Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)
Кафедра обчислювальної техніки
(повна назва кафедри)

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Кросплатформна інформаційна система для допомоги у вивченні іноземної мови»

08-54.БКР.007.00.000 ПЗ

Виконав: студент 4 курсу, групи 1КІ-21б
спеціальності
123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
освітня програма «Комп'ютерна інженерія»
Загубін О.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ОТ
Крупельницький Л.В.
(прізвище та ініціали)
Рецензент: к.т.н, доц.каф. ПЗ
Ракитянська Г.Б.
(прізвище та ініціали)

Допущено до захисту

Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д

"10" 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Обчислювальної техніки
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 - Інформаційні технології
Спеціальність 123 - «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д
_____
"21" 03 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Загубіну О.В.
(прізвище, ім'я, по батькові)

1. Тема роботи «Кросплатформна інформаційна система для допомоги у вивченні іноземної мови», керівник роботи к.т.н., доц. Крупельницький Л.В., затверджені наказом ВНТУ від «20» березня 2025 року №97.

2. Термін подання студентом роботи 13.05.25

3. Вихідні дані до роботі: призначення - набір базової інформації, параметри тестувальних блоків, технічних вимог, навчального контенту та попереднього аналізу, на основі якого здійснюється планування і реалізація всієї системи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз предметної області, вибір та обгрунтування інструментів для розробки інформаційної системи, розробка веб застосунку, тестування розробленої системи в кросплатформному середовищі.

5. Перелік графічного матеріалу: UML схема інформаційної системи, ER схема бази даних.

6. Консультанти розділів роботи приведений в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завдання прийняв
Розділи 1 – 4	Крупельницький Л.В., доцент кафедри ОТ	21.03.25	13.05.25
Нормоконтроль	Швець С.І., Асис. каф. ОТ	14.05.25	30.05.25

7. Дата видачі завдання « 21 » березня 2025 р.

8. Календарний план етапів бакалавровської кваліфікаційної роботи наведе
в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання комплексної бакалаврської роботи	Строк виконання етапів роботи		Примітка
		початок	закінчення	
1	Постановка задачі	22.03.25	26.03.25	виконан
2	Аналіз предметної області	28.03.25	06.04.25	виконан
3	Вибір та обгрунтування інструментів для розробки інформаційної системи	08.04.25	14.04.25	виконан
4	Розробка веб застосунку	15.04.25	28.04.25	виконан
5	Тестування розробленої системи в кросплатформному середовищі	28.04.25	10.05.25	виконан
6	Оформлення пояснювальної записки	11.05.25	1.05.25	виконан
7	Попередній захист БКР	13.05.25	13.05.25	виконан
8	Захист БКР	03.06.25	03.06.25	виконан

Студент _____ (підпис) Загубін О.В.

Керівник роботи _____ (підпис) Крупельницький Л.В.

АНОТАЦІЯ

УДК 004.451

Загубін О.В. Кросплатформна інформаційна система для допомоги у вивченні іноземної мови. Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, Вінниця: ВНТУ, 2025. 76 с.

На укр. мові. Бібліогр.: 22 назв; рис.: 20.

В даній бакалаврській кваліфікаційній роботі було розроблено вебзастосунок для допомоги у вивченні іноземної мови. На основі аналізу актуальності теми було здійснено дослідження сучасних рішень у сфері мовного навчання, їх функціональних можливостей, переваг і недоліків. З урахуванням цього було спроєктовано інформаційну систему, яка поєднує навчальні модулі, тести, систему оцінювання результатів та гейміфіковані елементи взаємодії з користувачем. Особливу увагу приділено інтуїтивному інтерфейсу, адаптивності дизайну та педагогічній ефективності подання матеріалу.

Ключові слова: вебзастосунок, вивчення іноземної мови, тестування знань, гейміфікація, навчальні модулі.

ABSTRACT

Zagubin O.V. Cross-platform information system to help in learning a foreign language. Bachelor's qualification work in specialty 123 - Computer Engineering, Vinnytsia: VNTU, 2025. 76 p.

In Ukrainian. Bibliography: 22 titles; fig.: 20.

In this bachelor's qualification work, a web-site was developed to help in learning a foreign language. Based on the analysis of the relevance of the topic, a study of modern solutions in the field of language learning, their functional capabilities, advantages and disadvantages was carried out. With this in mind, an information system was designed that combines training modules, tests, a system for evaluating results and gamified elements of interaction with the user. Particular attention is paid to the intuitive interface, the adaptability of design and the pedagogical effectiveness of the presentation of the material.

Key words: web-session, foreign language learning, knowledge testing, gamification, training modules.

ЗМІСТ

ВСТУП

У сучасному світі знання іноземних мов є важливою складовою професійного та особистісного розвитку. З огляду на активне впровадження цифрових технологій у сферу освіти, зростає потреба у зручних, ефективних і доступних інструментах для самостійного навчання. Одним із таких рішень є вебзастосунки, що поєднують інтерактивність, візуальну привабливість та логічну структуру навчального матеріалу, який допоможе користувачам поступово опановувати іноземну мову, поєднуючи теоретичні знання, практичні вправи та систему мотивації для підтримки регулярності та зацікавлення у навчальному процесі.

Актуальність теми створення веб застосунку для допомоги у вивченні іноземної мови зумовлена кількома ключовими факторами. По-перше, глобалізація та розвиток міжнародних зв'язків вимагають від людей знання хоча б однієї іноземної мови, зокрема англійської, для успішної комунікації, працевлаштування, навчання чи подорожей. По-друге, традиційні методи навчання часто не забезпечують належного рівня гнучкості, адаптивності та мотивації, які необхідні сучасному користувачу. По-третє, поширення доступу до інтернету та цифрових пристроїв створює сприятливі умови для самостійного онлайн-навчання в будь-який зручний час і з будь-якого місця. Водночас на ринку все ще існує попит на якісні освітні продукти, які б поєднували простий інтерфейс, продуману структуру, інтерактивні завдання, персоналізований підхід до користувача та ігрові елементи для підтримки мотивації. У цьому контексті розробка веб застосунку, що відповідає сучасним вимогам до онлайн-освіти, є своєчасною та затребуваною ініціативою, яка має як навчальну, так і практичну цінність.

Метою дослідження є розробка, реалізація та аналіз ефективності веб застосунку, що сприяє вивченню іноземної мови шляхом інтерактивної подачі навчального матеріалу, структурування інформації за логікою поступового ускладнення тем, впровадження системи тестування після кожного розділу, а

також використання елементів для підвищення мотивації користувача. Особлива увага приділяється створенню зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє самостійно освоювати навчальні блоки, слідкувати за власним прогресом і отримувати миттєвий зворотний зв'язок, що відповідає сучасним підходам до цифрової освіти.

Об'єктом дослідження є процес вивчення іноземної мови за допомогою цифрових технологій, зокрема засобами веб застосунків, які поєднують в собі навчальні матеріали, інтерактивні завдання, тести для забезпечення ефективного та мотивованого навчання користувача.

Предметом дослідження є структура, функціональність та принципи розробки вебзастосунку, орієнтованого на підтримку користувачів у процесі вивчення іноземної мови, з акцентом на інтерактивність, персоналізацію навчального контенту та зручність користувацького інтерфейсу.

Задачі проєкту:

— оцінити вимоги до веб застосунку;

— вибрати інструменти для розробки інформаційної системи;

— реалізувати структуру та основну логіку веб застосунку;

— протестувати систему у кросплатформному середовищі;

Апробація результату бакалаврської кваліфікаційної роботи була проведена на конференції "Молодь в науці: дослідження, проблеми, перспективи" (МН-2025) [1]:

Загубін О. В., Крупельницький Л. В. Кросплатформна інформаційна система для допомоги у вивченні іноземної мови // Матеріали всеукраїнської науково-практичної Інтернет-конференції студентів та молодих науковців — Режим доступу : https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25305

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вимоги до веб застосунку для допомоги у вивченні іноземної мови

Веб-застосунок для вивчення іноземної мови — це не просто набір сторінок із текстами чи завданнями, а повноцінне інтерактивне середовище, яке має відповідати сучасним вимогам до цифрових освітніх інструментів. Зростаючі очікування користувачів, конкуренція серед освітніх платформ, технологічний прогрес і методичні рекомендації в галузі викладання мов — усе це формує комплекс вимог до подібних застосунків. Перш за все, ключовою є інтуїтивність інтерфейсу та зручність користування. Сучасний користувач, незалежно від віку чи технічного досвіду, очікує, що інтерфейс буде зрозумілим з перших секунд, без потреби у додаткових інструкціях. У веб-застосунку для мовного навчання особливо важливо, щоб розділи були логічно впорядковані, а доступ до матеріалів — максимально швидким і простим. Користувач має бачити, де він знаходиться, які уроки вже пройдені, скільки балів або зірок він набрав, та що йому потрібно зробити далі.

Наступною критичною вимогою є адаптивність інтерфейсу для різних пристроїв. Люди вивчають мови на телефонах, планшетах, ноутбуках, іноді перемикаються між пристроями — і вся їхня історія навчання має зберігатися, а інтерфейс повинен однаково якісно працювати на будь-якому екрані. Технічно це означає застосування адаптивного дизайну, оптимізацію швидкості завантаження, правильне масштабування тексту й інтерактивних елементів.

Також не менш важливим є створення гнучкої системи навчальних модулів, яка дозволяє налаштувати шлях користувача відповідно до його рівня та темпу. Це включає розбиття курсу на теми, підрозділи, вправи, тести — усе має працювати автономно, але бути частиною цілісної системи. Важливо, щоб завдання не йшли суцільним потоком без логіки — кожне з них має бути прив’язане до конкретної навчальної мети: вивчення нових слів, граматичних правил, тренування читання або слухового сприйняття. При цьому ідеальним є поступовий ріст складності. Для цього необхідно дотримуватись дидактичної послідовності: наприклад, починати з вивчення алфавіту, переходити до простих

фраз у діалогах, далі — граматика, і зрештою — читання автентичних текстів. Саме так реалізована побудова курсу в нашому проєкті, де послідовність блоків виконує не лише структурну, але й методичну функцію.

Крім навчального контенту, веб-застосунок має передбачати систему оцінювання та відстеження прогресу. Це можуть бути бали, зірки, значки — усе, що створює мотивацію та дає користувачеві зрозуміле уявлення про власний рівень. Такі елементи гейміфікації мають бути простими, але водночас інформативними. Оцінювання повинне давати зворотний зв’язок — наприклад, не лише фіксувати помилки, але й пояснювати їх, підказувати, на що звернути увагу. У нашому проєкті це реалізовано через зіркову систему, яка наочно демонструє результат за кожен розділ, і мотивує користувача покращити свої показники, повторюючи тести при потребі.

З точки зору безпеки та персоналізації важливо впровадити систему акаунтів, щоб кожен користувач мав можливість зберігати свій прогрес, повертатися до навчання з того самого місця, редагувати профіль або вийти з акаунту, коли потрібно. Така система також дозволяє у майбутньому реалізувати більш складні функції, такі як синхронізація між пристроями, персональні рекомендації, або навіть спільне навчання в групах. У нашому випадку вже реалізовано базову авторизацію та кнопку виходу з акаунту, що формує підґрунтя для розширення функціональності в подальшому.

Серед додаткових вимог, характерних для сучасних застосунків для вивчення мов, варто згадати мультимедійність, тобто підтримку аудіо та відео контенту, інтерактивні вправи типу "перетягни слово", "обери відповідь зі списку", "склади речення", тощо. Хоча наш проєкт наразі не містить складних інтерактивних елементів, його структура вже повністю готова до включення таких функцій у наступних ітераціях. Прості тести, реалізовані зараз, вже демонструють гнучку систему взаємодії з користувачем і забезпечують збереження результатів у зрозумілій формі.

