методами - можливість пов'язати ставлення цілі з діями, що підлягають реалізації в сьогоденні. Основна ідея щодо побудови дерева цілей - декомпозиція.

Розглянемо технологічні засади побудови дерева цілей. Не існує універсальних методів побудови дерева цілей. Способи його побудови залежать від характеру цілі, обраного методологічного підходу, а також від того, хто розробляє дерево цілей, як він представляє собі поставлені перед ним завдання, як він бачить їхній взаємозв'язок.

На рис. 1 представлено ієрархічне дерево цілей підсистеми додатку.

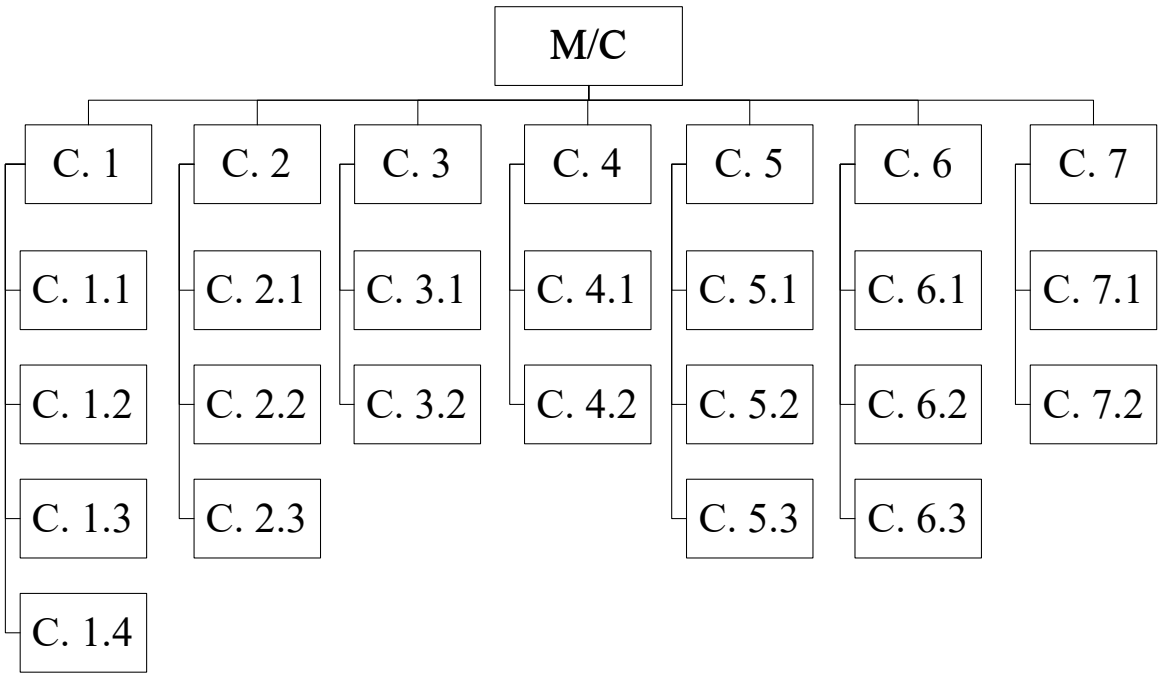

Рисунок 1. Дерево цілей підсистеми

М/С. – розробка додатку для організації «SpeakNow».

С.1 Сформувати ролі у web-додатку.

С.1.1 створити загальну базу даних для всіх користувачів.

С.1.2 створити критерії пошуку.

С.1.3 сформувати зручну функцію вибору.

С.1.4 забезпечення інтерфейсу на вибір.

С.2 Створити зручну функцію обговорення.

С.2.1 забезпечення переписки за згодою з обох сторін.

С.2.2 сформування меню налаштувань переписки.

С.2.3 забезпечення можливостей робити дописи.

С.3 Сформувати спец можливості.

С.3.1 забезпечення категорій.

С.3.2 забезпечення функціонального виводу.

С.4 Створити перевірку.

С.4.1 забезпечення виправлення повідомлень.

С.4.2 забезпечення музикального супроводження/зручного інтерфейсу.

С.5 Сформувати статистику.

С.5.1 забезпечити тестування.

С.5.2 забезпечити тестування на час.

С.5.3 забезпечити статистики тестувань.

С.6 Сформувати навчальний процес у вигляді відеокастів.

С.6.1 забезпечення зручного перегляду.

С.6.2 забезпечення тесту на рівень знань в кінці.

С.6.3 забезпечення нагадувань про заняття.

С.7 Створити граматичний тренажер.

С.1 забезпечення тесту на рівень граматики.

С.2 сформування тренажеру для покращення навичок.

На рис. 2 представлене ієрархічне дерево задач, що повинна виконувати розроблювана система.

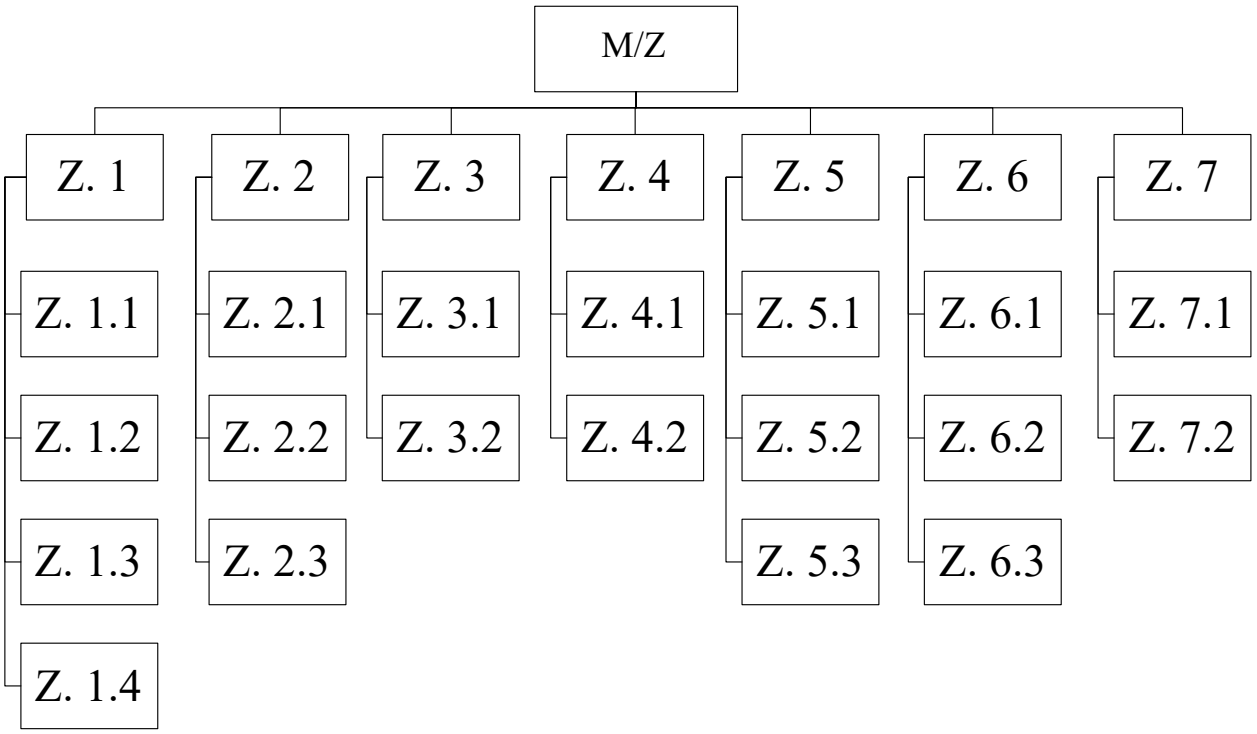

Рисунок 2. Дерево задач підсистеми додатку для організації «SpeakNow»

M/Z. – розробка додатку для організації «SpeakNow».

Z.1 Створення ролей у web-додатку.

Z.1.1 сформування загальної бази даних для всіх користувачів.

Z.1.2 сформування критеріїв пошуку.

Z.1.3 сформування зручної функції вибору.

Z.1.4 вибір інтерфейсу.

Z.2 Створення зручної функції обговорення.

Z.2.1 забезпечення переписки за згодою з обох сторін.

Z.2.2 сформування меню налаштувань переписки.

Z.2.3 забезпечення можливостей робити дописи.

Z.3 Сформування спец можливості.

Z.3.1 класифікація за категоріями.

Z.3.2 забезпечення функціонального виводу.

Z.4 Створення перевірки.

Z.4.1 забезпечення виправлення повідомлень.

Z.4.2 забезпечення музикального супроводження/зручного інтерфейсу.

Z.5 Сформування статистики.

Z.5.1 розробка тестування.

Z.5.2 розробка тестування на час.

Z.5.3 розробка статистики тестувань.

Z.6 Сформування навчального процесу у вигляді відеокастів.

Z.6.1 забезпечення зручного перегляду.

Z.6.2 забезпечення тесту на рівень знань в кінці.

Z.6.3 забезпечення нагадувань про заняття.

Z.7 Створення граматичного тренажеру.

Z.7.1 створення тесту на рівень граматики.

Z.7.2 сформування тренажеру для покращення навичок.

На рис. 3 представлене ієрархічне дерево функцій, що повинна виконувати розроблювана система.

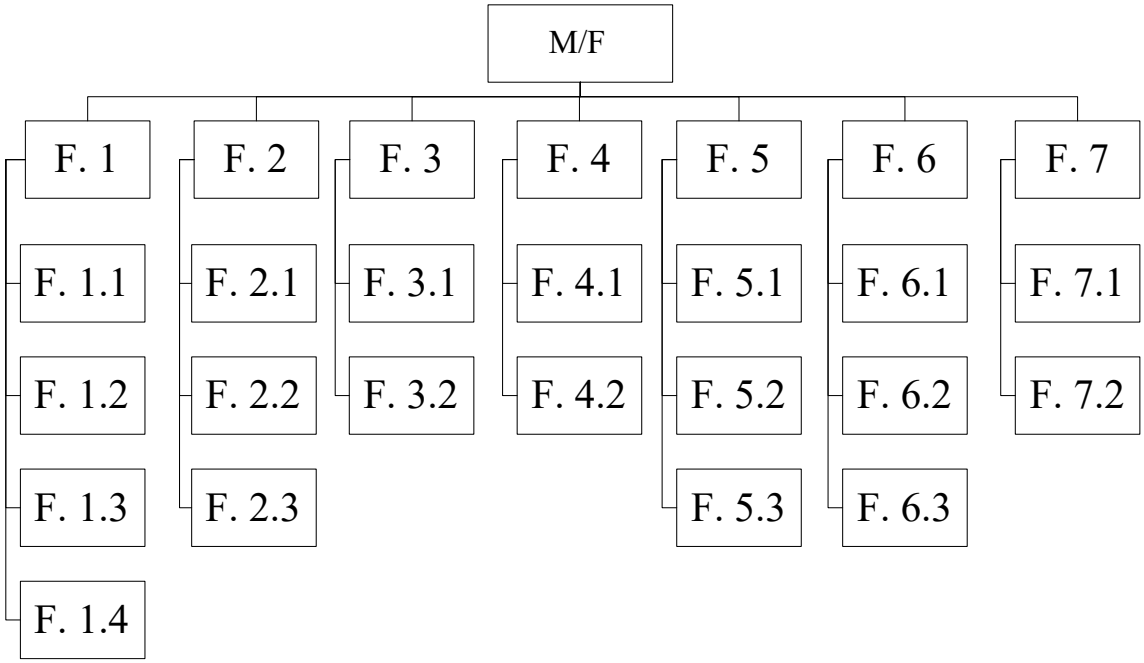

Рисунок 3. Дерево функцій підсистеми додатку для організації «SpeakNow»

M/F. – розробка додатку для організації «SpeakNow».

F.1 Розробка ролей у web-додатку.

F.1.1 розробка загальної бази даних для всіх користувачів.

F.1.2 розробка критеріїв пошуку.

F.1.3 розробка зручної функції вибору.

F.1.4 розробка інтерфейсу на вибір.

F.2 Розробка зручної функції обговорення.

F.2.1 розробка переписки за згодою з обох сторін.

F.2.2 розробка меню налаштувань переписки.

F.2.3 розробка можливостей робити дописи.

F.3 Розробка спец можливостей.

F.3.1 розробка класифікації за категоріями.

F.3.2 розробка функціонального виводу.

F.4 Розробка перевірки.

F.4.1 розробка виправлення повідомлень співрозмовника.

F.4.2 розробка музикального супроводження/зручного інтерфейсу.

F.5 Розроблення статистики.

F.5.1 розробка тестування.

F.5.2 розробка тестування на час.

F.5.3 розробка статистики тестувань.

F.6 Розробка навчального процесу у вигляді відеокастів.

F.6.1 розробка зручного перегляду.

F.6.2 розробка тесту на рівень знань в кінці.

F.6.3 розробка нагадувань про заняття.

F.7 Розробка граматичного тренажеру.

F.7.1 розробка тесту на рівень граматики.

F.7.2 розробка тренажеру для покращення навичок.

1.3 Постановка задачі

Додаток створюється для:

- підвищення свого розмовного рівня;
- допомоги у підвищення рівня співрозмовнику(обговорення);
- ведення статистики свого навчання;
- збагачення новими словами;
- постійних занять у додатку;
- покращення рівня граматики;
- підвищення кваліфікації викладачів;
- аналізу їх ефективності;
- аналізу сильних та слабких сторін окремого студента;

Вимоги до додатку:

- додаток повинен бути зручним та простим у використанні;
- додаток повинен правильно функціонувати, тобто виконувати ті функції, що передбачено при розробці;
- додаток повинен допомагати працівникам організації правильно керувати роботою студентів та підвищити кваліфікацію працівників;
- додаток повинен допомагати працівникам організації краще орієнтуватися у сильних та слабких сторонах своїх студентів;
- додаток повинен допомагати працівникам організації правильно моделювати навчальний процес;

1.4 Аналіз варіантів реалізації системи

Передбачається, що розробляється система повинна бути на багато користувачів і в ній буде передбачено поділ користувачів за ролями. Таких ролей має бути чотири: «адміністратор», «викладач (тьютор)», «студент», «гість».