Розроблений веб-застосунок виконує основні сучасні вимоги до освітніх мовних платформ: має логічну структуру курсу, простий і доступний інтерфейс,

базову гейміфікацію, адаптацію до особистого темпу навчання, збереження прогресу та систему акаунтів. Він закладає надійний фундамент для подальшого вдосконалення та масштабування, при цьому вже зараз здатен забезпечити ефективне, послідовне та мотивуюче вивчення іноземної мови [2].

1.2 Аналіз функціоналу та можливостей користувачів

Веб застосунок є багатофункціональним освітнім середовищем, що поєднує послідовну логіку навчального процесу з гнучкістю у взаємодії користувача та системною підтримкою навчального прогресу. У його основі лежить чітко структурована послідовність уроків, кожен із яких є окремим тематичним блоком: Алфавіт, Діалоги, Граматика, Читання книги (секції 1 та 2). Ця логіка не є випадковою — вона відповідає природному шляху опанування нової мови, де кожен наступний крок базується на засвоєному раніше, а сам підбір тем реалізований з урахуванням як педагогічної доцільності, так і зручності для користувача.

На першому етапі користувач стикається з алфавітом — це фундаментальний блок, у якому людина знайомиться з буквами, їх написанням, звучанням і візуальним розпізнаванням. Це надзвичайно важливо, оскільки без навички впізнавання букв не можна ефективно читати чи формулювати слова. Блок "Алфавіт" виступає в ролі стартового фільтра — він не є складним, але задає ритм навчанню і допомагає зануритися в контекст мови. Далі йде розділ "Діалоги", який розширює сприйняття мови через прості, часто вживані мовні конструкції. Це дозволяє перейти від букв до реального мовного вжитку, коли користувач починає бачити, як слова формують змістовні речення. Завдання в цьому блоці спрямовані на розвиток розуміння коротких фраз, звичних для щоденного спілкування, що створює базу для формування почуття мовної логіки.

Третім кроком є розділ "Граматика", у якому користувач поглиблює свої знання та вчиться самостійно створювати правильні речення, оперуючи граматичними правилами. Цей перехід надзвичайно важливий, адже саме граматика дозволяє людині не лише повторювати готові фрази, а й будувати

власні. Завдання тут вже вимагають більшої уваги, аналізу та гнучкості мислення, але зважаючи на логічний поступ попередніх блоків, користувач підготовлений до цього виклику. І завершальним етапом, який інтегрує все вивчене, є практика у вигляді читання книги, зокрема фрагментів із "Гаррі Поттера". Це не просто художній текст — це знайома багатьом історія, що викликає зацікавлення і дозволяє застосувати знання в реальному контексті, у живій мові, яка містить як прості, так і складні структури. Книга розбита на секції, кожна з яких містить адаптований фрагмент, після якого користувач проходить тест, що дозволяє оцінити не тільки розуміння прочитаного, а й граматичну точність, словниковий запас і здатність до інтерпретації змісту [3].

Особливу увагу варто приділити тому, як реалізовані тести після кожного розділу. Вони є окремими та самостійними, що означає, що користувач не зобов’язаний проходити весь курс одразу або в строго заданому темпі. Якщо певна тема виявилася складною або незрозумілою, її можна повторювати стільки разів, скільки потрібно, без обмежень. Це не лише дозволяє відшліфовувати знання до досконалості, а й формує здорову звичку — вчитись у своєму ритмі, не боячись помилок. Такий підхід також важливий для користувачів із різними стилями навчання — хтось потребує більше повторень, хтось краще сприймає з першого разу, а хтось — візуально через читання. Гнучкість, яку надає ця структура, гарантує, що кожен знайде для себе зручний спосіб опанування матеріалу. Зіркова система оцінювання додатково мотивує користувача — з одного боку, вона проста і зрозуміла, з іншого — дає можливість бачити не тільки факт проходження, але й якість результату. Якщо результат нижчий за очікуваний, видно це одразу, і це спонукає повторити спробу [3].

Усі ці елементи взаємодіють із загальною системою акаунту користувача, що зберігає прогрес, результати тестів, кількість зірок і дозволяє розпочинати роботу з будь-якого пристрою без втрати інформації. Інтерфейс побудований так, що кожен розділ легко відкривається, відображає саме той контент, який очікується, без збоїв чи логічних помилок, що критично важливо для стабільності навчального досвіду. У нижній частині сторінки розміщена кнопка виходу з

акаунту — вона забезпечує контроль над особистим простором і даними. Це свідчить про наявність базової системи авторизації, що також є важливою умовою для персоналізованого досвіду — результат кожного користувача зберігається окремо, забезпечуючи безперервність навчання. Кнопка працює коректно, виконує завершення сесії, і користувач повертається у вітальний або логін-екран.

У підсумку, застосунок забезпечує глибокий, поетапний і логічно обґрунтований шлях навчання, що відповідає сучасним вимогам до мовної освіти. Він поєднує теорію, практику, мотиваційні елементи й технічну надійність, дозволяючи кожному користувачу вчитися ефективно, комфортно й натхненно.

1.3 Огляд аналогів

Коли мова заходить за вивчення будь-якої мови кожна людина одразу згадає такий додаток як – Duolingo. Duolingo — це один із найвідоміших веб-застосунків і мобільних сервісів для вивчення іноземних мов, який базується на гейміфікованому підході до навчання. Користувачі проходять короткі уроки, виконують завдання у вигляді вправ на переклад, аудіювання, вимову, граматику та вибір правильної відповіді. Навчальний процес структуровано у вигляді дерева тем, кожна з яких містить кілька блоків завдань. Після успішного виконання користувач отримує віртуальні нагороди: серця (які зменшуються при помилках), XP-бали (які впливають на рейтинг і просування), а також «лінготи» — внутрішню валюту платформи. Duolingo пропонує зручний і захопливий спосіб вивчати мови, який підходить як для новачків, так і для тих, хто хоче вдосконалити свої навички. Завдяки продуманому режиму «істинного розуміння» можна практикувати аудіювання та вимову, використовуючи реалістичні приклади, що допомагають краще відчувати мову. Інтерфейс програми інтуїтивно зрозумілий і однаково зручний на комп’ютері чи смартфоні, що робить навчання доступним у будь-який час [4].

Одна з головних переваг Duolingo — це гейміфікація, яка перетворює навчання на цікаву гру. Щоденні цілі, змагання з іншими користувачами та нагороди за прогрес мотивують не кидати заняття. Програма пропонує величезний

вибір мов, від популярних, як іспанська чи французька, до рідкісних, наприклад, валлійської чи гавайської. Основні функції доступні безкоштовно, що робить Duolingo (рисунок 1.1) привабливим для широкої аудиторії [4].

Навчання починається з простих тем і поступово стає складнішим, що ідеально для початківців. Уроки структуровані, а повторення слів і фраз допомагає закріпити матеріал. Проте іноді приклади можуть здаватися трохи штучними, а не такими, як у реальному спілкуванні. Граматика пояснюється не завжди глибоко — часто доводиться просто запам’ятовувати фрази, а не розбиратися в правилах.

Автоматичне розпізнавання вимови — корисна функція, але воно не завжди працює ідеально, особливо якщо зв’язок слабкий або навколо шумно. Також у програмі бракує повноцінних діалогів чи можливості поспілкуватися з носіями мови, що могло б зробити навчання більш живим. Попри це, Duolingo залишається чудовим інструментом для тих, хто хоче вивчати мову у своєму темпі, поєднуючи веселощі та прогрес.

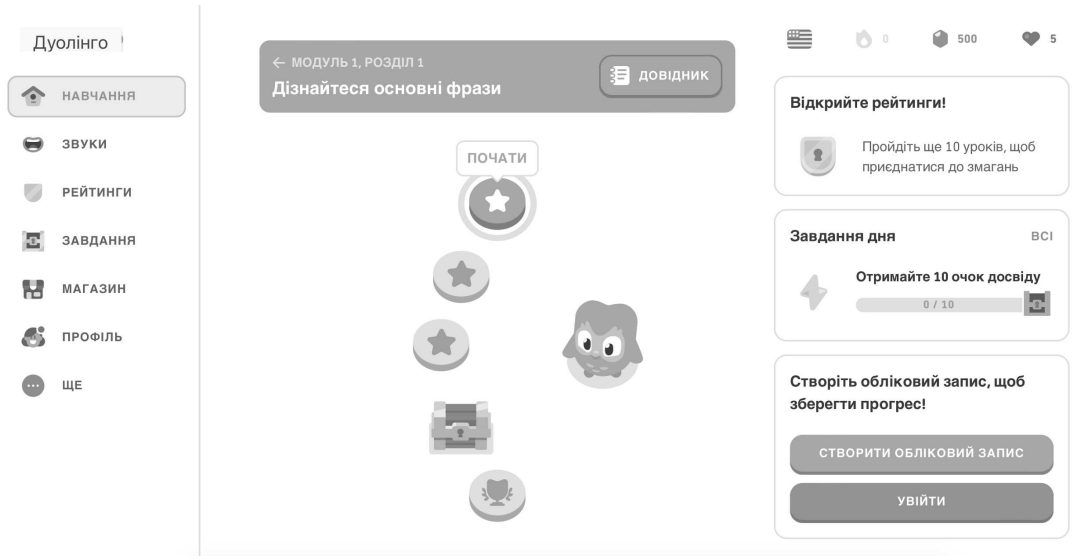


Рисунок 1.1 — Інтерфейс застосунку Duolingo

Memrise — це популярний веб-застосунок і мобільна платформа для вивчення іноземних мов, яка зосереджена на швидкому засвоєнні нової лексики та виразів за допомогою методів мнемоніки, асоціацій та активної повторюваності. Основна концепція Memrise базується на використанні карток зі словами, які супроводжуються прикладами вживання, вимовою носіїв мови, іноді

навіть короткими відео з живими діалогами. Матеріал подається у вигляді модулів, які послідовно проходить користувач, вивчаючи нові слова, а також повторюючи вивчені згідно з інтервальним алгоритмом запам’ятовування. Користувачі також можуть створювати власні курси або проходити готові, що створені як самою платформою, так і іншими учасниками. Система активно використовує візуальні підказки та інтерфейс, який мотивує до щоденної практики. Практика вимови також представлена, хоча більше у формі прослуховування правильного варіанту, ніж інтерактивної перевірки мови користувача. Memrise — це платформа, яка робить вивчення мов живим і захопливим, наближаючи користувачів до справжнього мовного середовища. Багато матеріалів містять автентичне звучання та приклади реального мовлення, що допомагає відчути мову так, ніби ти вже спілкуєшся з носіями. Основний акцент зроблено на запам’ятовування практичної лексики, стійких виразів і фраз, які легко застосувати в повсякденному житті. Граматика вводиться поступово і не перевантажує, дозволяючи зосередитися на розширенні словникового запасу [5].

Інтерфейс Memrise яскравий і зручний, він однаково приваблює як дорослих, так і молодь. Асоціації та мнемоніки значно полегшують запам’ятовування слів, роблячи процес швидшим і цікавішим. Відео з носіями мови створюють відчуття занурення, а можливість проходити курси, створені іншими користувачами, або навіть розробляти власні додає індивідуальності навчанню. Платформа адаптується до твого рівня, використовуючи інтервальні повторення, щоб закріпити знання, а мотивація підтримується завдяки рейтингам, щоденним завданням і системі досягнень.

Memrise ідеально підходить для коротких навчальних сесій — навіть кілька хвилин на день можуть бути продуктивними. Проте граматика тут пояснюється дуже поверхово, що може ускладнити створення власних речень. Вимову користувача платформа не перевіряє, тож доводиться покладатися на пасивне прослуховування. Деякі курси, створені іншими користувачами, бувають неідеальними чи містять помилки, оскільки не всі вони проходять ретельну перевірку. У безкоштовній версії доступ до частини відеоконтенту та

індивідуальних підказок обмежений, а повторення слів іноді стає одноманітним, що може знизити ентузіазм. Також бракує інтерактивних діалогів чи рольових вправ, які б допомогли відпрацювати розмовні навички.

Загалом Memrise (рисунок 1.2) є ефективним інструментом для тих, хто хоче швидко збільшити словниковий запас, вивчати живу мову, яка дійсно використовується в щоденному спілкуванні, та мати можливість повторювати вивчене з урахуванням принципів довготривалої пам'яті. Його простота, візуальна привабливість і сильна база лексичних курсів роблять його чудовим вибором для початківців і людей із середнім рівнем знань. Однак для комплексного вивчення мови його варто поєднувати з іншими платформами, де є більше уваги до граматики, говоріння та письма.

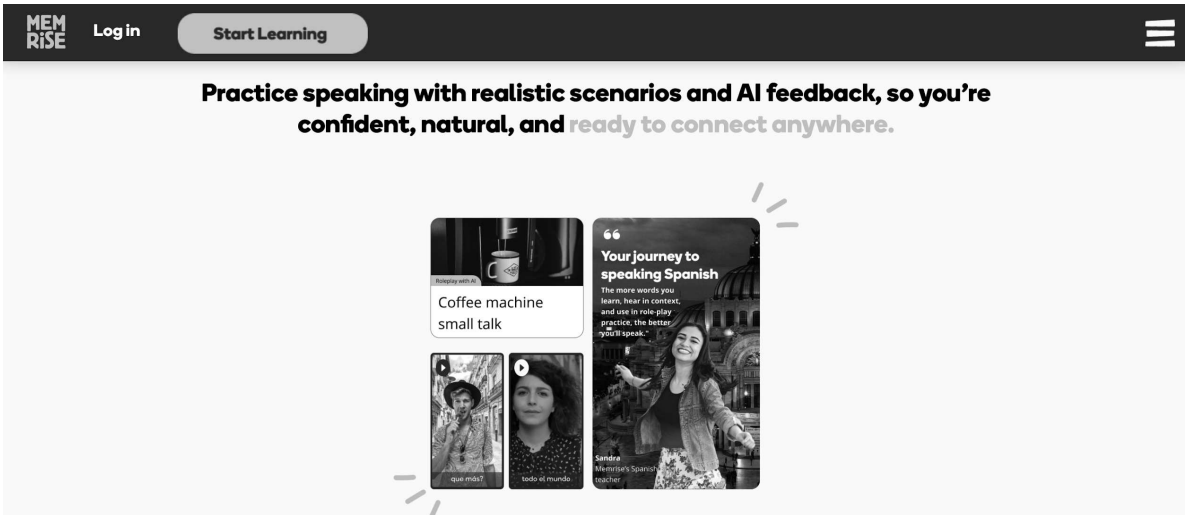


Рисунок 1.2 — Інтерфейс застосунку Memrise

Babbel — це один із провідних веб-застосунків і мобільних сервісів для вивчення іноземних мов, орієнтований переважно на дорослу аудиторію, яка хоче швидко та практично опанувати мову для реального спілкування. Основна відмінність Babbel полягає в його структурованому підході до подачі матеріалу — курси розбиті на рівні відповідно до загальноєвропейської системи CEFR (від A1 до B2), що дозволяє поступово формувати навички володіння мовою в комплексі. Навчання включає лекції з лексики, граматики, діалогів і вправ на слухання, читання та письмо. Babbel — це платформа, яка робить вивчення мов

структурованим і практичним, допомагаючи швидко застосовувати нові знання в реальному житті. Уроки побудовані так, щоб охоплювати життєві теми та діалоги, які одразу стають у пригоді. Завдяки інтелектуальній системі повторення Babbel нагадує про слова чи фрази саме тоді, коли їх потрібно освіжити в пам'яті, що робить навчання ефективним. До того ж, функція розпізнавання голосу дозволяє практикувати вимову, порівнюючи своє мовлення з еталоном, хоча в шумному середовищі вона може працювати неідеально [6].

Курси створені за участі професійних викладачів, тому матеріали якісні, а структура чітка — від простого до складного, залежно від твого рівня. Уроки тривають від 10 до 15 хвилин, що ідеально для зайнятих людей, які хочуть вчитися під час перерви чи в дорозі. Інтерактивні вправи включають письмо, аудіювання, граматику та повторення, а ти можеш обирати теми, які тобі цікаві чи актуальні. Прогрес синхронізується між пристроями, тож можна почати урок на телефоні, а закінчити на комп'ютері.

Babbel зосереджується на найпопулярніших мовах, тому вибір менший, ніж у деяких конкурентів. Безкоштовна версія досить обмежена, а повний доступ потребує передплати. На просунутому рівні (B2+) контенту може бракувати, а одноманітність вправ іноді знижує інтерес. На відміну від інших платформ, тут немає гейміфікації чи змагань, що може не підійти тим, хто любить додаткову мотивацію. Також бракує соціальної взаємодії, як-от спільнот чи парного навчання.

Babbel (рисунок 1.3) є одним із найнадійніших інструментів для системного та структурованого вивчення мови з нуля або для поглиблення базових знань. Завдяки акценту на реальні ситуації, правильній побудові навчальних модулів, якісному озвученню та можливості відпрацювання граматики платформа добре підходить для дорослих користувачів, які цінують результативність і чіткий порядок у навчанні. Тим, хто очікує більш ігрового або соціального підходу, варто доповнити Babbel іншими додатками, але як базова освітня платформа для самостійного або додаткового навчання — це один із кращих варіантів.

Рисунок 1.3 — Інтерфейс застосунку Babbel

LingQ — це веб-застосунок і мобільна платформа для вивчення іноземних мов, яка відрізняється унікальним підходом, заснованим на зануренні в автентичний контент. Основна ідея LingQ полягає в тому, що користувачі вивчають мову через читання та слухання реальних текстів, а не через традиційні вправи. Усі тексти супроводжуються аудіо, і кожне слово можна миттєво перекласти або додати до свого словника для подальшого повторення. Користувач читає або слухає реальні діалоги, статті, історії, подкасти, відео з YouTube, книги, які були адаптовані або завантажені іншими користувачами. Ключовою частиною платформи є система LingQ — коли користувач натрапляє на нове слово, воно стає «жовтим» і зберігається в особистому словнику. Надалі ці слова повторюються за системою SRS (розумне повторення), щоб з часом переходити зі статусу «нове» до «вивчене». Також користувач може бачити статистику за кількістю прочитаних слів, прослуханих годин, кількістю вивчених LingQ. LingQ дозволяє користувачам завантажувати власні тексти, аудіо чи відео, щоб створювати інтерактивні уроки. Завдяки цьому можна навчатися через улюблені книги, інтерв'ю блогерів чи професійні матеріали, роблячи процес цікавим і особистим. Платформа підтримує понад 40 мов, від англійської, німецької та французької до латини, івриту чи

суахілі. Мобільний додаток дозволяє вчитися будь-де, а офлайн-режим додає зручності [7].

Філософія LingQ — це занурення в мову через зміст, а не сухі граматичні правила. Ви можете читати автентичні тексти з миттєвим перекладом, зберігати нові слова в словник і повторювати їх за системою SRS для ефективного запам’ятовування. LingQ (рисунок 1.4) пропонує гнучкість: обирайте літературу, новини, подкасти чи відео, які вам до душі. Статистика допомагає бачити прогрес у читанні, аудіюванні та кількості вивчених слів, що мотивує рухатися далі [7].

Платформа ідеально підходить для тих, хто любить працювати з оригінальними матеріалами й готовий багато читати та слухати. Вона найкраще пасує для середнього чи просунутого рівня, хоча початківцям може бути складно через брак структурованих пояснень граматики чи інтерактивних вправ для говоріння та письма. Безкоштовна версія обмежена в перекладах, а інтерфейс може сп спочатку здаватися непростим. Для менш популярних мов контенту може бути менше, і відсутність гейміфікації чи соціальних змагань може вплинути на мотивацію.

Рисунок 1.4 — Інтерфейс застосунку LingQ

2 ВИБІР ТА ОБГРУНТУВАННЯ ІНСТРУМЕНТІВ ДЛЯ РОЗБОРКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір та огляд сховищ для зберігання даних

У сучасному світі дані стали критичним ресурсом для будь-якої організації. Вибір відповідної системи управління базами даних (СУБД) має вирішальне значення для ефективної роботи з інформацією. Системи зберігання та обробки даних розвивалися протягом десятиліть, і сьогодні існує широкий спектр рішень, що відповідають різним вимогам бізнесу та технічним потребам.

Системи управління базами даних — це програмне забезпечення, яке дозволяє створювати, змінювати та адмініструвати бази даних, виконувати пошук потрібної інформації та маніпулювати нею. СУБД забезпечують надійне зберігання даних, підтримують їх цілісність, безпеку та доступність [8].

Реляційні СУБД залишаються найпоширенішими системами для зберігання структурованих даних. Вони засновані на реляційній моделі, запропонованій Едгаром Коддом у 1970 році, де дані організовані в таблиці з рядками та стовпцями, а між таблицями встановлюються зв'язки.

Oracle Database відзначається надзвичайно високою надійністю та стабільністю під навантаженням, що робить її ідеальним вибором для великих корпорацій з критично важливими даними. Потужні можливості обробки транзакцій та підтримки цілісності даних забезпечують безпеку інформації навіть у найскладніших умовах. Система пропонує розширені інструменти безпеки та аудиту, які відповідають найсуворішим галузевим стандартам. Багатий набір корпоративних функцій, включаючи високу доступність, шардинг та кластеризацію, дозволяє створювати надійні масштабовані рішення. Професійна підтримка та регулярні оновлення гарантують актуальність системи. Oracle оптимізована для складних аналітичних запитів та великих обсягів даних.

Однак Oracle має суттєві недоліки: висока вартість ліцензування та обслуговування створює значний фінансовий бар'єр для малого та середнього бізнесу. Складність налаштування та адміністрування вимагає висококваліфікованих спеціалістів, що підвищує загальну вартість володіння.

Система достатньо ресурсомістка і потребує потужного апаратного забезпечення, а надмірна функціональність може бути надлишковою для простіших проектів [9].

MySQL завоювала популярність завдяки своїй простоті встановлення та використання, що робить її доступною навіть для початківців. Як система з відкритим кодом, вона пропонує безкоштовне використання для більшості випадків та має величезну спільноту підтримки. MySQL добре інтегрується з поширеними веб-технологіями, особливо з PHP, що зробило її стандартом для веб-розробки. Вона демонструє хорошу продуктивність для операцій читання та підтримує різноманітні типи сховищ даних, дозволяючи налаштувати систему під конкретні потреби.

Проте MySQL має свої обмеження: менша функціональність порівняно з Oracle або PostgreSQL, особливо щодо складних транзакцій та тригерів. Після придбання компанією Oracle з'явилися побоювання щодо майбутнього вільної версії. Система показує не найкращу продуктивність при дуже високих навантаженнях та складних запитах, а також має обмеження щодо масштабованості для надвеликих систем.

PostgreSQL виділяється повною відповідністю стандартам SQL та підтримкою розширеної об'єктно-реляційної моделі даних. Вона пропонує найбільшу розширюваність серед відкритих СУБД, дозволяючи створювати власні типи даних, функції та процедури. Особлива увага приділяється цілісності даних та коректності транзакцій, що робить PostgreSQL надійною основою для критично важливих додатків. Система демонструє відмінну продуктивність для складних запитів та аналітики, підтримує паралельне виконання запитів та має вбудовані можливості для мультиверсійності [10].

До недоліків PostgreSQL можна віднести дещо нижчу швидкість простих операцій порівняно з MySQL, більшу споживаність ресурсів та складніший процес налаштування та оптимізації. База даних потребує кваліфікованого адміністрування для розкриття всього потенціалу та має менш розвинену екосистему комерційних інструментів підтримки порівняно з пропрієтарними рішеннями.