Викладач повинен мати можливість завантажувати в систему змонтовані відеокасти. Викладач повинен мати можливість призначати групи студентів до перегляду певних відеокасти. Також викладач повинен мати можливість відповідати на запитання студентів, схвалювати стоять і видаляти непотрібні запитання.

Адміністратор повинен мати всі привілеї викладачів. Також адміністратор повинен мати можливість модерувати обговорення відеокастів і самі відеокасти.

Гість повинен не мати права доступу до завантажених в систему відеокастів.

Так як розроблювана система призначена для забезпечення інформаційної підтримки, необхідно реалізувати її з урахуванням максимальної доступності звернення до неї.

При такій постановці завдання, найбільш вигідним варіантом реалізації системи є створення веб-додатку або модуля існуючої системи. В цьому випадку з метою зручності наповнення та управління інформацією має здійснюватися через веб-інтерфейс. Також необхідно прагнути до досягнення максимальної автоматизації операцій в системі, так як це економить час доступу до інформації і спрощує роботу користувачів.

Даний підхід найбільш раціональний зважаючи на забезпечення, таким чином, можливості не закріплюватись на одному робочому місці.

Звернутися до системи, а також керувати нею при наявності відповідних прав можна з будь-якого приміщення, в якому є комп'ютер при наявності доступу в мережу Інтернет. Це дозволить викладачам і адміністратору системи бути менш скутими часом перебування на робочому місці, так як вони зможуть завантажити відеокаст, призначити студентів до перегляду відео та відповісти на всі питання студентів з будь-якого комп'ютерного класу або поза стінами школи, а також провести тестування набутих знань.

1.5 Вибір технології реалізації

1.5.1 Забезпечення бізнес-логіки

На сьогоднішній момент мова програмування C # один з найпотужніших, що швидко розвиваються і затребуваних мов в IT-галузі. На даний момент на ній пишуться найрізноманітніші програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів. [3]

У порівнянні з іншими мовами C # досить молодий, але в той же час він вже пройшов великий шлях. Перша версія мови вийшла разом з релізом Microsoft Visual Studio .NET в лютому 2002 року. Поточною версією мови є версія C # 8.0, яка вийшла у вересні 2019 року разом з релізом .NET Core 3.

C # є мовою з Сі-подібним синтаксисом і близький в цьому відношенні до C++ і Java.

C # є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C++. Наприклад, C # підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. І C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональностей, як, наприклад, лямбда, динамічне зв'язування, асинхронні методи і т.д. [4]

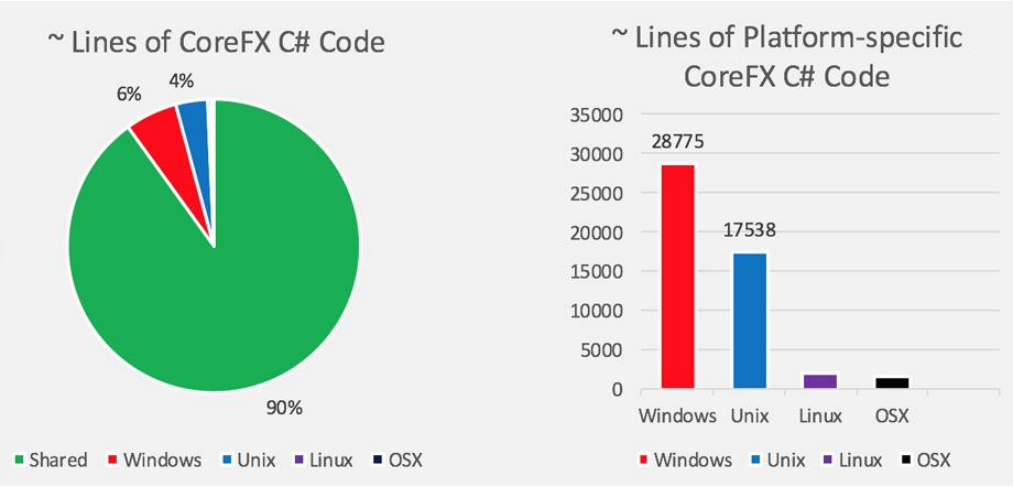

Рисунок 4. Співвідношення загального коду і коду специфічного для кожного окремо-взятого сімейства ОС

Роль платформи .NET

Коли говорять C#, нерідко мають на увазі технології платформи .NET (WindowsForms, WPF, ASP.NET, Xamarin). І, навпаки, коли говорять .NET, нерідко мають на увазі C#. Однак, хоча ці поняття пов'язані, ототожнювати їх невірно. Мова C # був створений спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше.

Якось Білл Гейтс сказав, що платформа .NET - це найкраще, що створила компанія Microsoft. Можливо, він мав рацію. Фреймворк .NET представляє потужну платформу для створення додатків. Можна виділити наступні її основні риси:

- **Підтримка декількох мов**. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C # це також VB.NET, C ++, F #, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi. NET. При компіляції код на будь-якому з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET. Тому ми можемо зробити окремі модулі однієї програми на окремих мовах.
- **Кросплатформність**. .NET є платформою яку можна застосувати на більшості платформ (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент .NET Core підтримується на більшості сучасних ОС

Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти програми на мові C# для самих різних платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

- **Потужна бібліотека класів**. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І яке б додаток ми не збиралися писати на C # - текстовий редактор, чат або складний веб-сайт - так чи інакше ми задіємо бібліотеку класів .NET.
- **Різноманітність технологій**. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом - технологія WPF і UWP, для створення більш простих графічних додатків - Windows Forms. Для розробки мобільних додатків - Xamarin. Для створення веб-сайтів - ASP.NET і т.д.

Також ще слід відзначити таку особливість мови C # і фреймворка .NET, як автоматичне прибирання сміття. А це означає, що нам в більшості випадків не доведеться, на відміну від C++, піклуватися про звільнення пам'яті. Вищезазначена загальномовного середовища CLR сама викликає збирач сміття і очистить пам'ять.

.NET Framework і .NET Core

Варто відзначити, що .NET довгий час розвивався головним чином як платформа для Windows під назвою .NET Framework. В 2019 вийшла остання версія цієї платформи - .NET Framework 4.8. Вона більше не розвивається.

З 2014 Microsoft став розвивати альтернативну платформу - .NET Core, яка вже призначалася для різних платформ і повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Тому слід розрізняти .NET Framework, який призначений переважно для Windows, і кросплатформних .NET Core.

Керований і некерований код

Нерідко додаток, створене на C#, називають керованим кодом (managed code). Це означає, що для цієї програми створено на основі платформи .NET і тому керується загальномовна середовищем CLR, яка завантажує додаток і при необхідності очищає пам'ять. Але є також додатки, наприклад, створені на мові C ++, які компілюються не в спільну мову CIL, як C # або VB.NET, а в звичайний машинний код. В цьому випадку .NET не керує додатком.

J I T-компіляція

Як вище писалося, код на C# компілюється в додатку або складання з розширеннями exe або dll на мові CIL. Далі при запуску на виконання подібного програми відбувається J I T-компіляція (Just-In-Time) в машинний код, який потім виконується. При цьому, оскільки наш додаток може бути великим і містити купу інструкцій, в поточний момент часу компілюватиметься лише та частина програми, до якої безпосередньо йде звернення. Якщо ми звернемося до іншої частини коду, то вона буде скомпільована з CIL в машинний код. При тому вже скомпільована частина програми зберігається до завершення роботи програми. У підсумку це підвищує продуктивність.

1.5.2 Забезпечення рівня робити з даними

Бази даних будуть створюватись за допомогою технології **Code First** та **Entity Framework Core 3.0**.

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану, легку технологію, що розширюється, від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping - відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати з базами даних, але має більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами. [5]

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СКБД, якщо для неї є потрібний провайдер.

За замовчуванням на даний момент Microsoft надає ряд вбудованих провайдерів: для роботи з MS SQL Server, для SQLite, для PostgreSQL. Також є провайдери від сторонніх постачальників, наприклад, для MySQL.

Також варто відзначити, що EF Core надає універсальний API для роботи з даними. І якщо, наприклад, ми вирішимо змінити цільову СКБД, то основні зміни в проекті будуть стосуватися насамперед конфігурації і налаштування підключення до відповідних провайдерів. А код, який безпосередньо працює з даними, отримує дані, додає їх в БД і т.д., залишиться незмінним.

Entity Framework Core багато успадкував від своїх попередників, зокрема, Entity Framework 6. У той же час треба розуміти, що EF Core - це не нова версія по відношенню до EF 6, а зовсім інша технологія, хоча в цілому принципи роботи у них будуть збігатися. Тому в рамках EF Core використовується своя система версій. Поточна версія - 3.0 була випущена у вересні 2019 року. І технологія продовжує розвиватися.

Як технологія доступу до даних Entity Framework Core може використовуватися на різних платформах стека .NET. Це і стандартні платформи типу Windows Forms, консольні додатки, WPF, UWP і ASP.NET Core. При цьому кросплатформна природа EF Core дозволяє задіяти її не тільки на ОС Windows, але і на Linux і Mac OS X.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік. Властивості необов'язково представляють прості дані типу int або string, але можуть також представляти і більш комплексні типи даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому суті можуть бути пов'язані асоціативної зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконання запиту транслює вирази LINQ в вирази, зрозумілі для конкретної СКБД (як правило, в вирази SQL).

1.5.3. Забезпечення користувацького інтерфейсу

Всю частину зв’язану з користувацьким інтерфейсом у даній роботі буде покладено на **Angular**.

Angular дозволяє вам з "коробки" створювати великі і складні за частиною бізнес-логіки додатка. Angular було повним переосмисленням AngularJS, сам фреймворк став куди чистіше і гнучкіше, більш enterprise-подібним і з цієї точки зору має високу масштабованість. [6]

Які плюси можна виділити:

- Підтримка Google, Microsoft;
- Інструменти розробника (CLI);
- Єдина структура проекту;
- TypeScript з "коробки" (ви можете писати строго типізований код);
- React програмування з RxJS;
- Єдиний фреймворк з Dependency Injection з "коробки";
- Шаблони, засновані на розширенні HTML;
- Кросбраузерна Shadow DOM з коробки (або його емуляція);
- Кросбраузерна підтримка HTTP, WebSockets, Service Workers;
- Не потрібно нічого додатково налаштовувати. Більше ніяких обгорток;
- Більш сучасний фреймворк, ніж AngularJS (на рівні React, Vue);
- Велике ком'юніті.

Варто виділити також і мінуси:

- Вище поріг входження через Observable (RxJS) і Dependency Injeciton;
- Щоб все працювало добре і швидко потрібно витрачати час на додаткові оптимізації (він не супер швидкий, за замовчуванням, але швидше AngularJS у багато разів і з кожною новою версією стає все швидше);
- Angular-Univesal має багато підводних каменів;
- Динамічне створення компонентів виявляється нетривіальною завданням.

Все, що потрібно дізнатися в Angular для розробки продуктивних і швидко-працюючих додатків будь-якого рівня складності, описано в нижче наступних концепціях:

- **Form Builder** - для розробки насправді складних форм, слід знати React форми, точніше принципово забути про декларативні форми. Ось один з хороших прикладів (React форма + валідація);
- **Change Detection** - так як Angular за замовчуванням використовує двостороннє зв'язування моделі даних, то при роботі з великим об'ємом таких даних ваші програми будуть працювати повільніше, тому в деяких випадках варто подбати про правильну стратегію виявлення змін.
- **Templating** - синтаксис шаблонів з точки зору абстракції змінився не дуже сильно в порівнянні з AngularJS, тобто так само можливо написати умови, цикли, зв'язати модель даних та інше. Слід добре розуміти і розібиратися в Angular шаблонах - що таке структурні і декларативні директиви, а також що таке Input-параметри і Output-події.
- **Routing** - напевно, це одне з основних явищ у розробці веб-додатків. Тут просто важливо розуміти, що маршрутизація так само, як і компоненти, має свій життєвий цикл. Ще варто відзначити: якщо на якійсь із маршрутів повішено модуль, а не компонент, який відповідає за відображення сторінки по цьому маршруту, то модуль буде довантажувати на сторінці на вимогу.
- **Annotations** - декоратори, які використовуються в достатку при написанні додатків на Angular, не є якоюсь жорсткою магією TypeScript. Декоратори є специфікацією EcmaScript і коли браузери почнуть підтримувати їх, вони будуть нативної виконуватися в browser runtime. Насправді, декоратори вельми корисні і забезпечують досить високою читаністю написаний код. Один із прикладів - це валідація моделей даних з використанням декораторів або десеріалізацією / серіалізацією даних.
- **Observables** - тут варто відзначити тільки те, що незабаром Observables будуть специфікацією EcmaScript і все це буде підтримуватися в браузерах. З точки зору теорії, якщо розкрити поняття Observer (спостерігач) - це поведінковий

шаблон проектування. Також відомий як «підлеглі» (Dependents). Створює механізм у класу, який дозволяє отримувати примірника об'єкта цього класу оповіщення від інших об'єктів про зміну їх стану, тим самим спостерігаючи за ними.

- **Shadow DOM** - це засіб для створення окремого DOM-дерева всередині елемента, яке не видно зовні без застосування спеціальних методів, є специфікацією W3C. Це зручний спосіб створення ізольованих веб-компонентів, які можна використати знову. Чисто технічно, якби браузери вже сьогодні підтримували багато концепцій, які використовує Angular, стали би непотрібними транспайлери і інші системи збирання.

2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ СИСТЕМИ. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Діаграма компонентів

Діаграма компонентів описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код. У багатьох середовищах розробки модуль або компонент відповідає файлу.

За допомогою діаграми компонентів представляються інкапсульовані класи разом з їх інтерфейсними оболонками, портами і внутрішніми структурами (які теж можуть складатися з компонентів і конекторів).

Компоненти зв'язуються через залежності (пунктирні стрілки), коли з'єднується необхідний інтерфейс одного компонента з наявними інтерфейсом іншого компонента. Таким чином ілюструються відносини клієнт-джерело між двома компонентами. [7]

Залежність показує, що один компонент надає сервіс, необхідний іншому компоненту. Залежність зображується стрілкою від інтерфейсу або порту клієнта до імпортованого інтерфейсу.

Коли діаграма компонентів використовується, щоб показати внутрішню структуру компонентів, необхідний інтерфейс і інтерфейс, що надається, складеного компонента можуть делегуватися до відповідних інтерфейсів внутрішніх компонентів.

Делегація показує зв'язок зовнішнього контракту компонента з внутрішньої реалізацією цієї поведінки внутрішніми компонентами.

Діаграма компонентів розробляється для наступних цілей:

- Візуалізації загальної структури вихідного коду програмної системи.
- Специфікації виконавчого варіанта програмної системи.
- Забезпечення багаторазового використання окремих фрагментів програмного коду.

- Уявлення концептуальної і фізичної схем баз даних.

У розробці діаграм компонентів беруть участь як системні аналітики та архітектори, так і програмісти. Діаграма компонентів забезпечує узгоджений перехід від логічного представлення до конкретної реалізації проекту у формі програмного коду.

Одні компоненти можуть існувати тільки на етапі компіляції програмного коду, інші - на етапі його виконання. Діаграма компонентів відображає загальні залежності між компонентами, розглядаючи останні як класифікатори.

Для уявлення фізичних сутностей в мові UML застосовується спеціальний термін - компонент (component). Компонент реалізує деякий набір інтерфейсів і служить для загального позначення елементів фізичного представлення моделі. [8]

Для графічного представлення компонента може використовуватися спеціальний символ - прямокутник зі вставленими зліва двома дрібнішими прямокутниками. Усередині прямокутника записується ім'я компонента і, можливо, деяка додаткова інформація. Зображення цього символу може незначно варіюватися в залежності від характеру асоційованої з компонентом інформації.

Ім'я компонента підпорядковується загальним правилам іменування елементів моделі в мові UML і може складатися з будь-якого числа букв, цифр і деяких розділових знаків.

Окремий компонент може бути представлений на рівні типу або на рівні примірника. Хоча його графічне зображення в обох випадках однакове, правила запису імені компонента дещо відрізняються. Якщо компонент представляється на рівні типу, то в якості його імені записується тільки ім'я типу з великої літери.

Як простих імен прийнято використовувати імена виконуваних файлів (із зазначенням розширення exe після точки-роздільник), імена динамічних бібліотек (розширення dll), імена Web-сторінок (розширення html), імена текстових файлів (розширення txt або doc) або файлів довідки (hip), імена файлів баз даних (DB) або імена файлів з вихідними текстами програм (розширення h, cpp для мови C ++, розширення Java для мови Java), скрипти (pi, asp) і ін.

Оскільки конкретна реалізація логічного представлення моделі системи залежить від використовуваного програмного інструментарію, то і імена компонентів будуть визначатися особливостями синтаксису відповідної мови програмування.

В окремих випадках до простого імені компонента може бути додана інформація про ім'я пакета і про конкретну версії реалізації даного компонента. Необхідно зауважити, що в цьому випадку номер версії записується як позначене значення в фігурних дужках.

В інших випадках символ компонента може бути розділений на секції, щоб явно вказати імена реалізованих в ньому інтерфейсів.

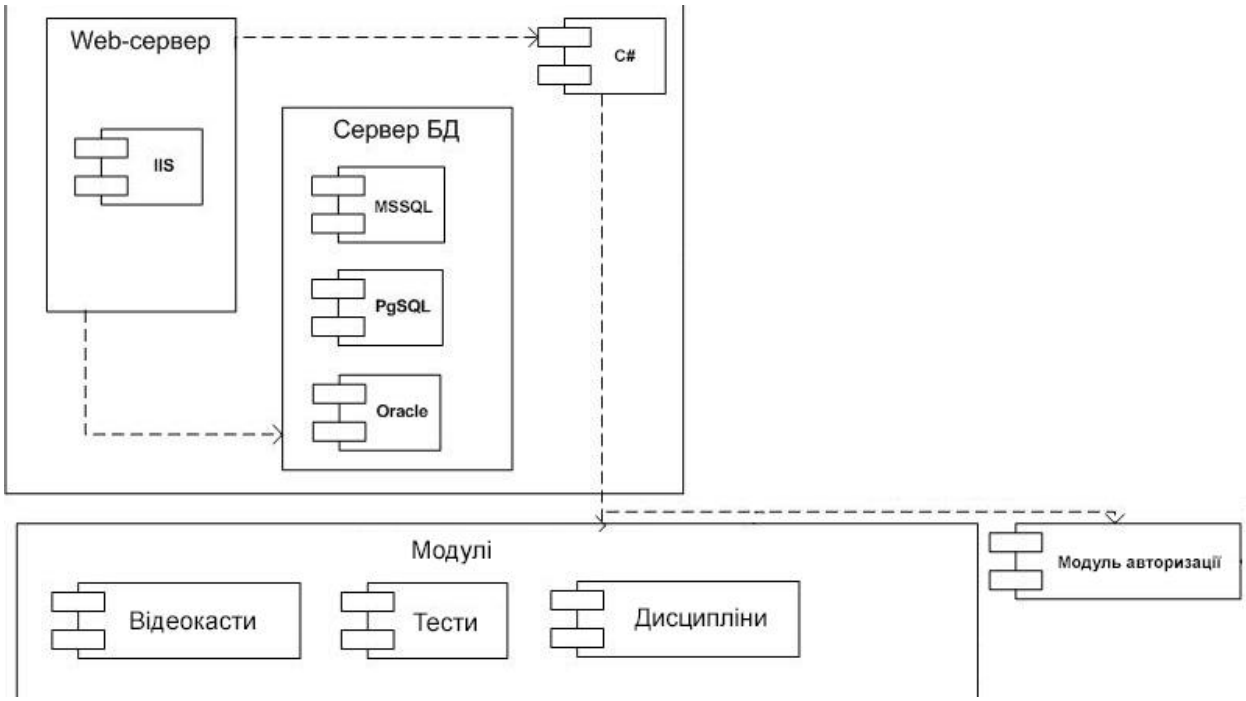

Рисунок 5. Діаграма компонентів

Діаграма компонентів (рис. 5) добре показує модулі відеокастів, тестів і дисциплін, що винесені як такі, що збільшують функціонал системи. У той же час система не залежить від них, і самі ці модулі самодостатні.

Модуль авторизації винесено в окрему частину, оскільки він не реалізує додатковий функціонал системи, а є однією з її частин.

Розроблюватись буде модуль тестування та незалежна його частина – авторизація.

Серверною розробленою частиною буде MSSQL, але як було зазначено розділом вище - EF Core надає можливість працювати з усіма можливими варіантами БД, не змінюючи код.

2.2 Діаграми варіантів використання

Діаграма варіантів використання (сценаріїв поведінки, прецедентів) є вихідним концептуальним поданням системи в процесі її проектування і розробки. Дана діаграма складається з акторів, варіантів використання і відносин між ними. При побудові діаграми можуть використовуватися також загальні елементи нотації: примітки і механізми розширення. [9]

Суть даної діаграми складається в наступному: проектована система представляється у вигляді безлічі акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (дійовою особою) називається будь-який об'єкт, суб'єкт або система, що взаємодіє з моделюється системою ззовні. У свою чергу варіант використання - це специфікація сервісів (функцій), які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, що здійснюються системою при взаємодії з актором. При цьому в моделі ніяк не відбивається те, яким чином буде реалізовано цей набір дій.

Згідно UML актора графічно можна відобразити трьома способами: а) «дротяний чоловічок» б) клас з текстовим стереотипом «actor» в) довільна іконка. Перший спосіб відображення в вигляді «дротяного чоловічка» є найпоширенішим.

Варіант використання позначається на діаграмі еліпсом, усередині якого міститься його опис, що означає виконання будь-якої операції або дії. Варіант використання, що інстанціюється за запитом користувача, є закінченою послідовність дій. Це означає, що після того, як система закінчить обробку запиту актора, вона повинна повернутися в стан, в якому готова до виконання наступних запитів. Варіанти використання можуть включати в себе опис особливостей способів реалізації сервісу і різних виняткових ситуацій, таких як коректна обробка помилок системи.

Примітки призначені для включення в діаграму довільної текстової інформації, що має безпосереднє відношення до контексту розроблюваної системи. В якості такої інформації можуть бути коментарі розробника і обмеження. Графічно примітки відображаються прямокутником із загнутим верхнім правим

куточком, всередині якого міститься текст примітки. Лінія, що з'єднує примітка і елемент діаграми, називається якорем (фіксацією).

Зв'язки між акторами і варіантами відображаються з використанням відносин чотирьох видів:

- Асоціацій (може відображатися у вигляді односпрямованої або двобічної стрілки, яка б показала напрямок потоків інформації або сигналів)
- Узагальнення (служить для вказівки того факту, що деяка сутність А може бути узагальнена до суті В. В цьому випадку сутність А буде спеціалізацією суті В. На діаграмі даний вид відносини можна відображати тільки між однотипними сутностями, тобто між двома варіантами використання або двома акторами. Графічно дане відношення позначається суцільною лінією зі стрілкою, у вигляді незамальованого трикутника, від нащадка до батька)
- Включення (вказує, що деяка задана поведінка одного варіанту використання обов'язково включається в якості складового компонента в послідовність поведінки іншого варіанту використання. Відображається штрихованою стрілкою з вказанням стереотипу. Стрілка включення повинна бути спрямована від базового варіанту до такого, що включається і позначена стереотипом «include» або «uses»)
- Розширення (визначає потенційну можливість включення поведінки одного варіанту використання до складу іншого. Тобто дочірній варіант використання може як викликатися, так і не викликатися батьківським. Відображається штрихованою стрілкою з вказанням стереотипу. Стрілка розширення повинна бути спрямована від дочірнього варіанту до базового і позначена стереотипом «extend»)

2.2.1 Діаграма варіантів використання для гостя

Як видно з мети диплома, гості хоч і є другорядною цільовою аудиторією, оскільки основна цільова аудиторія - студенти школи SpeakNow, відповідно на даний момент функціонал доступних ззовні подкастів не реалізований, хоча і

може бути реалізований на вимогу. Відповідно, гостю показується лише привітальна сторінка і форма авторизації (рис. 6). Завантажені в систему відеокасти гостям на даний момент не показуються.

Рисунок 6. Діаграма варіантів використання для гостя

2.2.2 Діаграма варіантів використання для викладача

Викладач створює матеріали подкасту, для подальшого навчання студентів.

На рис. 7 зображено варіанти використання системи для викладача (автора матеріалів).

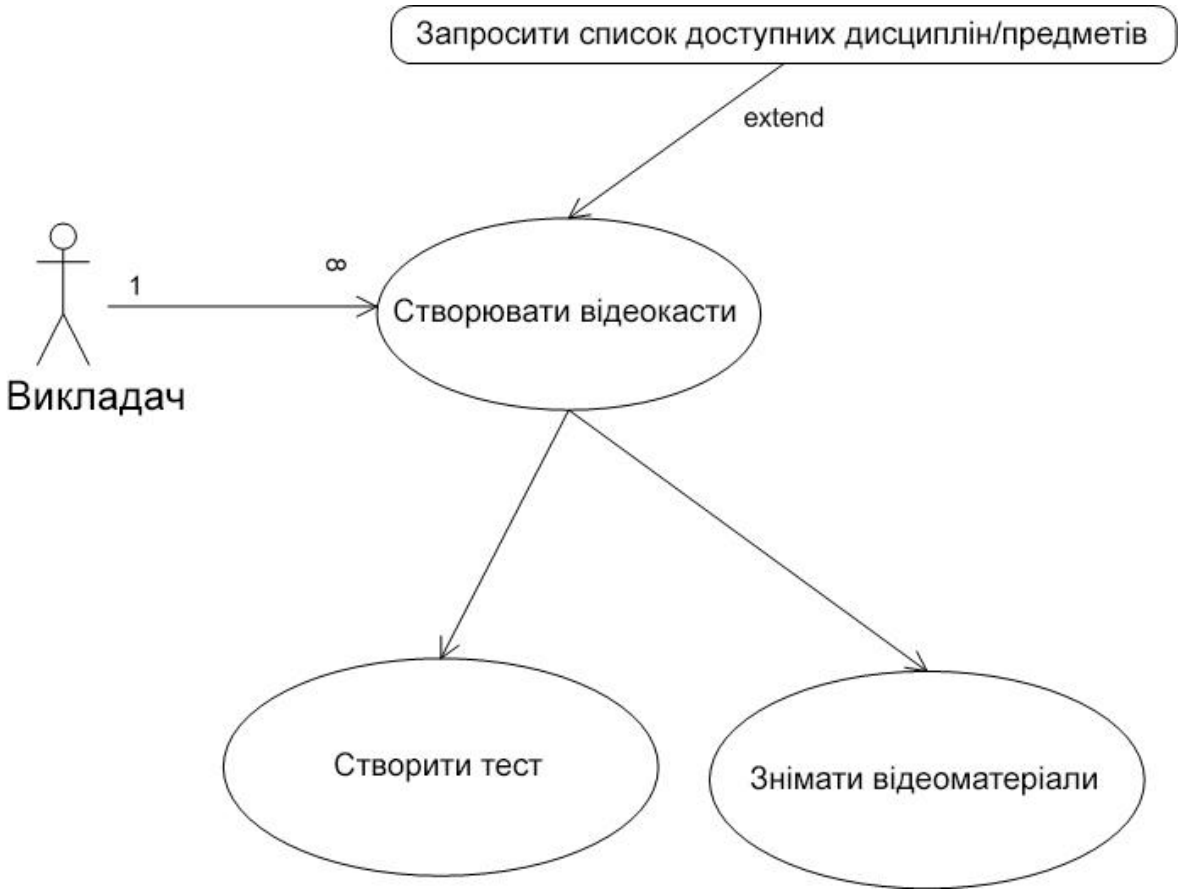

Рисунок 7. Діаграма варіантів використання для викладача

2.2.3 Діаграма варіантів використання для студента

Діаграма варіантів використання для студента - найбільша з усіх, оскільки саме вони навчаються в школі SpeakNow і на даний момент є цільовою аудиторією даного проекту. Студенти можуть переглядати доступні їм відеокасти (рис. 8), переглядати обговорення, запитання студентів і відповіді викладачів на ці питання в "Обговорення". Студенти можуть як переглядати відеокасти, завантажені авторами навчальних матеріалів, так і проходити тести.