Microsoft SQL Server відрізняється чудовою інтеграцією з екосистемою Microsoft, включаючи .NET, Azure та інші сервіси компанії. Система пропонує зручні інструменти для розробки та адміністрування, зокрема Management Studio з інтуїтивним інтерфейсом. SQL Server включає потужні інструменти бізнес-аналітики та звітності, такі як Analysis Services та Reporting Services. Система забезпечує високу продуктивність для змішаних навантажень та має добре опрацьовані механізми резервного копіювання та відновлення.

Серед недоліків можна виділити прив'язку переважно до платформи Windows, що обмежує гнучкість розгортання, а також високу вартість ліцензування для корпоративних версій. SQL Server потребує значних апаратних ресурсів для оптимальної роботи та має менші можливості для кастомізації порівняно з PostgreSQL.

MongoDB революціонізувала підхід до зберігання даних завдяки своїй документо-орієнтованій структурі, яка дозволяє зберігати складні ієрархічні дані без нормалізації. Система пропонує гнучку схему даних, що дозволяє змінювати структуру "на льоту" без простоїв. MongoDB забезпечує горизонтальне масштабування через шардинг та реплікацію, що робить її ідеальною для додатків з високим навантаженням. База даних має вбудовану підтримку географічних даних та індексів, а також зручний запит через API для багатьох мов програмування [11].

До слабких сторін MongoDB належать менші можливості для складних об'єднань таблиць та транзакцій порівняно з реляційними базами даних, особливо в ранніх версіях. Система споживає більше дискового простору через зберігання дублікатів даних (денормалізація) та вимагає ретельного проектування схеми для ефективної роботи. MongoDB також має обмеження атомарності операцій на рівні кількох документів.

Redis вирізняється надзвичайно високою швидкістю роботи завдяки зберіганню даних в оперативній пам'яті, що забезпечує мікросекундний час відгуку. Система підтримує різноманітні структури даних: рядки, хеші, списки, множини, впорядковані множини та інші спеціалізовані типи. Redis має вбудовану

публікацію/підписку, транзакції та скриптування на Lua. Система забезпечує персистентність даних через знімки та журналювання змін, а також підтримує реплікацію та автоматичне відновлення після збоїв.

Проте Redis має обмеження: розмір бази даних обмежений обсягом оперативної пам'яті, що підвищує вартість інфраструктури. Система має обмежені можливості для складних запитів та аналітики порівняно з традиційними СУБД, а також обмежену підтримку транзакцій між кількома ключами. Redis краще використовувати як допоміжне сховище для кешування або обміну повідомленнями, а не як основну базу даних. Amazon DynamoDB відзначається повністю керованою інфраструктурою без необхідності адміністрування серверів, що дозволяє розробникам зосередитись на створенні додатків. Система забезпечує автоматичне масштабування відповідно до навантаження та гарантовану продуктивність з малим часом відгуку.

Серед недоліків DynamoDB можна виділити обмежену гнучкість запитів, зокрема складність виконання складних об'єднань та агрегацій. Система має жорсткі обмеження на розмір документів та кількість індексів, а також потенційно високу вартість при великих обсягах даних та інтенсивному читанні/записі. Прив'язка до екосистеми AWS створює залежність від одного постачальника та ускладнює міграцію на інші платформи.

Таблиця 2.1 — Порівняння продуктивності та масштабованості

СУБД	Операції читання	Операції запису	Горизонтальне масштабування	Вертикальне масштабування
MySQL	Дуже висока	Середня	Обмежена	Висока
Oracle	Висока	Висока	Висока	Дуже висока
PostgreSQL	Дуже висока	Висока	Висока	Дуже висока
MongoDB	Висока	Середня	Дуже висока	Дуже висока
SQL Server	Висока	Дуже висока	Середня	Висока
Redis	Надзвичайна	Надзвичайна	Середня	Обмежена

Документо-орієнтовані системи управління базами даних представляють окремий клас NoSQL рішень, які принципово відрізняються від традиційних реляційних систем. На відміну від реляційної моделі з фіксованими таблицями та

схемами, документо-орієнтовані СУБД (рисунок 2.1) зберігають дані у гнучких структурах, що називаються документами.

Також можемо порівняти популярність баз даних, опираючись на інформацію на просторах інтернету, яку можна згрупувати в відповідні графіки.

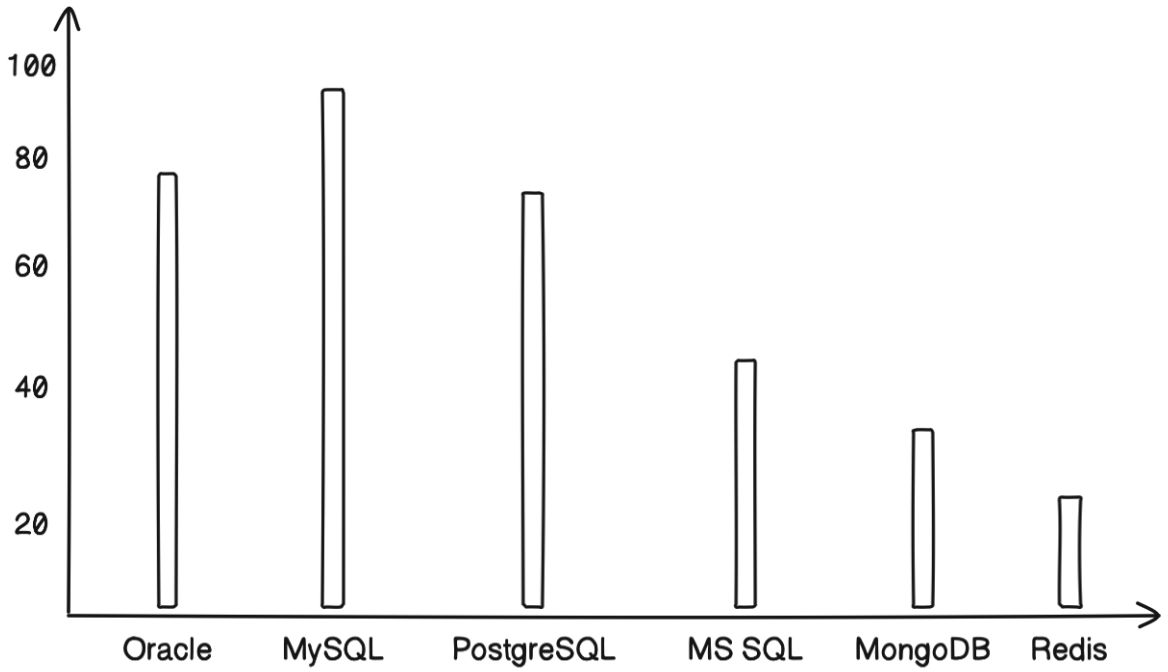


Рисунок 2.1 — Діаграма популярності СУБД

Основною концепцією таких систем є поняття документа, який зазвичай представлено у форматі, подібному до JSON або BSON (бінарний JSON). Кожен документ являє собою самодостатню одиницю даних, що містить всю необхідну інформацію без необхідності звертатися до інших таблиць. Документи можуть мати складну ієрархічну структуру з вкладеними об'єктами та масивами, причому документи в одній колекції (аналог таблиці в реляційних СУБД) можуть мати різну структуру.

Ключовою перевагою документо-орієнтованих баз даних є їхня гнучкість. Розробники можуть змінювати структуру документів "на льоту", не перебудовуючи всю базу даних. Це особливо корисно для проектів з швидкою ітерацією, де вимоги до даних постійно еволюціонують. Така гнучкість також спрощує роботу з неструктурованими або напівструктурованими даними.

Документні бази даних забезпечують природне відображення об'єктів з об'єктно-орієнтованого програмування на документи в базі даних, що скорочує "об'єктно-реляційний розрив". Розробники можуть зберігати об'єкти так, як вони представлені в коді, без складного перетворення в реляційну модель, що значно прискорює розробку.

Горизонтальна масштабованість є ще однією сильною стороною документних СУБД. Вони зазвичай проектуються з урахуванням розподіленої архітектури, що дозволяє легко додавати нові сервери для розподілу навантаження. Ця властивість робить їх привабливими для великих веб-додатків з високою інтенсивністю операцій читання/запису.

Найвідомішим представником документо-орієнтованих СУБД є MongoDB, яка поєднує гнучкість документної моделі з багатими можливостями для запитів, індексування та аналітики. MongoDB використовується багатьма компаніями, від стартапів до великих підприємств, для побудови масштабованих веб-додатків, систем управління контентом та аналітичних платформ.

Поряд з MongoDB існують й інші відомі документні СУБД. CouchDB відрізняється своєю моделлю реплікації на основі REST API та підтримкою офлайн-роботи. Apache Couchbase поєднує документну модель з кешуванням в оперативній пам'яті для досягнення високої продуктивності. RavenDB фокусується на транзакційній цілісності та зручності використання у середовищі .NET. Amazon DocumentDB пропонує сумісність з MongoDB API як керований сервіс на платформі AWS [12].

Незважаючи на численні переваги, документо-орієнтовані СУБД мають певні обмеження. Вони менш ефективні для складних агрегацій та об'єднань даних порівняно з реляційними системами. Бази даних цього типу також можуть споживати більше дискового простору через дублювання даних та відсутність нормалізації. У деяких рішеннях може бути обмежена підтримка транзакцій, особливо для операцій, що охоплюють кілька документів.

Документо-орієнтовані бази даних (рисунок2.2) найкраще підходять для сценаріїв, де важлива гнучкість схеми даних, наприклад, для управління

контентом, каталогів продуктів, профілів користувачів та аналітичних систем. Вони також ефективні для обробки великих обсягів даних з високою швидкістю вставки та оновлення, а також у випадках, коли дані мають ієрархічну структуру.

Сучасні системи, такі як MongoDB, постійно еволюціонують, додаючи функціональність, яка традиційно асоціювалася з реляційними базами даних, включаючи підтримку транзакцій ACID та складні агрегаційні операції. Це розмиває межі між різними типами баз даних та дозволяє документо-орієнтованим системам вирішувати ширше коло задач.