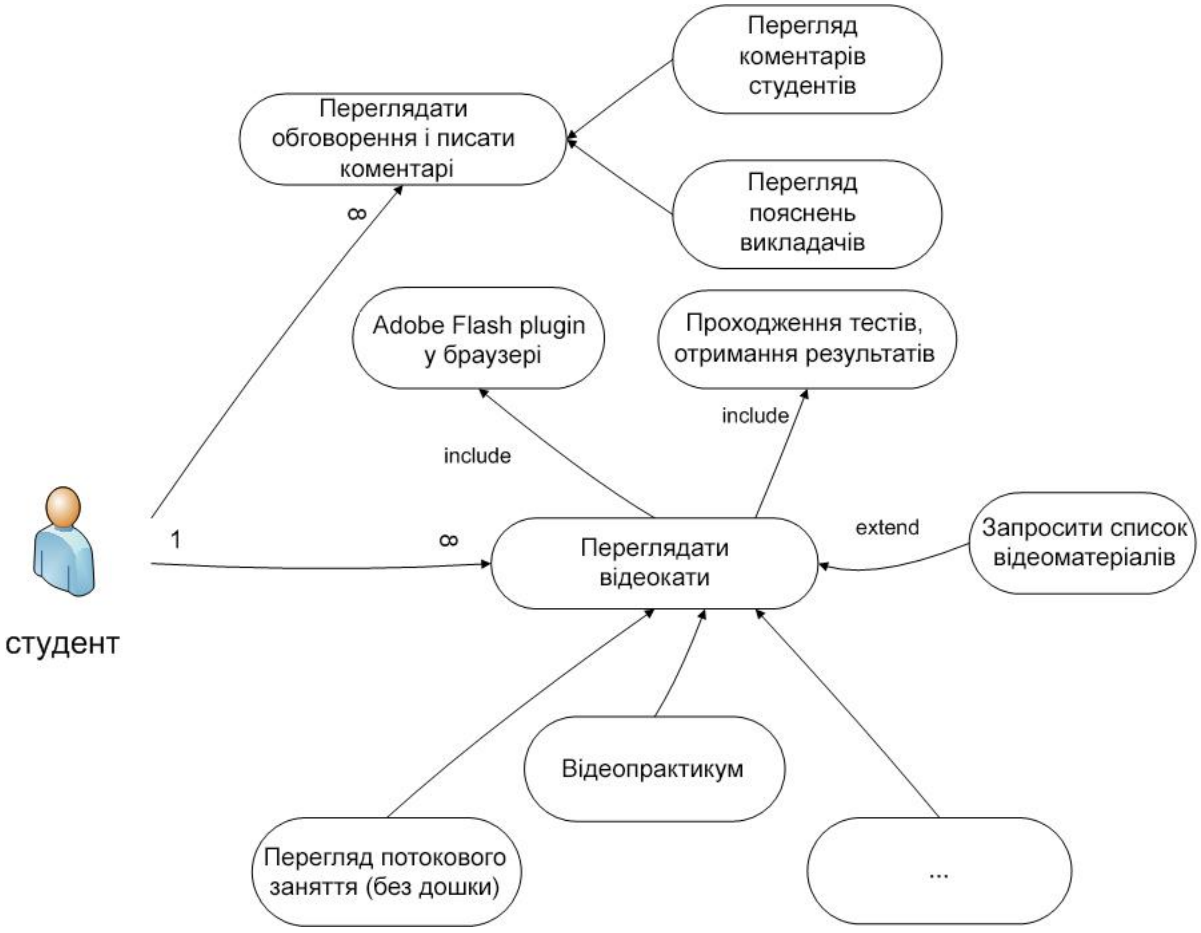

Рисунок 8. Діаграма варіантів використання для студента

2.3 Інфологічна модель бази даних

Інфологічна модель - це набір сутностей і зв'язків між ними. Вона містить інформацію про сутності системи і способи їх взаємодії, включає ідентифікацію об'єктів, важливих для предметної області (сутностей), властивостей цих об'єктів (атрибутів) і їх відносин з іншими об'єктами (зв'язків). У багатьох випадках інфологічна модель дуже складна і містить безліч об'єктів. Основні поняття інфологічної моделі представлені нижче:

- Сутність (клас однотипних об’єктів, інформація про котрі повинна бути врахована у моделі)
- Атрибут сутності (іменована характеристика, що є деякою властивістю сутності)
- Ключ сутності (ненадлишковий набір атрибутів, значення котрих у сумі є унікальними для кожного екземпляру сутності)
- Зв’язок (деяка асоціація між двома сутностями)

Мета інфологічного моделювання - забезпечення найбільш природних для людини способів збору і представлення тієї інформації, яку передбачається зберігати в створюваній базі даних. Тому інфологічна модель даних будується за аналогією з природною мовою. Основними конструктивними елементами інфологічних моделей є сутності, зв'язки між ними і їх властивості (атрибути). [10]

На рис. 10 зображена інфологічна модель бази даних системи. Інфологічна модель ядра системи зображена на рис. 9.

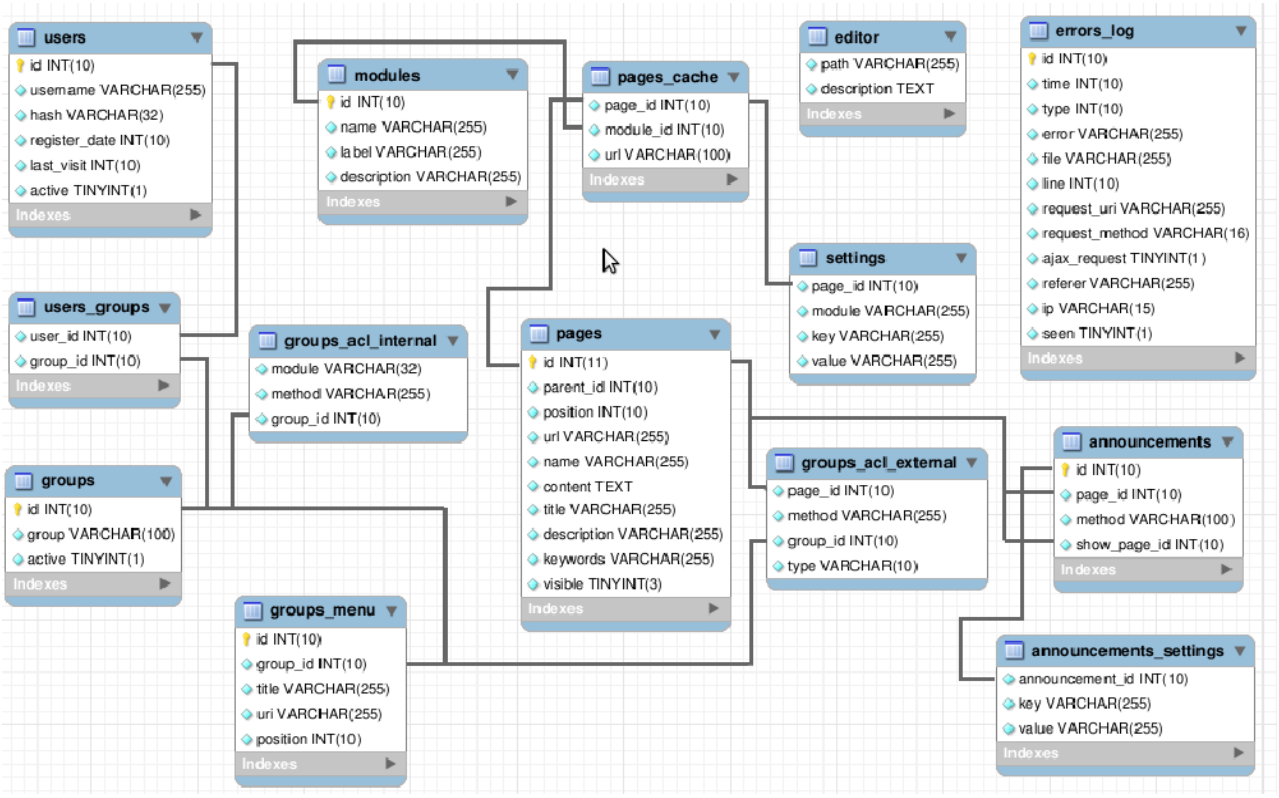

Рисунок 9. Інфологічна модель ядра системи

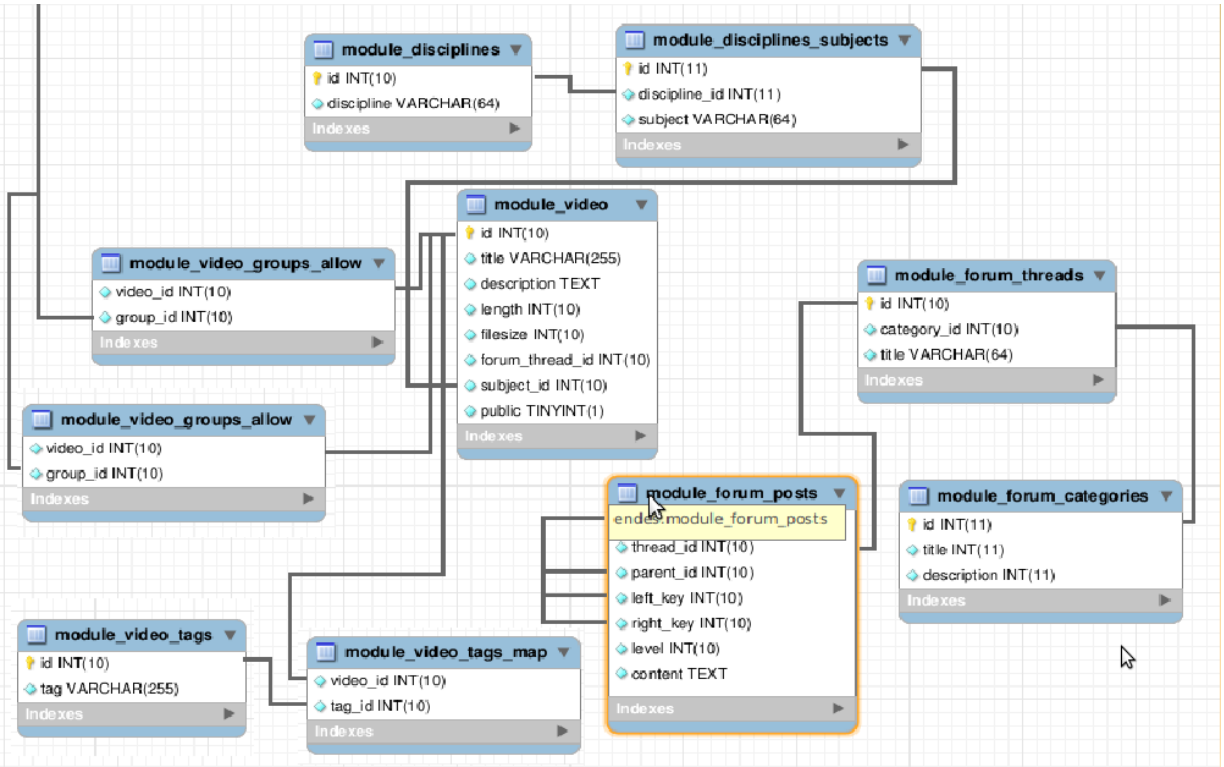

Рисунок 10. Інфологічна модель системи

2.4 Інформаційне забезпечення

SQL Server є однією з найбільш популярних систем управління базами даних (СКБД) в світі. Дана СКБД підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів.

SQL Server був створений компанією Microsoft. Перша версія вийшла в 1987 році. А поточною версією є версія 16, яка вийшла в 2016 році і яка буде використовуватися в поточному керівництві. [11]

SQL Server довгий час був винятково системою управління базами даних для Windows, проте починаючи з версії 16 ця система доступна і на Linux.

SQL Server характеризується такими особливостями як:

- Продуктивність. SQL Server працює дуже швидко.
- Надійність і безпека. SQL Server надає шифрування даних.
- Простота. З даної СКБД відносно легко працювати і вести адміністрування.

Центральним аспектом в MS SQL Server, як і в будь-який СУБД, є база даних. База даних являє сховище даних, організованих певним способом. Нерідко фізично база даних представляє файл на жорсткому диску, хоча така відповідність необов'язкова. Для зберігання і адміністрування баз даних застосовуються системи керування базами даних (database management system) або СКБД (DBMS). І якраз MS SQL Server є однією з такою СКБД.

Для організації баз даних MS SQL Server використовує реляційну модель. Ця модель баз даних була розроблена ще в 1970 році Едгаром Коддом. А на сьогоднішній день вона фактично є стандартом для організації баз даних.

Реляційна модель передбачає зберігання даних у вигляді таблиць, кожна з яких складається з рядків і стовпців. Кожен рядок зберігає окремий об'єкт, а в стовпчиках розміщуються атрибути цього об'єкта.

Для ідентифікації кожного рядка в рамках таблиці застосовується первинний ключ (primary key). В якості первинного ключа може виступати один або декілька стовпців. Використовуючи первинний ключ, ми можемо посилатися на певний

рядок в таблиці. Відповідно два рядки не можуть мати один і той же первинний ключ.

Через ключі одна таблиця може бути пов'язана з іншою, тобто між двома таблицями можуть бути організовані зв'язки. А сама таблиця може бути представлена у вигляді відносини ("relation").

Для взаємодії з базою даних застосовується мова SQL (Structured Query Language). Клієнт (наприклад, зовнішня програма) відправляє запит на мові SQL за допомогою спеціального API. СКБД належним чином інтерпретує і виконує запит, а потім посилає клієнту результат виконання.

Спочатку мова SQL була розроблена в компанії IBM для системи баз даних, яка називалася System / R. При цьому сама мова називався SEQUEL (Structured English Query Language). Хоча в підсумку ні база даних, ні сама мова не були згодом офіційно опубліковані, за традицією сам термін SQL нерідко вимовляють як "сиквел".

У 1979 році компанія Relational Software Inc. розробила першу систему керування баз даних, яка називалася Oracle і яка використовувала мову SQL. У зв'язку з успіхом даного продукту компанія була перейменована в Oracle.

Згодом стали з'являтися інші системи баз даних, які використовували SQL. У підсумку в 1989 році Американський Національний Інститут Стандартів (ANSI) кодифікував мову і опублікував його перший стандарт. Після цього стандарт періодично оновлювався і доповнювався. Останнє його оновлення відбулося в 2011 році. Але незважаючи на наявність стандарту нерідко виробники СКБД використовують свої власні реалізації мови SQL, які трохи відрізняються один від одного.

Виділяються два різновиди мови SQL: PL-SQL і T-SQL. PL-SQL використовується в таких СКБД як Oracle і MySQL. T-SQL (Transact-SQL) застосовується в SQL Server. Власне тому в рамках поточної роботи буде використовуватись саме T-SQL.

Залежно від завдання, яку виконує команда T-SQL, він може належати до одного з наступних типів:

DDL (Data Definition Language / Мова визначення даних). До цього типу належать різні команди, які створюють базу даних, таблиці, індекси, збережені процедури і т.д. У загальних рисах визначають дані.

DML (Data Manipulation Language / Мова маніпуляції даними). До цього типу відносять команди на вибір даних, їх оновлення, додавання, видалення - в загальному всі ті команди, за допомогою яких можна управляти даними.

DCL (Data Control Language / Мова управління доступу до даних). До цього типу відносять команди, які керують правами щодо доступу до даних.

Найвідомішими СКБД при роботі з сайтами є MSSQL і PostgreSQL. Також використовуються MySQL, Oracle, Firebird і деякі інші. Велика популярність MSSQL і PostgreSQL в порівнянні з пропрієтарними СКБД обумовлена великим спільнотою розробників, відкритістю продуктів і величезними можливостями по налаштуванню швидкодії баз даних.

Якщо порівнювати конкретно MSSQL і PostgreSQL, то можна виявити наступні переваги MSSQL:

- відповідність стандартам SQL - велику увагу розробники віддавали відповідності стандартам SQL. В MSSQL запити максимально відповідають стандартам SQL'99;
- більшу кількість платформ - MSSQL спочатку розроблявся як кросплатформна СКБД. У Windows MSSQL працює як служба, в *nix - як демон. PostgreSQL спочатку розроблявся як СКБД, що працює в *nix-системах;
- швидкість роботи на простих запитах - величезна перевага MSSQL полягає саме в швидкості роботи простих запитів.
- стабільність роботи - історично склалося, що MSSQL досить стабільна СКБД. PostgreSQL - більш молода, у той час як з-за більш раннього початку розробки у MSSQL склалося більша спільнота розробників;
- безпеку, пов'язану зі стабільністю - спільнота розробників MSSQL за всі ці роки знайшла і усунула величезну кількість вразливостей, що дозволяє вважати MSSQL однією з найбільш захищених СКБД;

- робота з великими об'єктами - в MSSQL реалізована підтримка бінарних об'єктів практично необмежених розмірів в полях типу BLOB, що відсутній у PostgreSQL;
- можливості для легкого зміни таблиць - в MSSQL реалізовані можливості легкого зміни таблиць, що відсутня в PostgreSQL.

У той же час у PostgreSQL є свої переваги:

- стабільність - незважаючи на те, що спільнота розробників MSSQL більше, сама PostgreSQL спочатку проектувалася як більш стабільна СУБД.
- швидкість роботи (процедури) - PostgreSQL виграє в продуктивності на складних запитах, логічно побудованих процедурах;
- цілісність даних - PostgreSQL дозволяє оперувати з даними, не перекладаючи логіку на ЯП. При розробці коду програмісту не доведеться думати про цілісність даних в БД;

Як можна побачити, головною перевагою MSSQL є швидкість роботи на простих запитах (логіка БД досить проста і не вимагає процедур для реалізації). Цю перевагу було обрано в якості основної при виборі СКБД.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТА ТЕСТОВИЙ ПРИКЛАД ПРОГРАМИ

3.1 Архітектурні рівні системи

В наш час існує безліч різних видів і типів архітектур, які успішно застосовуються. Однієї з найбільш використовуваних є класична трирівнева система, яка передбачає поділ додатка на три рівні.

Треба відразу відзначити, що багаторівневої архітектурою часто позначають два не зовсім пов'язаних поняття: n-layer і n-tier. I layer, і tier, як правило, позначаються словом "рівень", іноді по відношенню до "layer" ще вживається слово "шар". Однак в обох випадках рівні будуть різного порядку.

Tier представляє фізичний рівень. Тобто, якщо мати на увазі трирівневу архітектуру, то n-tier додаток міг бути розділений на такі рівні: сервер бази даних, веб-додаток на веб-сервері і браузер користувача. Тобто кожен рівень представляв би особливий окремий фізичний процес, навіть якби і сервер баз даних, і веб-сервер, і браузер користувача знаходилися б на одному комп'ютері. Якби в якості клієнта альтернативно використовувалося мобільний додаток, то це був би ще один фізичний рівень.

Layer представляє собою логічний рівень. Тобто у додатку може бути рівень доступу до даних, рівень бізнес-логіки, рівень уявлення, рівень сервісів і так далі. При цьому логічні рівні не збігаються з фізичними. Так, звичайно рівень надання в додатку ASP.NET містить і контролери, які обробляють введення, і уявлення, які відображаються в веб-браузері, тобто розділяється на два фізичних рівня.

В даній роботі на увазі матиметься саме логічні рівні, тобто n-layer архітектура.

Класична трирівнева система складається з наступних рівнів:

Presentation layer (рівень представлення): це той рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти для користувача інтерфейсу, механізм отримання введення від користувача. Стосовно до asp.net mvc на даному рівні розташовані уявлення і всі ті компоненти, який

складають призначений для користувача інтерфейс (стилі, статичні сторінки html, javascript), а також моделі уявлень, контролери, об'єкти контексту запиту.

Business layer (рівень бізнес-логіки): містить набір компонентів, які відповідають за обробку отриманих від рівня представлень даних, реалізує всю необхідну логіку додатка, все обчислення, взаємодіє з базою даних і передає рівнем подання результат обробки.

Data Access layer (рівень доступу до даних): зберігає моделі, що описують використовувані суті, також тут розміщуються специфічні класи для роботи з різними технологіями доступу до даних, наприклад, клас контексту даних Entity Framework. Тут також зберігаються репозиторії, через які рівень бізнес-логіки взаємодіє з базою даних.

При цьому треба зазначити, що крайні рівні не можуть взаємодіяти між собою, тобто рівень представлення (стосовно ASP.NET MVC, контролери) не можуть безпосередньо звертатися до бази даних і навіть до рівня доступу до даних, а тільки через рівень бізнес-логіки.

Рівень доступу до даних не залежить від інших рівнів, рівень бізнес-логіки залежить від рівня доступу до даних, а рівень представлення - від рівня бізнес-логіки.

Компоненти, як правило, повинні бути слабо зв'язаної (loose coupling), тому невід'ємною ланкою багаторівневих додатків є впровадження залежностей.

При чому ASP.NET MVC - перш за все стосується до рівня уявлення, інші ж рівні можуть бути реалізовані незалежно і можуть використовуватися в додатках на інших технологіях, як Windows Forms, WPF і т.д. І, як правило, весь додаток в цілому буде представляти рішення (solution) в Visual Studio, а окремі рівні - проекти. У той же час невірно думати, що якщо рівень обов'язково повинен відповідати окремий проект. При необхідності ми можемо роздрібнити один рівень на кілька проектів, головне, щоб його функціонал представляв єдину логічну ланку.

Тепер опишемо застосування трирівневої архітектури у розроблюваному проекті.

Першим рівнем, який було реалізовано, буде Data Access Layer або рівень доступу до даних. Цей рівень зазвичай містить всі моделі даних, що зберігаються в БД, а також класи, через які йде взаємодія з БД.

Після створення рішення і проекту додається в рішення новий проект по типу Class Library, який називається DAL. Цей проект представляє собою рівень доступу до даних.

Насамперед визначаються моделі, об'єкти яких будуть зберігатися в бд. Для цього в проект додається нова папка Entities, а в цій папці визначаються наступні класи, потрібні для розробки модуля тестування після проходження подкасту(рис.11).

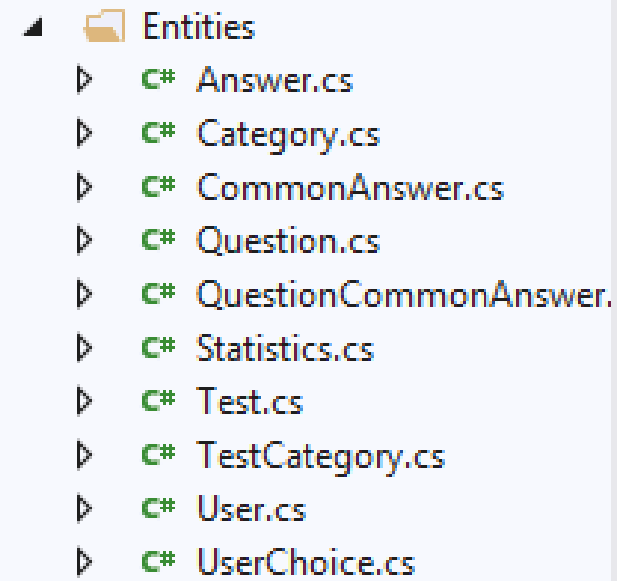

Рисунок 11. Моделі потрібні для системи тестування після подкастів.

В проект через NuGet додається пакет Entity Framework, додається клас контексту даних та конфігурації для кожної з моделей(рис.12).

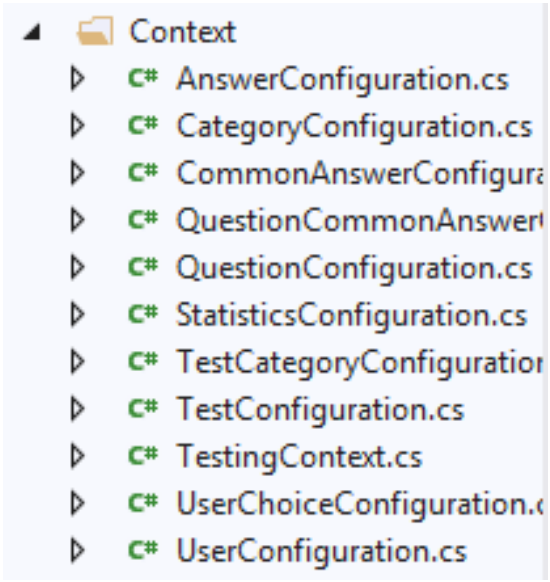

Рисунок 12. Контекст даних та конфігурації

Для збільшення гнучкості підключення до БД використовуються репозиторії. Тому спочатку в проекті визначається ще одна папка Interfaces. В ній створюються інтерфейси для кожного з репозиторіїв I[EntityName]Repository(рис.13).

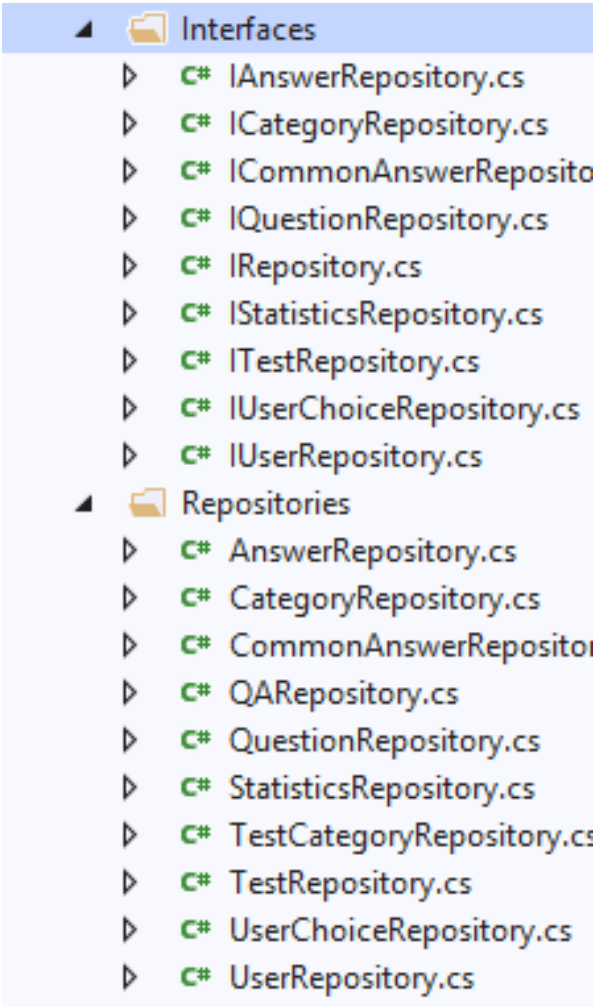

Рисунок 13. Репозиторії та їх інтерфейси

Оскільки використовуються кілька сховищ для кожної сутності, то для спрощення доступу до бд використовується патерн Unit Of Work. Для нього також додається новий інтерфейс IUnitOfWork(рис.14).

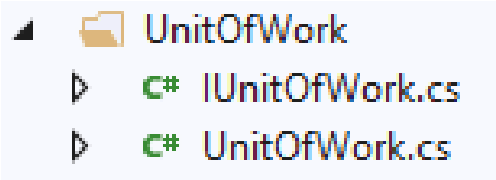

Рисунок 14. Unit of Work для системи тестування після подкастів.

Business Logic Layer або бізнес-рівень інкапсулює всю бізнес-логіку, всі необхідні обчислення, отримує об'єкти з рівня доступу до даних і передає їх на рівень представлення, або, навпаки, отримує дані з рівня уявлення і передає їх на рівень даних.

В рішення додається новий проект по типу Class Library, який називається BLL. Оскільки бізнес-рівень буде використовувати класи з рівня доступу до даних, додається референс на DAL рівень.

Рівень представлення не може безпосередньо отримувати дані з бази даних. В даному випадку BLL виступає в ролі посередника між двома рівнями. Але також у розроблюваному продукті враховується, що безпосередньо він не може передавати в контролери об'єкти DAL, так як рівень представлення не повинен мати доступ до функціональності рівня DAL. Тому використовуються проміжні DTO сутності(рис.15).