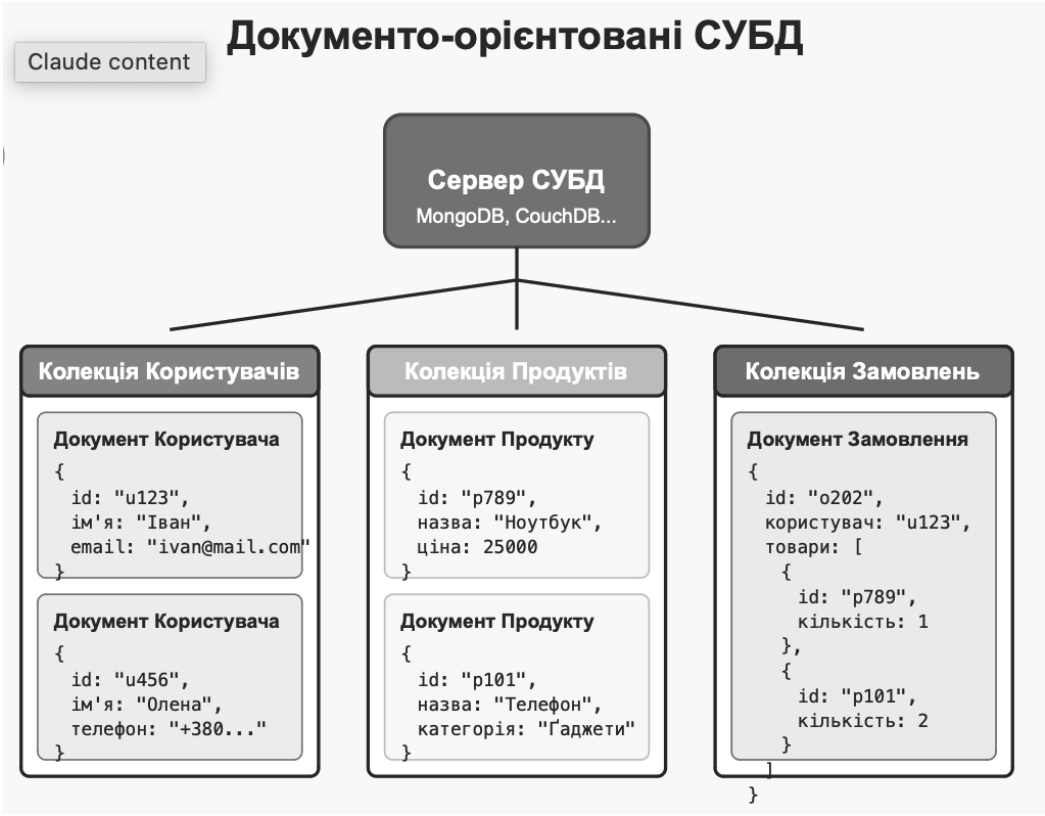


Рисунок 2.2 — Принцип зберігання у документо-орієнтовані СУБД

Сховища типу "ключ-значення" є одним із найпростіших і найшвидших способів зберігання даних у базах даних. Вони засновані на ідеї збереження інформації у вигляді пари: унікальний ключ і відповідне йому значення. Такий підхід схожий на використання словника або хеш-таблиці в програмуванні. Ключ виконує роль ідентифікатора, який дозволяє швидко знайти або змінити пов’язане з ним значення без необхідності аналізу всієї бази даних. Це робить такі сховища особливо ефективними в системах, де потрібна висока швидкодія та простота

доступу до даних, наприклад, у кешуванні, обробці сесій користувачів або зберіганні налаштувань.

У порівнянні з реляційними базами даних, де структура даних складніша і пов’язана таблицями, сховища ключ-значення не мають фіксованої схеми. Значенням може бути простий текст, число, JSON, або навіть бінарні дані. Такий підхід дає гнучкість, але водночас накладає обмеження: не можна легко виконувати складні запити або фільтрувати значення без знання ключів. Однак це не заважає їх популярності — такі системи як Redis, Amazon DynamoDB або Memcached широко використовуються у веб-розробці, мобільних додатках і масштабованих системах.

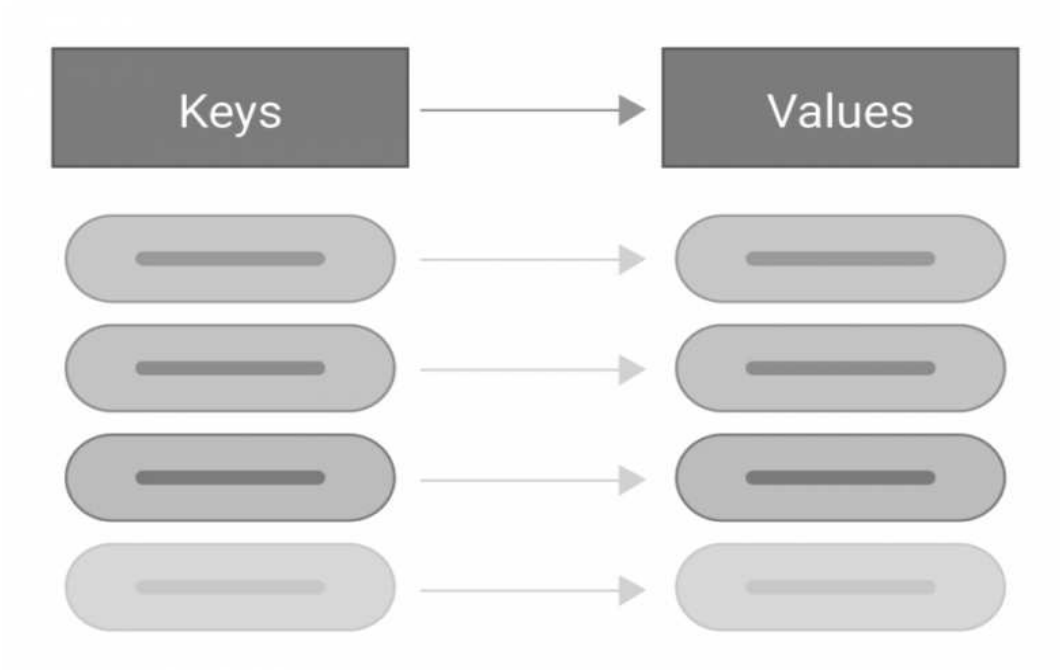


Рисунок 2.3 — Принцип роботи ключ-значення

Переваги реляційних баз (рисунок2.3) включають чітку структуру даних, що полегшує моделювання зв’язків, стандартизовану мову SQL і широку підтримку інструментів для адміністрування. Однак їхня масштабованість обмежена при обробці великих обсягів неструктурованих даних. Горизонтальне масштабування (додавання серверів) ускладнене через необхідність синхронізації даних між вузлами. Крім того, реляційні бази можуть бути менш ефективними для застосунків із високою частотою читання/запису, таких як соціальні мережі.

Нереляційні бази даних (таблиця 2.2), або NoSQL, з’явилися як відповідь на обмеження реляційних систем у контексті великих даних і розподілених архітектур. Вони не використовують табличну структуру й підтримують різноманітні моделі даних, зокрема:

— ключ-значення, прості бази, в яких дані зберігаються у вигляді пари ключ-значення (рис.);

— документо-орієнтовані, збереження даних в яких, відбувається у вигляді JSON- або BSON-документів (рис.);

— колонкові, бази які оптимізовано для аналітичних запитів на великих обсягах даних (рис.);

— графові, які призначені для роботи зі складними зв’язками (рис.).

Таблиця 2.2 — Порівняння PostgresSQL з іншими реляційними СУБД

Характеристика	PostgreSQL	MySQL	SQLite	SQL Server	MongoDB
Тип бази	Серверна	Серверна	Вбудована	Серверна	Серверна
Модель консистентності	ACID	ACID	ACID	ACID	ACID
Підтримка JSON	JSONB	JSON	Обмежена	JSON	JSON
Складні запити	Висока	Середня	Низька	Висока	Висока
Вартість	Безкоштовна	Безкоштовна	Безкоштовна	Платна	Безкоштовна
Кросплатформність	Висока	Висока	Висока	Середня	Висока
Масштабованість	Помірна	Помірна	Низька	Помірна	Помірна
Паралельний доступ	Високий	Високий	Низький	Високий	Високий

2.2 Огляд сучасних фреймворків для написання веб застосунків

Backend-фреймворки є важливими інструментами для розробників, оскільки вони надають структурований підхід до створення серверної частини веб-додатків. Вони містять набір готових рішень та інструментів, які спрощують розробку, забезпечують кращу організацію коду та підвищують продуктивність. Замість того, щоб починати кожен проєкт з нуля, розробники можуть використовувати функціональність, яку пропонують фреймворки, та зосередитися на унікальних аспектах свого застосунку.

Існує велика кількість backend-фреймворків, кожен з яких має свої особливості, переваги та недоліки. Наприклад, Python-фреймворки, такі як Django та Flask, користуються популярністю завдяки своїй простоті, гнучкості та великій кількості доступних бібліотек. Django є повноцінним фреймворком, який надає "з коробки" багато необхідних компонентів, таких як ORM (Object-Relational Mapping) для роботи з базами даних, адміністративна панель та система шаблонів. Це робить його чудовим вибором для швидкої розробки складних веб-застосунків. Flask, навпаки, є мікрофреймворком, який надає лише базові інструменти, дозволяючи розробникам вибирати та інтегрувати необхідні бібліотеки на свій розсуд. Це дає більшу гнучкість, але може вимагати більше зусиль для налаштування складних проєктів. Django ORM дозволяє працювати з базами даних через Python-об’єкти. Він підтримує PostgreSQL та інші СУБД, забезпечуючи зручний синтаксис для запитів і автоматичне створення міграцій. Django ORM є інтуїтивним для Python-розробників, але менш гнучким порівняно з EF Core, особливо в складних реляційних сценаріях. Його продуктивність нижча через накладні витрати Python [14].

Spring Boot — це фреймворк для мови Java, що спрощує розробку веб-застосунків на основі екосистеми Spring. Він пропонує автоматичну конфігурацію, вбудований сервер і підтримку REST API, MVC і мікросервісів. Spring Boot інтегрується з Hibernate, популярним ORM для Java, що забезпечує ефективну роботу з базами даних. Фреймворк підходить для великих корпоративних систем завдяки своїй масштабованій архітектурі та підтримці

розподілених систем. Проте складність конфігурації та великий час запуску можуть бути недоліками для невеликих проєктів. Крім того, Java має високі вимоги до пам’яті, що може впливати на витрати в хмарних середовищах.

Hibernate використовується в Spring Boot (таблиця 2.3). Він пропонує потужні можливості для відображення складних реляційних структур і підтримки транзакцій. Hibernate підтримує PostgreSQL і забезпечує гнучкість у налаштуванні через анотації або XML.

Express.js — це мінімалістичний фреймворк для Node.js, що використовується для створення REST API та легковагових веб-додатків. Він пропонує гнучкість у виборі бібліотек і інструментів, що дозволяє розробникам налаштовувати стек відповідно до потреб. Express.js працює в асинхронному режимі, що забезпечує високу швидкість обробки запитів, особливо для застосунків із великою кількістю I/O-операцій. Однак його мінімалізм вимагає додаткових бібліотек для таких завдань, як аутентифікація чи робота з базами даних. Це може ускладнювати розробку складних систем, а слабка типізація JavaScript підвищує ризик помилок у великих проєктах.

Таблиця 2.3 — Порівняння фреймворків для веб-серверних застосунків

Мова програмування	Java	C#	Python	JavaScript
Продуктивність	Висока	Висока	Середня	Висока (легкі запити)
Швидкість розробки	Висока	Середня	Дуже висока	Середня
Рівень безпеки	Висока	Висока	Висока	Середня
Масштабованість	Висока	Висока	Середня	Середня
Інтеграція з БД	Hibernate	EF Core	Django ORM	Sequelize
Кросплатформність	Висока	Висока	Висока	Висока

Однією з ключових переваг Spring Boot є його легкість налаштування та запуску. Він значно спрощує процес ініціалізації нового проєкту завдяки своїй функції автоконфігурації. Spring Boot автоматично налаштовує більшість

необхідних компонентів, зменшуючи обсяг ручної конфігурації, яка потрібна в інших фреймворках, таких як Flask або Express.js. Це дозволяє розробникам швидше розпочати роботу та зосередитися на бізнес-логіці.

Spring Boot також пропонує комплексну екосистему Spring Framework, яка включає безліч готових рішень для різних завдань, таких як керування залежностями, безпека, робота з базами даних (через Spring Data JPA), обробка черг повідомлень та багато іншого. Хоча Django також є повноцінним фреймворком, екосистема Spring є надзвичайно широкою та зрілою, особливо в корпоративному середовищі.

Ще однією важливою перевагою є вбудовані можливості для створення мікросервісів. Spring Cloud, частина екосистеми Spring, надає потужні інструменти для побудови розподілених систем, що робить Spring Boot чудовим вибором для сучасних архітектур. Хоча інші фреймворки також можуть використовуватися для мікросервісів, Spring Boot пропонує більш інтегрований та комплексний набір інструментів для цього.

Spring Boot також відзначається своєю продуктивністю та масштабованістю, особливо в порівнянні з фреймворками, що працюють на інтерпретованих мовах, таких як Python (Django, Flask) або Ruby (Ruby on Rails). Java, на якій побудований Spring Boot, є компільованою мовою, що зазвичай призводить до кращої продуктивності в умовах високого навантаження.