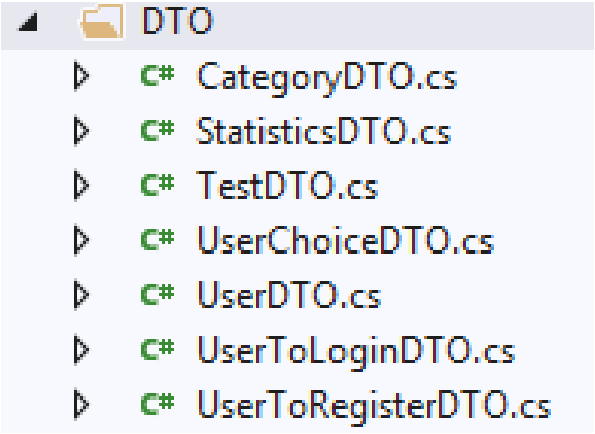

Рисунок 15. DTO для моделей

Для спрощення зіставлення класів моделей використовується допоміжний клас, який приховує логіку біліотеки AutoMapper.

Крім простих класів типу DTO BLL містити класи, які описують бізнес-логіку. Зокрема, паролі, токени для авторизації та інше. Це бізнес-модель, яка не тільки зберігає стан, як класи DTO, але і може описувати деякі дії.

Взаємодія між іншими двома рівнями відбувається через спеціальні сервіси, які потім імплементуються завдяки механізму Dependency Injection. Знову ж для більшої гнучкості визначаються також інтерфейси сервісів(рис.16).

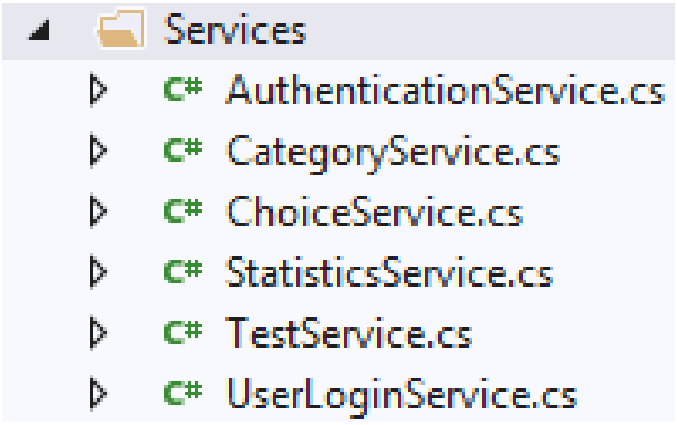

Рисунок 16. Сервіси та їх інтерфейси

Presentation Layer або рівень представлення відповідає за взаємодію з користувачем. Для цього в розроблюваному рішенні є проект WebAPI. Додається посилання на проект BLL.

Контролери забезпечують базові CRUD-операції та обробку помилок, а також базову маршрутизацію(рис. 17).

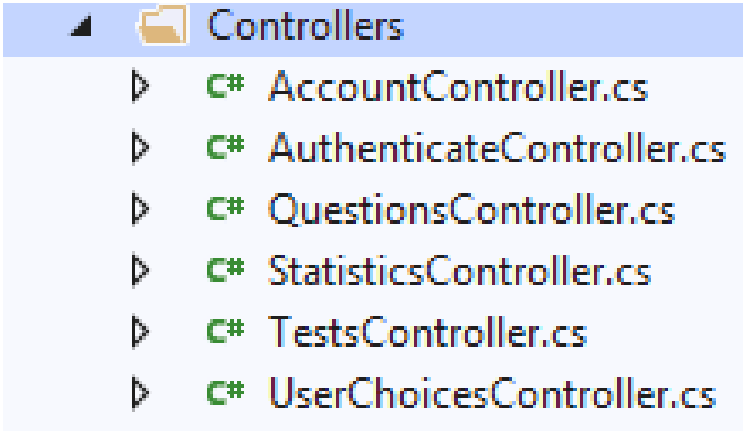

Рисунок 17. Контролери розроблюваного модулю

Таким чином створена серверна частина додатку програму, що реалізує трирівневу архітектуру, яке виконує всі ті ж дії, що і проект з монолітною

архітектурою. Однак тепер додаток легше тестувати, розробляти в команді розподілено, розвивати і розширювати.

3.2 Інтерфейс комплексу

Протестуємо додаток, запустивши серверну частину разом з Angular-модулем. Модуль тестування був розроблений окремим проектом, тобто після прослуховування того чи іншого подкасту API, що забезпечує тестування, буде з'являтись безпосередньо у додатку для вивчення мови, але також є можливість зайти на сайт тестувальної системи та обрати тести для проходження. Одразу відбувається переадресація на домашню сторінку, де відображаються ТОП-10 найпопулярніших тестів у додатку (рис. 18).

Рисунок 18. Домашня сторінка

Створена система пошуку за категорією або за назвою – обидва критерії оптимальні (рис.19).

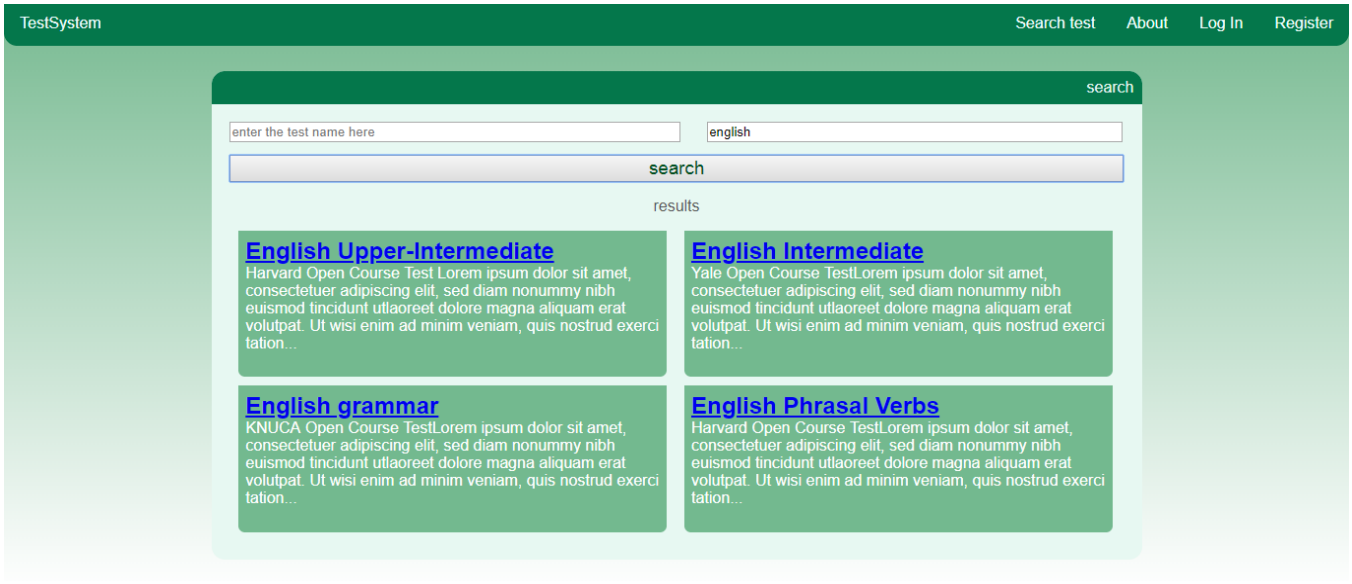

Рисунок 19. Пошукова система

На кожен з тестів можна перейти, та прочитати деталі тесту (рис. 20).

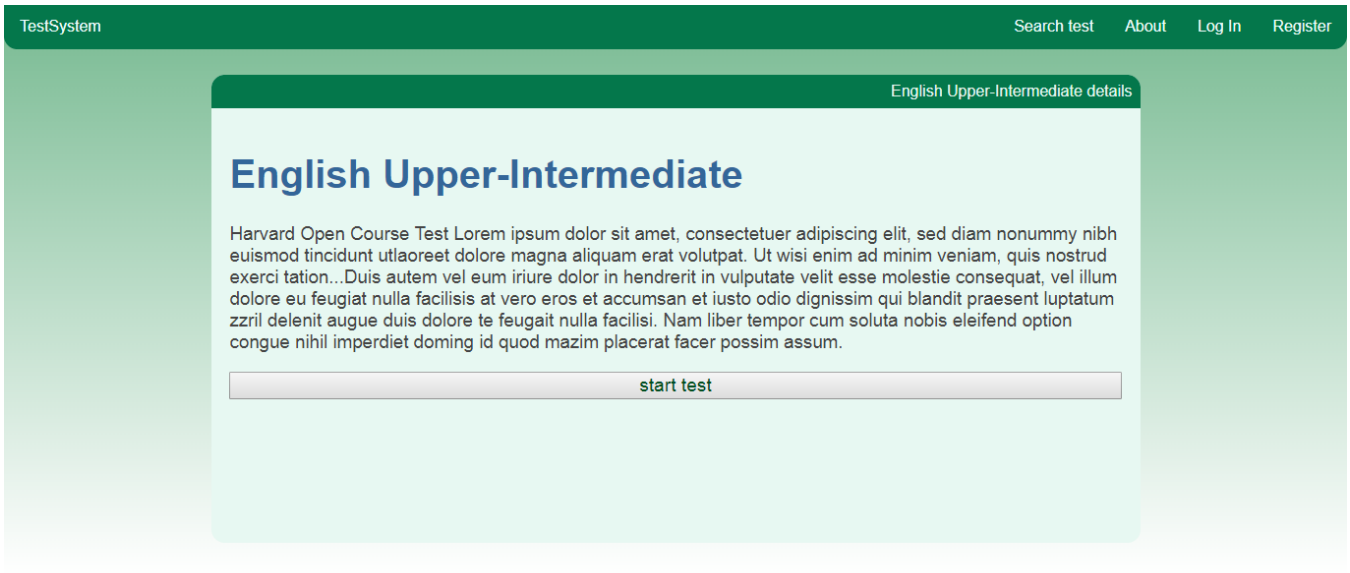

Рисунок 20. Деталі тесту

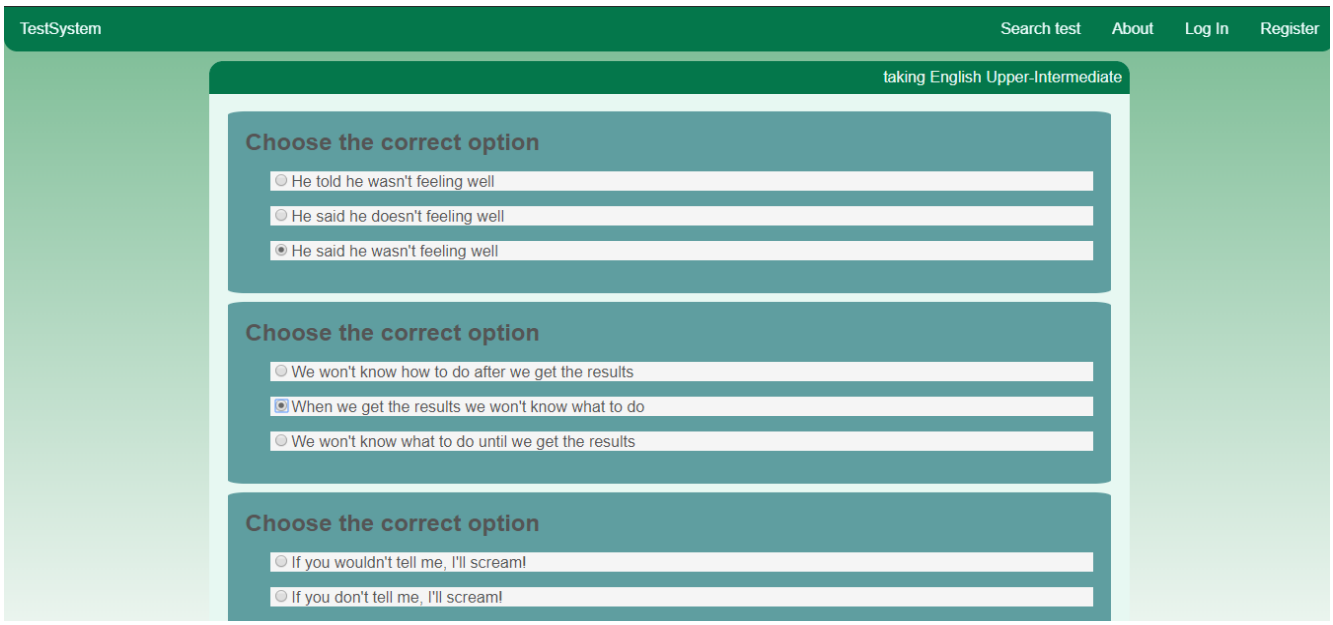

Рисунок 21.1. Процес проходження тесту

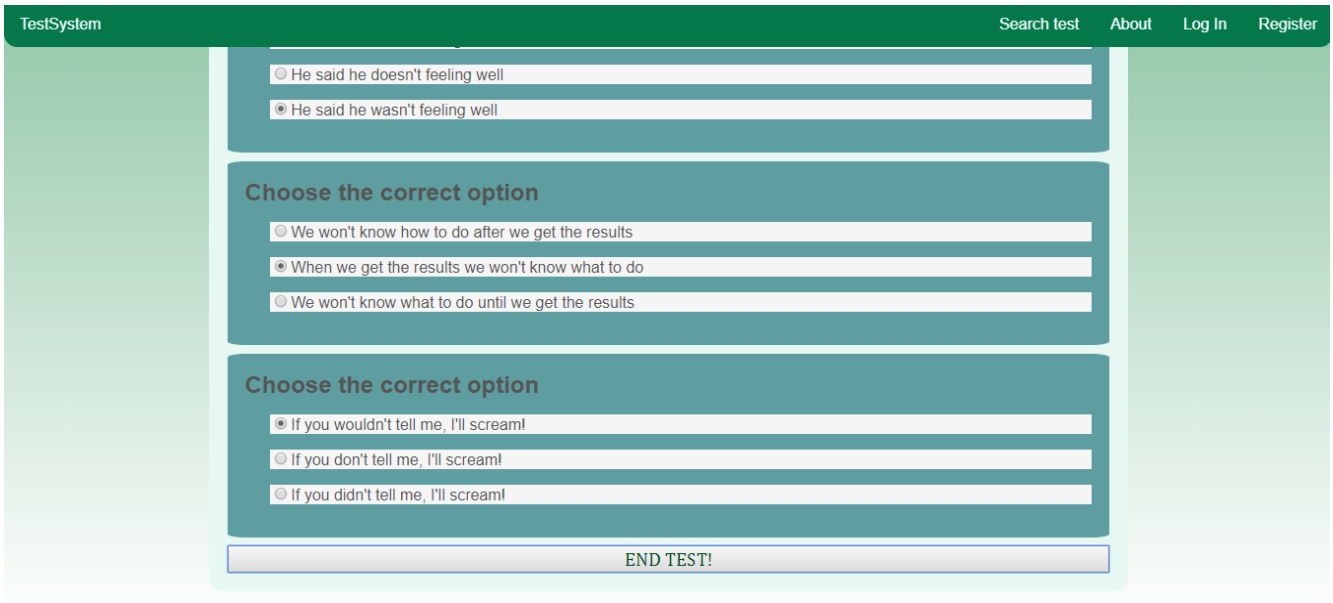

Рисунок 21.2. Процес проходження тесту

При завершенні тестування користувача очікує результат проходження тесту (рис. 22).

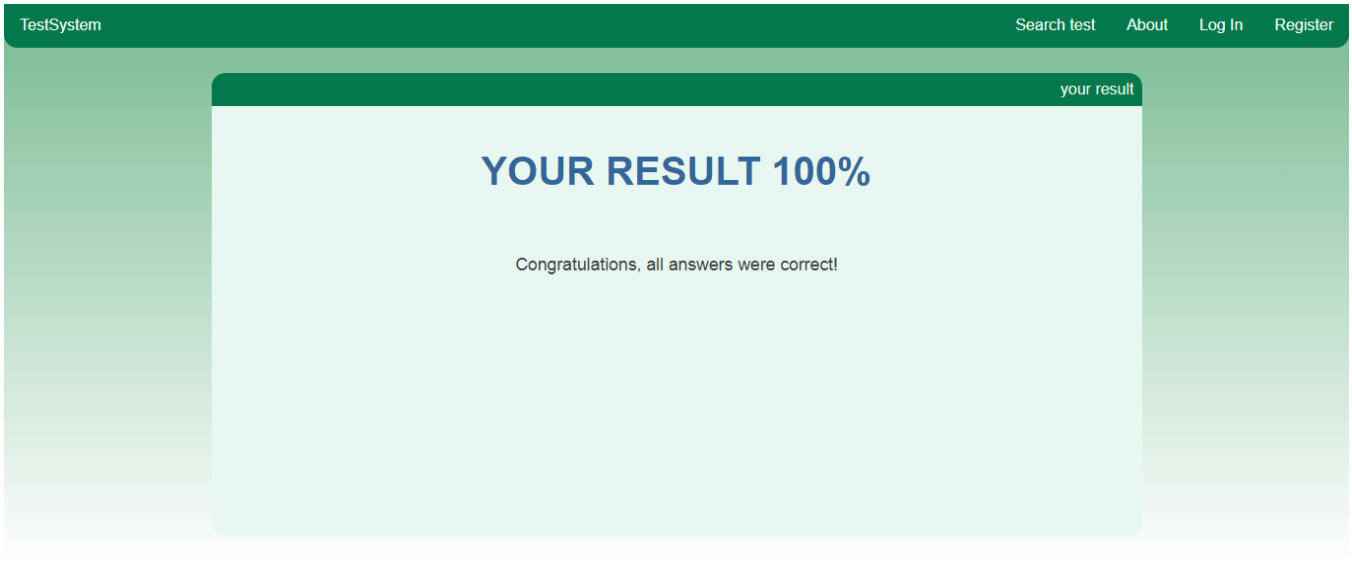

Рисунок 22. Результат проходження тестування

4. ЕРГОНОНОМІКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

4.1 Розрахунок часу евакуації людей при пожежі в приміщенні

Підприємство є одноповерховою будівлею, що відображена на рис. 23 розмірами 10 м. на 20м.; кількість робочих кімнат 8; кількість працюючих 13; кількість виходів 1.

Для розрахунку загального часу евакуації необхідно розрахувати час на кожній ділянці руху людей, починаючи від максимально віддаленої точки.

Рух людей під час процесу евакуації є вимушеним, тобто пов’язаним із необхідністю покинути приміщення чи будівлю через виниклу небезпеку. Вимушений рух людей має свої специфічні особливості, вже на початковій стадії, людині погрожує небезпека в результаті того, що пожежа супроводжується виділенням теплоти, продуктів повного й неповного згорання, токсичних речовин, обвалення конструкцій, що так чи інакше погрожує людині. Із цього слід зробити висновок, що при плануванні будівлі і устрої приміщень в них необхідно прийняти заходи, щоб процес евакуації міг закінчитися безпечно і в необхідний час.

Друга особливість полягає у тому, що в силу погрожуючої людині небезпеки рух інстинктивно починається одночасно в один і той же напрям – у сторону виходів. Це призводить до того, що проходи швидко заповнюються людьми при визначеній щільності потоків. Із збільшенням щільності потоків швидкість руху зменшується, що створює певний визначений ритм руху. В цій ситуації з’являється погроза утворення затору, і дуже важко запобігти їй.

Показником ефективності процесу вимушеної евакуації є час, на протязі якого люди можуть при необхідності покинути окремі приміщення і будівлю в цілому. Безпечність, досягнута тоді, коли цей час менший, ніж тривалість пожежі. Короткочасність процесу евакуації повинна досягатися не тільки конструктивно-планувальними рішеннями, на які звертали увагу раніше, але й організаційними рішеннями.

Процес евакуації людей можна поділити на три етапи :

- рух людей від найбільш віддаленої точки приміщення до евакуаційних виходів;
- рух людей від евакуаційних виходів до виходів на зовні ;
- рух людей від виходів із будівлі та їх розсіювання.

При евакуації основними параметрами, які характеризують процес руху людей є :

1) щільність людського потоку – *D*, люд/м2;

2) швидкість руху людського потоку – *v*, м/хв;

3) пропускна спроможність шляху (виходів) - *Q* ;

4) інтенсивність руху людського потоку - *q* ;

1) Щільність людського потоку *D*, яка складається з *N* людей, дорівнює:

$$D_1 = \frac{N_1 f}{A}, \text{ м}^2/\text{м}^2 \qquad (5.1),$$

де $A = g \cdot l$ – площа шляху евакуаційної ділянки [м²];

l – довжина ділянки; *g*- ширина ділянки;

f – площа горизонтальної проекції людини.

Якщо $D < 0.05$ людина має повну свободу пересування;

Якщо $0.05<D<0.15$ людина не може вільно змінювати напрямок свого руху;

Якщо $0.15<D<=0.92$ люди рухаються вкупі. Величина 0.92 є верхньою межею, коли люди рухаються вкупі, та нею обмежується щільність при проектуванні евакуаційних шляхів.

2) Швидкість руху людського потоку *v* залежить від його щільності *D* та виду шляху (горизонтальні чи похилі). Значення швидкості v, а також інтенсивності руху людського потоку q в залежності від його цільності D приведено в табл. 4.3.

Таблиця 4.1 Значення швидкості v і інтенсивності q руху людського потоку залежно від його щільності D

Щільність потоку м²/м², Д	Горизонтальний шлях		Дверний проем	Сходи вниз		Сходи вверх	
	Швидкість м/хв. v	Інтенсивність, q м/хв.	Інтенсивність, q м/хв.	Швидкість м/хв. v	Інтенсивність, q м/хв.	Швидкість м/хв. v	Інтенсивність, q м/хв.
0,01	100	1	1	100	1	60	0,6
0,05	100	5	5	100	5	60	3
0,1	80	8	8,7	95	9,5	53	5,3
0,2	60	12	13,4	68	13,6	40	8
0,4	40	16	18,4	40	16	26	10,4
0,6	27	16,2	19	24	14,4	18	10,8
0,8	19	15,2	17,3	13	10,4	13	10,4
0,9 и більше	15	13,5	8,5	8	7,2	11	9,9

3) Пропускна спроможність шляху Q (м/хв чи люд/хв)

$$Q = D \cdot v \cdot \delta \text{ , м}^2\text{/хв.}$$

(5.2)

4) Інтенсивністю руху людського потоку q (м/хв чи люд/хв)

$$q = D \cdot v$$

(5.3) Інтенсивність руху не залежить від ширини шляху і являється характеристикою потоку. Інтенсивністю руху людського потоку на кожному відрізку дорівнює:

$$q_i = \frac{q_{i-1}\delta_{i-1}}{\delta_i} \text{ , м/хв.}$$

(5.4)

де: δ_I, δ_{i-1} – ширина розглядаючого і-го і перед ним (і – 1) відрізків шляху, м;

q_i, q_{i-1} – значення інтенсивності руху потоку на розглядаючому і-му і перед ним (і - 1) відрізках шляху, м/хв.

Якщо q_i менше чи рівно q_{max}, то час руху на відрізку можні визначити по формулі:

$$t_1 = \frac{l_1}{v_1},$$

(5.5)

при цьому значение q_{max} треба приниймати рівним, м/хв.:

- для горизонтальних шляхів 16,5
- для дверних проємів 19,6
- для сходів вниз 16
- для сходів вверх 11

Розрахунковий час евакуації людей із приміщення й будівлі *tp* встановлюється по розрахунку часу руху людських потоків від найбільш віддалених місць розташування. При розрахунку весь шлях руху людського потоку поділяється на ділянки (прохід, коридор, сходинковий марш, дверний проріз, тамбур) довжиною l_i і шириною g_i.

Початковими ділянками являються проходи між робочими місцями.

Розрахунковий час евакуації дорівнює :

$$t_p = t_1 + t_2 + t_3 + \ldots + t_i = t \text{ [хв]}, \qquad t_i = l_i / v_i \text{ [хв]}.$$

де t_i – час руху людського потоку на кожній окремій ділянці.

Умова безпечної евакуації характеризується виразом $t_p <= t_{нб}$, тобто розрахункова тривалість вимушеної евакуації на різноманітних ділянках при розрахункових швидкостях людей і розрахунковій пропускній спроможності евакуаційних дверей повинна бути рівна або менша необхідного часу тривалості евакуації. Необхідний час евакуації $t_{нб}$ визначається по таблиці.

Використовуючи вище зазначений опис, за винятком таких ділянок як дверний проріз та тамбур (не передбачена у будівлі), проведемо розрахунок часу евакуації людей для прийнятого приміщення.

Маршрут евакуації розбивається на дев'ять етапів (ділянок). Для проведення розрахунку задамося планом евакуації людей (рис. 5.1).

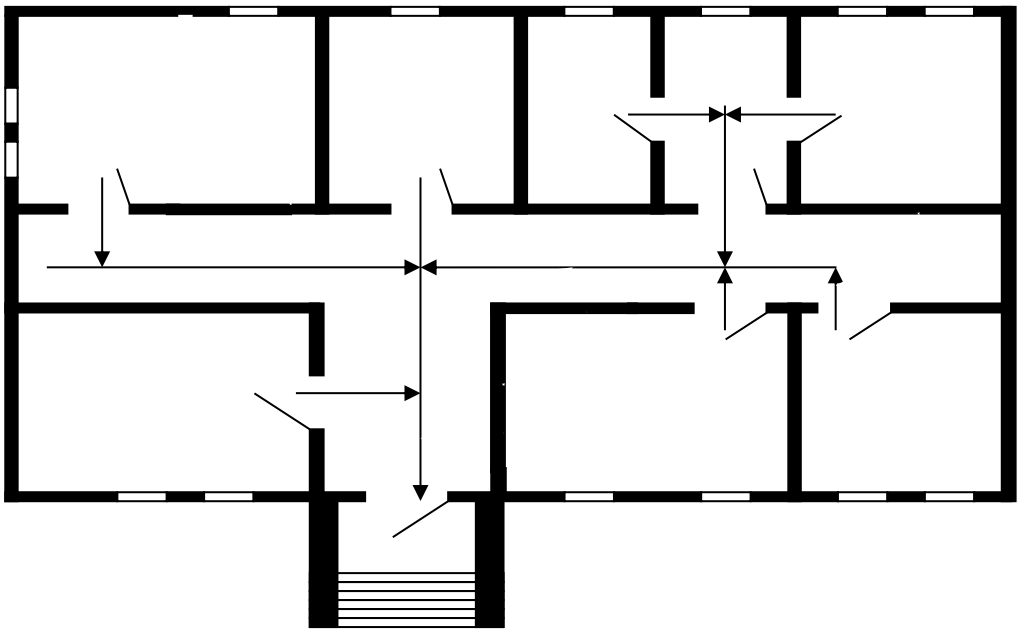

Рисунок 5.1 План евакуації людей

Перша ділянка.

Час руху людського потоку – вихід людей з кімнати № 1:

де l= 13 м – довжина ділянки ; v – швидкість руху на ділянці.

f = 0.113 м^2 – середня площа горизонтальної проекції людини ;

N = 2 – кількість людей ; S= 3 м – ширина ділянки .

$D_1 = 2\left(\frac{0.113}{3 \cdot 13}\right) = 0.006$ [м^2/м^2], тоді v_1 = 100 м/хв ; q_1= 1 м/хв.

t_1 = 13/100 =0,13 хв.

Друга ділянка.

Час руху людського потоку – вихід людей з кімнати № 2:

$D = 3\left(\frac{0.113}{11 \cdot 3}\right) = 0.01$ [м^2/м^2], тоді v_3 = 100 м/хв ; q_3= 1 м/хв.

t_2 = 11/100 =0,11 хв.

де l= 11 м; f= 0.113 м^2; N= 3; S= 3 м.

Третя ділянка.

Час руху людського потоку – вихід людей з кімнати № 3:

$D = 1\left(\frac{0.113}{12 \cdot 3}\right) = 0.003$ [м^2/м^2], тоді v_2 = 100 м/хв; q_2= 1 м/хв.

t = 12/100 =0,12 хв.

де l = 12 м; f = 0.113 м^2; N = 1; S = 3 м.

Четверта ділянка.

Час руху людського потоку – вихід людей з кімнати № 4:

$D = 2\left(\frac{0.113}{5 \cdot 3}\right) = 0.01$ [м^2/м^2], тоді v_4 = 100 м/хв; q_4= 1 м/хв.

t = 5/100 =0,05 хв.

де l = 5 м; f = 0.113 м^2; N = 2; S = 3 м.

П'ята ділянка.

Час руху людського потоку – вихід людей з кімнати № 5:

$D = 2\left(\frac{0.113}{12 \cdot 3}\right) = 0.007$ [м^2/м^2], тоді v_5 = 100 м/хв ; q_5= 1 м/хв.

t = 12/100 =0,12 хв.