2.3 Огляд сучасних систем для розробки

IDE (Integrated Development Environment) є важливим інструментом для розробників, що об'єднує в собі редактор коду, компілятор або інтерпретатор, налагоджувач та інші корисні інструменти для спрощення процесу розробки. Visual Studio Code вирізняється своєю легкістю та швидкістю роботи, а також кросплатформністю, адже доступний на Windows, macOS та Linux. Однією з його ключових переваг є величезна екосистема розширень, які додають підтримку різноманітних мов програмування та інструментів, а вбудований термінал та потужна інтеграція з Git значно полегшують робочий процес. Проте, порівняно з

більш комплексними IDE, VS Code може вимагати встановлення додаткових розширень для отримання повної функціональності, а велика кількість встановлених розширень іноді може впливати на продуктивність.

Інтегровані середовища розробки, зокрема продукти JetBrains (рисунок 2.4) (IntelliJ IDEA, PyCharm, Rider, CLion) та Microsoft Visual Studio, пропонують найбільш повний набір інструментів. Вони надають розширену підтримку рефакторингу, статичного аналізу коду, профілювання, налагодження в реальному часі, управління залежностями, CI/CD інтеграції, запуску юніт-тестів та іншого.

Висока щільність функціоналу робить ці середовища оптимальними для розробки складних систем: корпоративного програмного забезпечення, багаторівневих додатків, мікросервісної архітектури. Проте така потужність супроводжується значним споживанням ресурсів, що обмежує ефективність на малопотужних пристроях. Також вартість комерційних IDE може бути значною, хоча багато продуктів мають безкоштовні освітні чи open-source ліцензії.

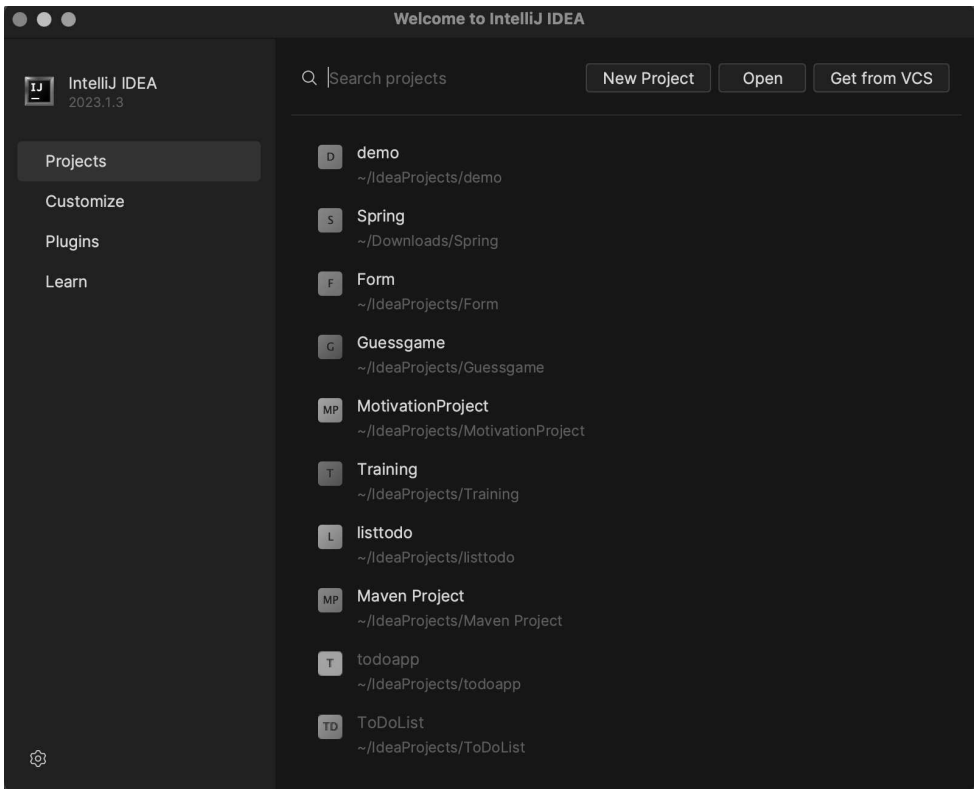


Рисунок 2.4 — JetBrains IntelliJ IDEA

Visual Studio Code (рисунок 2.5), розроблений Microsoft, став де-факто стандартом легких, але потужних редакторів. Підтримка LSP, багатотисячна екосистема розширень роблять його універсальним інструментом для широкого спектра завдань. Через своє походження він ідеально підходить для веброзробки, розробки на JavaScript, Python, Go, Rust та C#.

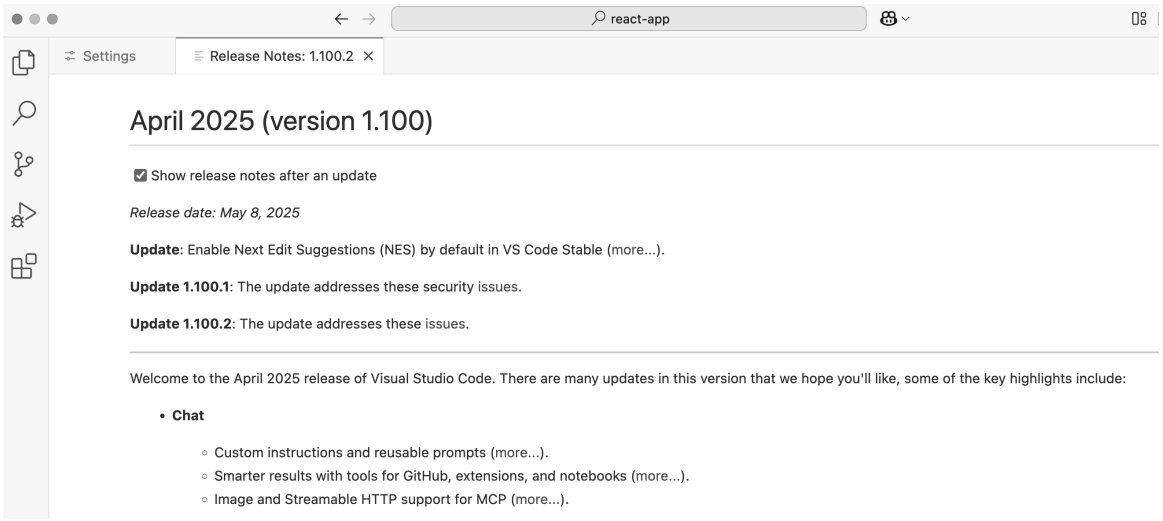


Рисунок 2.5 — Текстовий редактор Visual Studio Code

Sublime Text (рисунок 2.6) — ще один приклад високопродуктивного редактора, що демонструє виняткову швидкодію навіть на старому обладнанні. Проте функціонал за замовчуванням обмежений, і ефективне використання потребує встановлення численних плагінів. Atom, який з 2022 року більше не підтримується GitHub, історично мав схожу роль, але поступився місцем VSC.

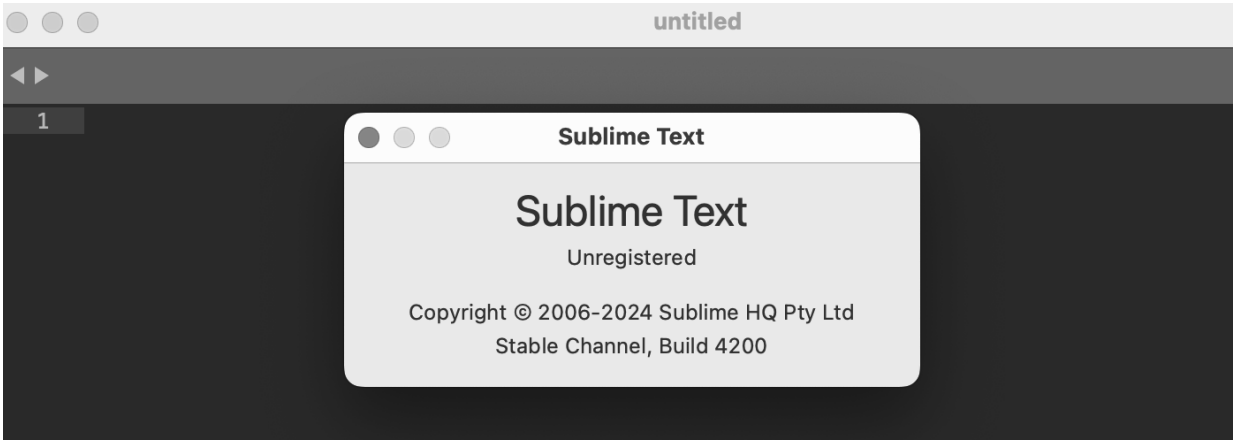


Рисунок 2.6 — Текстовий редактор Sublime Text

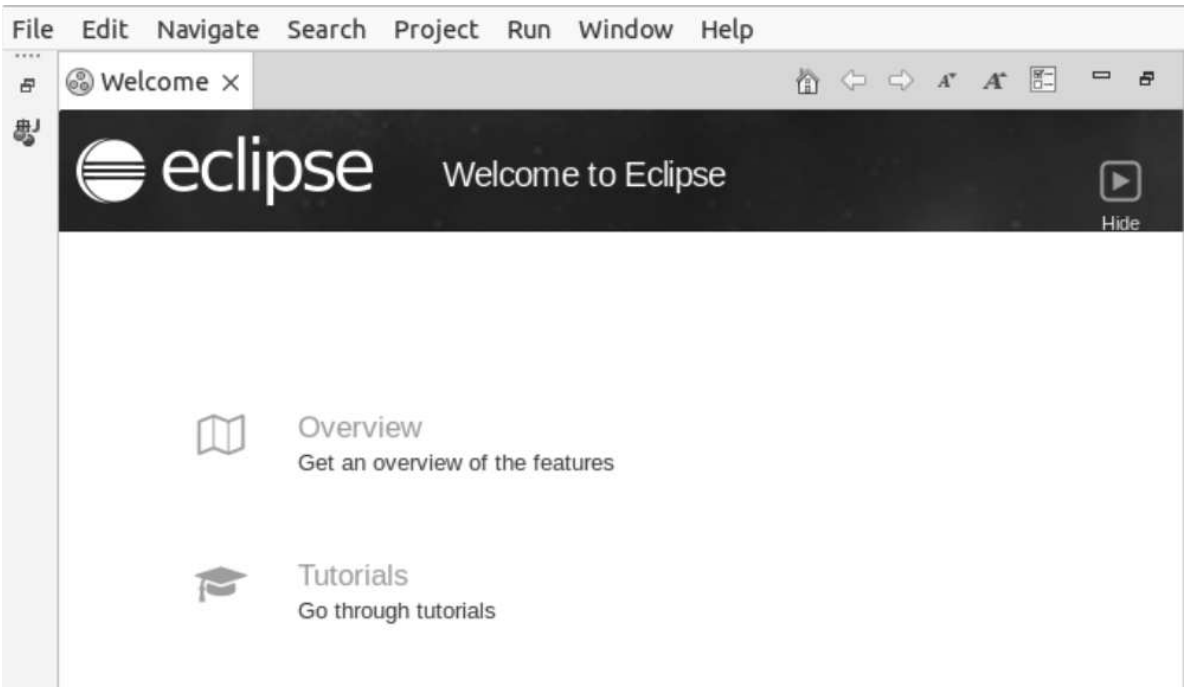


Рисунок 2.7 — Текстовий редактор Eclipse

IntelliJ IDEA є ідеальним вибором завдяки своїй неперевершеній інтелектуальній підтримці коду. Її можливості автодоповнення, аналізу коду та рефакторингу значно підвищують продуктивність розробника та допомагають підтримувати чистоту та якість кодової бази, що особливо важливо для великих та складних проєктів. Крім того, широка підтримка різноманітних технологій та фреймворків, ймовірно необхідних у сучасному проєкті, дозволяє працювати з усіма аспектами розробки в одному середовищі без необхідності постійного перемикання між різними інструментами. Потужні інструменти налагодження та профілювання в IntelliJ IDEA є незамінними для виявлення та усунення помилок, а також для оптимізації продуктивності застосунку. Зручна інтеграція з системами контролю версій, базами даних та серверами застосунків робить робочий процес більш ефективним та організованим.