де l = 12 м; f = 0.113 м^2; N = 2; S = 3 м.

Шоста ділянка.

Час руху людського потоку – вихід людей з кімнати № 6:

$D = 2\left(\frac{0.113}{12 \cdot 3}\right) = 0.007$ [м^2/м^2], тоді v_6 = 100 м/хв; q_6= 1 м/хв.

t = 12/100 =0,12 хв.

де l = 12 м; f = 0.113 м^2; N = 2; S = 3 м.

Сьома ділянка.

Час руху людського потоку – вихід людей з кімнати № 7:

$D = 2\left(\frac{0.113}{9 \cdot 3}\right) = 0.008$ [м^2/м^2], тоді v_7 = 100 м/хв; q_7= 1 м/в..

Т = 9/100 =0,09 хв.

Де l = 9 м; f = 0.113 м^2; N = 2; S = 3 м.

Восьма ділянка.

Час руху людського потоку – вихід людей з кімнати № 8:

$D = 2\left(\frac{0.113}{3 \cdot 3}\right) = 0.02$ [м^2/м^2], тоді v_8 = 100 м/хв; q_8 = 1 м/хв.

t = 3/100 =0,03 хв.

де l = 3 м; f = 0.113 м^2; N = 2; S = 3 м.

Дев'ята ділянка.

Час руху людського потоку – вихід людей з кімнати № 9:

$D = 7\left(\frac{0.113}{9 \cdot 3}\right) = 0.03$ [м^2/м^2], тоді v_9 = 100 м/хв; q_9 = 1 м/хв.

t = 9/100 =0,09 хв.

де l = 9 м; f = 0.113 м^2; N = 7; S = 3 м.

Десята ділянка.

Час руху людського потоку – вихід людей з кімнати № 10:

$D = 11\left(\frac{0.113}{5 \cdot 3}\right) = 0.08$ [м^2/м^2], тоді v_{10} = 100 м/хв; q_{10} = 1 м/в..

T = 5/100 =0,05 хв.

Де l = 5 м; f = 0.113 м^2 ; N = 11; S = 3 м.

Одинадцята ділянка.

Час руху людського потоку – вихід людей з кімнати № 11:

$D = 13\left(\frac{0.113}{3 \cdot 3}\right) = 0.1632$ [м^2/м^2], тоді v_{11} = 60 м/хв; q_{11} = 12 м/хв.

t = 3/60 =0,05 хв.

де l = 3 м; f = 0.113 м^2 ; N = 13; S = 3 м.

Загальний час евакуації : $t = t1 + t2 + ... + t18$ = 1,01 [хв].

$t_{нб}$ = 2,5 хвилин для одноповерхового будинку;

t = 1,01 < $tнб$ = 2,5 хв, тобто вимоги пожежної безпеки виконуються.

В зв'язку з можливістю виникнення пожежі на території будівлі внаслідок несправної роботи комп'ютернрї техніки, яка підключена до електромережі, я вирішив вибрати вуглекислотні вогнегасники моделі ОУ-8 та

порошкові – моделі ОП-8Б. Розмістити їх необхідно на пожежних щитах в вестибюлі та біля пожежного, по одному екземпляру кожного типу.

За допомогою вогнегасника ОУ-8 можна гасити різні речовини, крім тих, які можуть горіти без доступу повітря. Також їм можна тушити пожежу в пристроях під напругою до 1000V, при умові приближення по струмопровідних частин не ближче одного метру.

Механізм припинення горіння за допомогою використання вуглекислого газу базується на його властивостях шляхом розбавлення знижувати концентрацію реагуючих речовин до рівня, при якому горіння становиться неможливим.

За допомогою вогнегасника ОП-8Б можна тушити палаюче електрообладнання під напругою до 1000V, легкозаймисті рідини, тліючі матеріали (навіть ті що горять без доступу повітря) праці в робочому приміщенні.

4.2 Ергономічні вимоги до організації робочих місць з комп`ютерною технікою

Оператор обробки інформації при виконанні своєї роботи майже весь робочий час знаходиться в сидячому положенні за робочим столом, на якому розташоване його робоче обладнання. Для запобігання виникнення, пов`язаних з таким видом робіт, хвороб (скаліоз, хвороби очей та ін.), а також для усунення загального дискомфорту, зменшення втомлюваності працівника, підвищенню його продуктивності необхілно правильно організувати робоче місце.

Організація робочого місця передбачає:

- правильне розміщення робочого місця у виробничому приміщенні;
- вибір ергономічного обгрунтованого робочого положення, виробничих меблів з урахуванням антропометричних характеристик людини;
- раціональну компановку обладнання на робочих місцях;
- урахування характеру та особливостей трудової діяльності;
- ДНАОП 0.00-1.31-99, ГОСТ 12.2.032-78, ДСанПІН 3.3.2.007-98 регламентує такі вимоги до організації робочого місця користувача ВДТ (візуальний дисплейний термінал):

1) Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Рекомендовані розміри столу: висота – 725 мм, ширина – 600-1400 мм, глибина – 80-1000 мм. Робочий стіл повинен мати простір для ніг висотою не менше ніж 450 мм, на рівні витягнутої ноги не менше 650 мм.

Робоче місце має бути обладнане підставкою для ніг шириною не менше ніж 300 мм, глибиною не менше ніж 400 мм, з можливістю регулювання по висоті в межах 150 мм та кута нахилу опорної поверхні – в межах 20°. Підставка повина мати рифлену поверхню і бортик по передньому краю заввишки 10 мм.

2) Робочий стілець користувача ВДТ повинен мати такі основні елементи: сидіння, спинку та стаціонарні або знімні підлокітники. Робочий стілець має бути підйомно – поворотним, регульованим за висотою, за кутом нахилу сидіння та спинки і за відстанню від спинки до попереднього краю сидіння. Поверхня сидіння має бути плоскою, передній край заокругленим.

Висота поверхні сидіння має регулюватися в межах 400…500 мм, а ширина і глибина становити не менше ніж 400 мм. Кут нахилу сидіння – до 15° вперед і до 5° назад.

Висота спинки має становити (300 ± 20) мм, ширина – не менше ніж 380 мм, радіус кривизни горизонтальної площини – 400 мм. Кут нахилу спинки має регулюватися в межах 0…30° від вертикального положення. Відстань від спинки до переднього краю сидіння має регулюватися в межах 260…400 мм.

Для зниження статичного навантаження м`язів верхніх кінцівок слід використовувати стаціонарні або знімні підлокітники довжиною не менше ніж 250 мм, шириною не менше ніж 50…70 мм. Що регулюються за висотою над сидінням у межах 230…260 мм і відстанню між підлокітниками в межах 350…500 мм.

Поверхня сидіння і спинки стільця має бути напівм`якою з нековзним, повітронепроникненим покриттям, що легко чиститься і не електризується.

Конструкція виробничих меблів для користувача ВДТ має бути такою, щоб забезпечувати йому підтримання оптимальної робочої пози з такими ергономічними характеристиками: ступні ніг – на підлозі або на підставці для ніг; стегна – в горизонтальній площині; верхні частини рук – вертикальні; кут ліктьового суглоба (між плечем та передпліччям) – 70 -90°; зап`ястки зігнуті під кутом не більше 20° відносно горизонтальної площини, нахил голови вперед в межах 15-20° до вертикалі.

3) Дисплей має розташуватися на столі на відстані від очей користувача не більше 700 мм (оптиальна відстань 450 – 500 мм). Розташування екрану має забезпечувати зручність зорового спостереження у

вертикальній площині під кутом + 30° до нормальної лінії погляду працюючого. В горизонтальній площині кут спостереження екрану не повинен перевищувати 60°.

4) Клавіатуру слід розташувати на поверхні столу на відстані 100…300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачитися опорний пристрій, який дає змогу зиінювати кут нахилу поверхні клавіатури у межах 5…10°. Висота середнього рядка клавіш має не перевищувати 30 мм. Поверхня клавіатури має бути матовою з коефіцієнтом відбиття 0,4.

5) Документ для вводу даних розташовується на відстані 450…500 мм від очей працівника, переважно зліва, кут між екраном дисплея та документом в горизонтальній площині має бути 30 - 40°.

6) Розміщення принтера або іншого пристрою введення – виведення інформації на робочому місці має забезпечувати добру видимість екрана ВДТ, зручність ручного керування пристроєм введення – виведення інформації в зоні досяжності: по висоті 900 – 1300 мм, по глибині 400 – 500 мм. Під принтери ударної дії потрібно підкладати вібраційні килимки для гасіння вібрації та шуму.

На рис. 5.2 зображено вид робочого місця з ВДТ:

A-принтер. B-монітор. C-системний блок. D-клавіатура. E-папка для документів.

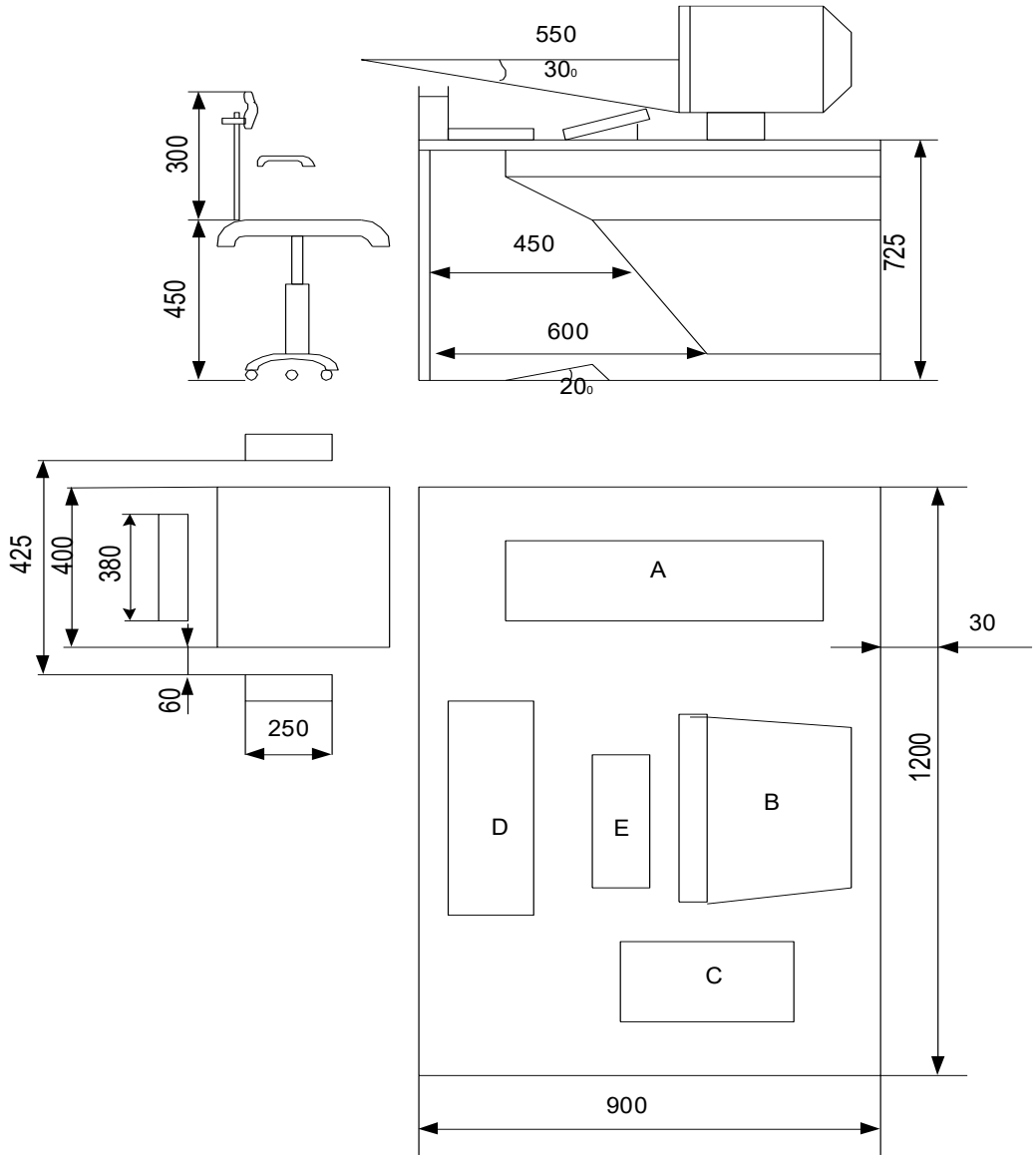

Рисунок 5.2 Вид робочого місця з ВДТ

ВИСНОВКИ

В результаті атестаційної випускової роботи було проведено:

1. аналіз предметної області.
2. проектування бази даних системи.
3. розробка програмного забезпечення системи в сучасних умовах.
4. ергономічні дослідження в області інформаційних технологій.

Розроблена система є гнучкою, отже підходить для впровадження на різних організаціях типу шкіл/курсів з вивчення іноземних мов, а також в університетах для забезпечення дистанційного навчання.

Розробка на ASP NET CORE забезпечила кросплатформність, а розроблений Web Api може бути використаний безпосередньо на сайтах шкіл/університетів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wikipedia – вільна енциклопедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BA%D0%B0%D1%81%D1%82

2. Buklib – Буковинська бібліотека [Електронний ресурс] – Режим доступу: https://buklib.net/books/25617/

3. Lviv Polytechnic National University Gebeto [Електронний ресурс] – Режим доступу: https://gebeto.github.io/nulp/software-construction/lab-1.html

4. GitHub, Inc. [Електронний ресурс] – Режим доступу: https://github.com/gebeto/nulp/blob/master/software-construction/lab-1.md

5. Microsoft – офіційна сторінка [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/ef/core/

6. Angular official [Електронний ресурс] – Режим доступу: https://angular.io/docs

7. Анисимов В. В. // Проектирование информационных систем [Електронний ресурс] – Режим доступу: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema15/tema15_2

8. ИНТУИТ Национальный открытый университет [Електронний ресурс] – Режим доступу: https://www.intuit.ru/studies/courses/32/32/lecture/1022

9. INFORMICUS [Електронний ресурс] – Режим доступу: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4

10. [Електронний ресурс] – Режим доступу: https://studopedia.info/6-19535.html

11. Microsoft – офіційна сторінка [Електронний ресурс] – Режим доступу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019