Окрім стандартних функцій, середовище забезпечує глибоку інтеграцію з Maven, Gradle, Docker, Kubernetes, Spring Boot, Jakarta EE, JavaFX, Kotlin, Scala, Android та багатьма іншими інструментами і технологіями, що дозволяє не тільки писати код, а й ефективно будувати, тестувати, розгортати та підтримувати складні системи. Вбудований аналізатор якості коду дозволяє знаходити

потенційні проблеми ще до виконання програми, а інструменти для написання юніт-тестів і робота з фреймворками на кшталт JUnit, TestNG чи Spock полегшують створення надійних тестових сценаріїв.

Завдяки функціям живого шаблонування, швидких дій, навігації по великому коду, автоматичного імпорту залежностей та синхронізації з репозиторіями, IntelliJ (таблиця 2.4) забезпечує максимально комфортне середовище розробки навіть для найвибагливіших програмістів. Постійне оновлення, активна спільнота, велика кількість плагінів і документації роблять це середовище не тільки функціонально потужним, але й дуже гнучким у налаштуванні під індивідуальні потреби. Хоча Ultimate версія є платною та ресурсомісткою, її переваги в плані продуктивності, підтримки корпоративних фреймворків, засобів аналізу, інструментів роботи з вебтехнологіями, REST API, GraphQL, JavaScript, TypeScript, CSS та HTML значною мірою перекривають витрати, роблячи її найкращим вибором для серйозної розробки в професійному середовищі.

Таблиця 2.4 — Порівняння популярних редакторів

Характеристика	Visual Studio Code	IntelliJ IDEA	Eclipse	Sublime Text
Тип середовища	Легкий редактор коду з можливаостями IDE через розширення	Повноцінне IDE, широка підтримка, особливо Java	Повноцінне IDE, розширювана архітектура через плагіни	Легкий текстовий редактор з можливостями IDE через плагіни
Швидкодія запуску	Дуже швидка	Середня	Середня	Дуже швидка
Підтримка LSP	Відмінна	Хороша	Хороша	Хороша
Можливості для командної розробки	Хороша інтеграція з Git	Відмінна інтеграція з системами контролю версій, потужні інструменти для навігації по коду	Хороша інтеграція з Git	Базова інтеграція з системами контролю версій
Ступінь налаштовуваності	Висока	Дуже висока	Дуже висока	Висока

3 РОЗРОБКА ВЕБ ЗАСТОСУНКУ

3.1 Розробка структури бази даних

Оскільки для розробки було обрано базу даних PostgreSQL (psql), це відкриває ряд потужних можливостей для зберігання та обробки даних. Її гнучкість, масштабованість і підтримка ACID-гарантій робить її чудовим вибором для сучасних веб-застосунків. Також PostgreSQL добре інтегрується з фреймворками на базі Spring, що спрощує роботу з даними через ORM-технології, зокрема JPA та Hibernate.

У наведеному коді представлено клас UserEntity, який є частиною пакету com.login.demo.entity. Цей клас позначено анотацією @Entity, що означає — він є частиною об'єктно-реляційного відображення і буде відображений у відповідну таблицю в базі даних. Завдяки анотації @Table(name = "userEntity"), явно вказується, що назва таблиці у базі буде userEntity, хоча за замовчуванням назва могла б генеруватись з назви класу.

```
@Table(name = "userEntity")
@AllArgsConstructor
@NoArgsConstructor
@ToString
@Entity
@Getter
@Setter
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    private String password;
    private ERole role = ERole.USER_ROLE;
}
```

Основні поля сутності — id, name, email, password, role — відповідають стовпцям у таблиці бази даних. Поле id є первинним ключем, про що свідчить анотація @Id, і воно автоматично генерується за стратегією

GenerationType.IDENTITY. Це означає, що PostgreSQL буде самостійно інкрементувати це значення, зазвичай використовуючи послідовність.

Поля name, email, password є простими текстовими даними типу String, які будуть представлені у базі як стовпці типу VARCHAR або TEXT (залежно від налаштувань JPA або міграційних скриптів).

Окремо слід звернути увагу на поле role, яке має тип ERole. Це ймовірно enum, що визначає тип ролі користувача (наприклад, USER_ROLE, ADMIN_ROLE тощо). За замовчуванням йому надано значення ERole.USER_ROLE, тобто при створенні нового користувача, якщо явно не вказати інше, буде встановлено стандартну роль. У PostgreSQL enum зазвичай зберігається або як VARCHAR, або як окремий тип (якщо явно створити тип ENUM в самій БД). У Spring JPA для коректної роботи з enum потрібно використовувати анотацію @Enumerated, яка тут відсутня, тому тип enum, ймовірно, буде автоматично збережений як рядок.

```
@Entity
@Table(name = "user_answers")
public class UserAnswer {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @Column(nullable = false)
   private String username;
   @Column(nullable = false)
   private String lessonId;
   @Column(nullable = false)
   private String questionId;
   @Column(nullable = false)
   private String userAnswer;
   @Column(nullable = false)
   private boolean isCorrect;
   @Column(nullable = false)
   private double score;
   public Long getId() {
      return id;
   }
   public void setId(Long id) {
      this.id = id;
```

Початкове поле id є первинним ключем і генерується автоматично при додаванні нових записів за допомогою стратегії GenerationType.IDENTITY. Це означає, що PostgreSQL буде створювати унікальні значення, найчастіше через вбудовану послідовність (sequence).

Поле username є текстовим і вказує, яка саме людина (або обліковий запис) дала відповідь. Його тип у базі — TEXT або VARCHAR, анотація @Column(nullable = false) вимагає, щоб це поле не було порожнім.

Поля lessonId і questionId також є обов'язковими і представляють відповідно ідентифікатор уроку та запитання, на яке була надана відповідь. У поточній реалізації вони збережені як String, хоча логічно було б зробити їх зовнішніми ключами на таблиці уроків і запитань відповідно — це дозволило б краще підтримувати зв'язки між таблицями, забезпечити цілісність і спростити запити при аналітиці.

Поле isCorrect зберігає логічне значення: чи правильна відповідь. Тип boolean автоматично транслюється у PostgreSQL у тип BOOLEAN.

Поле score представляє кількість балів, яку користувач отримав за відповідь. Тип double транслюється у PostgreSQL як DOUBLE PRECISION, що дозволяє зберігати дробові значення з високою точністю.

І остання сутніть — клас LessonProgress, який пов’язаний з таблицею lesson_progress у базі даних. Клас позначено анотаціями @Entity та @Table, де додатково вказано унікальне обмеження — комбінація username та lessonId повинна бути унікальною. Це означає, що кожен користувач може мати лише один запис про прогрес для кожного окремого уроку, що є логічним для відстеження результатів.

```
public void setTotalQuestions(int totalQuestions) {
    this.totalQuestions = totalQuestions;
}
public int getProgressPercentage() {
    return progressPercentage;
}
public void setProgressPercentage(int progressPercentage) {
    this.progressPercentage = progressPercentage;
```

```
    }
    public void calculatePercentage() {
        if (totalQuestions > 0) {
            this.progressPercentage = (int) (((double) correctAnswers / totalQuestions)
* 100);
        } else {
            this.progressPercentage = 0;
        }
```

Поле id є первинним ключем і автоматично генерується PostgreSQL через стратегія GenerationType.IDENTITY, що створює унікальні цілі числа.

username та lessonId — це текстові обов’язкові поля. Перше визначає, який саме користувач виконує урок, друге — ідентифікатор конкретного уроку. Зберігаються як TEXT або VARCHAR, і можуть бути пов'язані з таблицею користувачів і таблицею уроків (через зовнішні ключі, якщо буде розширення структури).

Поля correctAnswers, totalQuestions і progressPercentage мають цілочисельний тип і зберігають статистику по конкретному уроку: кількість правильних відповідей, загальну кількість запитань і відсоток проходження. Поле progressPercentage оновлюється через метод calculatePercentage(), який динамічно обчислює відсоток на основі правильних відповідей. Якщо totalQuestions > 0, то відсоток визначається як (правильні / загальна кількість) * 100, і округлюється до цілого числа. Якщо кількість запитань дорівнює нулю, встановлюється нульовий прогрес, щоб уникнути ділення на нуль.

3.2 Огляд основної логіки

Основна логіка веб застосунку для допомоги у вивченні іноземної мови полягає у таких пунктах: — реєстраційна форма з багатьма видами безпеки шифрування та комфортного входження в акаунт, — головна сторінка, на якій реалізовано різноманітний контент від слайдерів до списків і текстових рекомендацій, — каталог з відповідними навчальними розділами, — блок тестування для кожного розділу.

Після логіна користувача перенаправляє на головну сторінку — це своєрідна "карта рівнів" у стилі гри, яка реалізована або через div-блоки, що

стилізовані як кнопки, або через акордеон (файл main.html). Ця сторінка динамічно заповнюється залежно від авторизованого користувача, зокрема з використанням ${#authentication.getPrincipal().getName()} для вітання, тобто Spring Security забезпечує передачу даних про сесію.

Візуально головна сторінка має фон із частинками (particles.js) і напівпрозорий контейнер (з CSS-файлу, styles.css), у якому розміщено інтерактивні блоки або акордеон із лекціями. При натисканні на рівень або лекцію, користувача перенаправляє на відповідну сторінку з вмістом лекції (окремий HTML або Thymeleaf шаблон lecture1.html). Там можна розмістити теоретичний матеріал і кнопку для початку тесту, яка веде на іншу сторінку або відкриває інтерактивний компонент.

Щоб реалізувати логіку роздільного доступу (наприклад, щоб користувач не бачив рівні без авторизації), використовуються Spring Security правила (у файлі SecurityConfig або WebSecurityConfigurerAdapter). Вони визначають, які сторінки доступні всім, а які — лише залогіненим. Дані про користувача беруться з бази, яка підключена через JPA (файл UserRepository).

Навчальні розділи й тестування реалізовано як частину інтерактивного навчального процесу, що поєднує теорію з практикою безпосередньо на одній сторінці. Кожен рівень або лекція має окрему HTML-сторінку (lecture3.html), яка виводиться на екран після вибору відповідного рівня з головного меню. Ця сторінка ділиться на дві ключові частини — теоретичну і тестову.

У теоретичному блоці викладається навчальний матеріал з поясненнями, прикладами, таблицями, іноді з використанням перекладів або паралелей з рідною мовою. Це звичайний HTML-контент, стилізований через зовнішній CSS (lecture3.css). Важливо, що він не є статичним назавжди — його можна змінювати, доповнювати або генерувати динамічно через Thymeleaf, через те, що навчальний матеріал зберігається у окремих шаблонах.

Після ознайомлення з темою, користувач переходить до блоку тестування — це HTML-форма з радіокнопками (типові питання з однією правильною відповіддю) та чекбоксами (питання з кількома правильними відповідями). Вони

згруповані у блоки test-item. Така структура дозволяє перевірити засвоєння теорії без переходу на іншу сторінку.

Форма має атрибут th:action="@{/lesson3/submit}", що означає: при натисканні кнопки "Akceptuj odpowiedzi", відповіді користувача не просто надсилаються як класичні HTML-дані, а JavaScript перехоплює подію та надсилає ці дані у форматі JSON методом fetch. У цьому скрипті дані збираються через FormData, перетворюються на об'єкт, і надсилаються POST-запитом на бекенд. На боці сервера створений обробник /lesson3/submit, який приймає JSON, перевіряє правильність відповідей (це може бути контролер Lesson3Controller з методом @PostMapping("/lesson3/submit")), і надсилає результат назад або записує його у базу.

Логіка в тому, що при успішній перевірці відповідей користувач перенаправляється назад на головну сторінку (window.location.href = '/main';) — що, ймовірно, буде сигналізувати про завершення поточного рівня. Якщо ж виникає помилка — виводиться відповідне повідомлення. CSRF-захист реалізовано через токен у заголовках ('X-CSRF-TOKEN': [[${_csrf != null ? _csrf.token : "}]]), що працює разом із Spring Security.

Результати також впливають на зміну інтерфейсу (зафарбування зірочки або маркера рівня, який тепер пройдено) — це реалізується або через зміни в базі даних (на сервері), або через JavaScript-анімацію при наступному завантаженні головної сторінки.

3.3 Реалізація реєстраційної форми

Під час входу в акаунт або реєстрації у вебзастосунку відбувається низка послідовних процесів як на стороні клієнта, так і на сервері. Коли користувач відкриває вебсторінку для входу або реєстрації, клієнтська частина відправляє HTTP-запит до серверного контролера Spring. У випадку реєстрації користувач заповнює форму з необхідними даними, такими як ім’я, електронна пошта та пароль. Ці дані надсилаються у вигляді POST-запиту, який обробляється контролером, позначеним відповідною анотацією, наприклад, @PostMapping.

Контролер приймає дані, зазвичай у вигляді DTO-об’єкта, і передає їх у сервісний шар. У сервісному шарі перевіряється валідність даних. Потім сервіс виконує додаткові перевірки, зокрема, чи вже існує користувач із таким email у базі даних. Якщо все коректно, пароль шифрується, найчастіше за допомогою BCryptPasswordEncoder, і створюється новий об’єкт користувача, який зберігається в репозиторії, що взаємодіє з базою даних через JPA або Hibernate.

У випадку входу користувач вводить логін і пароль, які також надсилаються POST-запитом до контролера. Система шукає відповідного користувача у базі за email, і якщо його знайдено, порівнює наданий пароль із зашифрованим паролем у базі даних, використовуючи той самий BCryptPasswordEncoder. Якщо автентифікація успішна, створюється об’єкт безпеки (UsernamePasswordAuthenticationToken), який передається в контекст безпеки Spring Security. У випадку використання сесійної автентифікації формується HTTP-сесія, і сервер додає відповідний JSESSIONID у cookie відповіді. Якщо ж застосунок використовує токенну автентифікацію (JWT), то на цьому етапі генерується токен, який підписується і передається клієнту у відповіді. Клієнт надалі додає цей токен у заголовок Authorization до кожного запиту. У будь-якому випадку, після успішної автентифікації або реєстрації користувач перенаправляється до головної сторінки або іншого дозволеного ресурсу, відповідно до конфігурації маршрутизації та безпеки, визначеної у Spring Security.

Коли користувач натискає кнопку "Вийти", вебсторінка відправляє запит (зазвичай POST) на адресу /logout. Цей запит перехоплюється Spring Security, яка вже має вбудовану обробку для виходу з облікового запису. У відповідь Spring знищує HTTP-сесію користувача, очищає контекст безпеки, тобто видаляє інформацію про автентифікацію, та видаляє cookie сесії, наприклад JSESSIONID. Після цього Spring перенаправляє користувача на сторінку логіна, додаючи параметр ?logout, що може бути використаний для відображення повідомлення типу "Ви успішно вийшли з облікового запису". Якщо ця логіка не працює, можливо, щось блокує її — наприклад, кеш браузера або неправильне налаштування фільтрів. У випадку з використанням тільки Thymeleaf можна

просто відображати сторінку логіна, де виводиться повідомлення про вихід, без складних механізмів, якщо немає суворих вимог до безпеки.

```
@GetMapping("/registration")
public String registration(Model model){
   model.addAttribute("userEntity", new UserEntity());
   return "registration";
}
@PostMapping("/registration")
public String registration(@ModelAttribute("userEntity") UserEntity
userEntity,Model model){
   userService.saveUser(userEntity);
   return "redirect:/login";
}
```

У цьому фрагменті коду представлено контролер, який відповідає за обробку реєстрації користувачів у вебзастосунку, побудованому за допомогою Spring Framework. Контролер позначено анотацією @Controller, що вказує на те, що він призначений для обробки HTTP-запитів і повернення представлень (HTML-сторінок). Анотація @AllArgsConstructor з бібліотеки Lombok генерує конструктор з усіма необхідними аргументами, що дозволяє автоматично впровадити залежність UserService, яка надає бізнес-логіку для роботи з користувачами. Метод registration, позначений анотацією @GetMapping("/registration"), відповідає за обробку GET-запиту на сторінку реєстрації. У цьому методі в об'єкт моделі додається новий порожній екземпляр UserEntity, який буде використовуватись у формі на HTML-сторінці для прив’язки даних, і повертається назва представлення — шаблон з ім’ям registration, яке Spring Boot передасть до відповідного HTML-файлу (registration.html).

Другий метод, також з назвою registration, але позначений @PostMapping("/registration"), обробляє POST-запити, що надходять при надсиланні форми з даними користувача. Анотація @ModelAttribute("userEntity") вказує Spring автоматично заповнити об’єкт UserEntity даними з форми. Після цього отриманий об’єкт передається у сервіс userService, де викликається метод saveUser, який, імовірно, зберігає користувача у базі даних. Після успішного збереження даних користувача метод повертає інструкцію редиректу на сторінку

входу /login, що означає перенаправлення браузера користувача на відповідну URL-адресу. Таким чином, цей контролер реалізує базову функціональність реєстрації нового користувача з обробкою як відображення форми, так і прийому її результатів.

```
    UserRepo userRepo;
    private BCryptPasswordEncoder encoder(){return new BCryptPasswordEncoder();}
    public void saveUser(UserEntity userEntity){
       userEntity.setPassword(encoder().encode(userEntity.getPassword()));
       userRepo.save(userEntity);
    }
    public List<UserEntity> getAll(){
       return userRepo.findAll();
    }
```

Клас є сервісним компонентом у Spring-застосунку і відповідає за бізнес-логіку, пов’язану з користувачами. Його позначено анотацією @Service, що вказує Spring на те, що цей клас слід автоматично розпізнавати як сервіс і впроваджувати там, де потрібно. Анотація @AllArgsConstructor автоматично створює конструктор з усіма необхідними залежностями, зокрема, UserRepo, який є інтерфейсом доступу до бази даних. Метод saveUser використовується для збереження нового користувача: перед тим як зберегти об’єкт UserEntity, пароль користувача шифрується за допомогою алгоритму BCrypt через об'єкт BCryptPasswordEncoder, який створюється в приватному методі encoder. Це забезпечує безпечне зберігання паролів у базі даних у вигляді хешів, що унеможливлює зворотнє відновлення оригінального пароля. Після цього викликається метод save з userRepo, який здійснює безпосередній запис користувача до бази. Метод getAll призначений для отримання списку всіх користувачів, зчитуючи їх через метод findAll з репозиторію, і повертає цей список як результат. Уся логіка класу зосереджена на роботі з об'єктами користувачів і взаємодії з базою через репозиторій.

```
    private BCryptPasswordEncoder encoder(){return new BCryptPasswordEncoder();}
    @Bean
```

```
    public DaoAuthenticationProvider daoAuthenticationProvider(){
       DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
       auth.setUserDetailsService(userDetailsService);
       auth.setPasswordEncoder(encoder());
       return auth;
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
       http .authorizeHttpRequests(requests ->
              requests
                     .requestMatchers("/registration").permitAll()
                     .anyRequest().authenticated()
       )
       .formLogin(form ->
              form
                     .loginPage("/login")
                     .permitAll()
       )
       .logout(logout -> logout
              .logoutUrl("/logout")
              .logoutSuccessUrl("/login?logout")
              .invalidateHttpSession(true)
              .clearAuthentication(true)
              .deleteCookies("JSESSIONID")
              .permitAll()
       );
```

Клас відповідає за конфігурацію системи безпеки у Spring-застосунку з використанням Spring Security. Його позначено анотаціями @Configuration і @EnableWebSecurity, що вказує Spring на те, що в цьому класі містяться налаштування безпеки. Анотація @AllArgsConstructor з бібліотеки Lombok генерує конструктор, який автоматично впроваджує залежність UserDetailsServiceImpl — сервіс, що відповідає за завантаження користувачів із бази даних.

У класі визначено приватний метод encoder, який створює новий екземпляр BCryptPasswordEncoder — це механізм для шифрування паролів, який буде використано при перевірці автентичності. Метод daoAuthenticationProvider, позначений анотацією @Bean, створює і налаштовує об’єкт DaoAuthenticationProvider, що є провайдером автентифікації, який працює з базою

даних через сервіс userDetailsService і використовує BCryptPasswordEncoder для перевірки паролів.

Основна частина налаштувань безпеки розміщена в методі securityFilterChain, який також позначено як @Bean, і він приймає об'єкт HttpSecurity. У цьому методі спочатку налаштовується авторизація: сторінка /registration дозволена всім без автентифікації, а всі інші запити вимагають входу в систему. Далі налаштовується форма входу: вона використовує власну сторінку за шляхом /login, яка також доступна без автентифікації. Завершує конфігурацію блок logout, який задає шлях до виходу з системи (/logout) і вказує, що після виходу користувач буде перенаправлений на сторінку /login?logout, при цьому сесія буде інвалідована, автентифікація очищена, а кукі JSESSIONID видалено. У підсумку, цей клас визначає повну політику безпеки для застосунку — від автентифікації користувачів до доступу до окремих маршрутів.

```
public void init(FilterConfig filterConfig) throws ServletException {
   Filter.super.init(filterConfig);
}
@Override
public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
        throws IOException, ServletException {

   HttpServletResponse httpResponse = (HttpServletResponse) response;
   httpResponse.setHeader("Cache-Control", "no-cache, no-store, must-
revalidate"); // HTTP 1.1
   httpResponse.setHeader("Pragma", "no-cache"); // HTTP 1.0
   httpResponse.setDateHeader("Expires", 0); // Proxies

   chain.doFilter(request, response);
}
@Override
public void destroy() {
   Filter.super.destroy();
}
```

Клас реалізує фільтр Java Servlet API, який відповідає за заборону кешування HTTP-відповідей браузером, і позначений анотацією @Component, що дозволяє Spring автоматично зареєструвати його як компонент фільтра у контексті

застосунку. Клас реалізує інтерфейс Filter і таким чином втручається в обробку всіх HTTP-запитів, перш ніж вони дійдуть до контролера або після формування відповіді.

Основна логіка розміщена в методі doFilter, який виконується при кожному HTTP-запиті. Всередині цього методу відповідь приводиться до типу HttpServletResponse, після чого в неї додаються спеціальні заголовки, що забороняють кешування. Заголовок Cache-Control з параметрами no-cache, no-store, must-revalidate використовується в HTTP 1.1 і забороняє зберігання копій ресурсу. Заголовок Pragma: no-cache призначений для сумісності з HTTP 1.0. Метод setDateHeader("Expires", 0) вказує, що ресурс уже прострочений і не повинен зберігатись кешуючими проксі. Після встановлення заголовків, викликається chain.doFilter(request, response), що передає запит і відповідь далі по ланцюжку фільтрів або до контролера.

Методи init і destroy залишені з викликами методів суперкласу, що дозволяє Spring за потреби керувати ініціалізацією або знищенням фільтра. У результаті цей фільтр гарантує, що динамічний вміст, такий як особисті сторінки або сторінки після входу в систему, не буде кешуватись браузером чи проксі-сервером, що є важливою мірою безпеки.

3.4 Створення окремих навчальних блоків

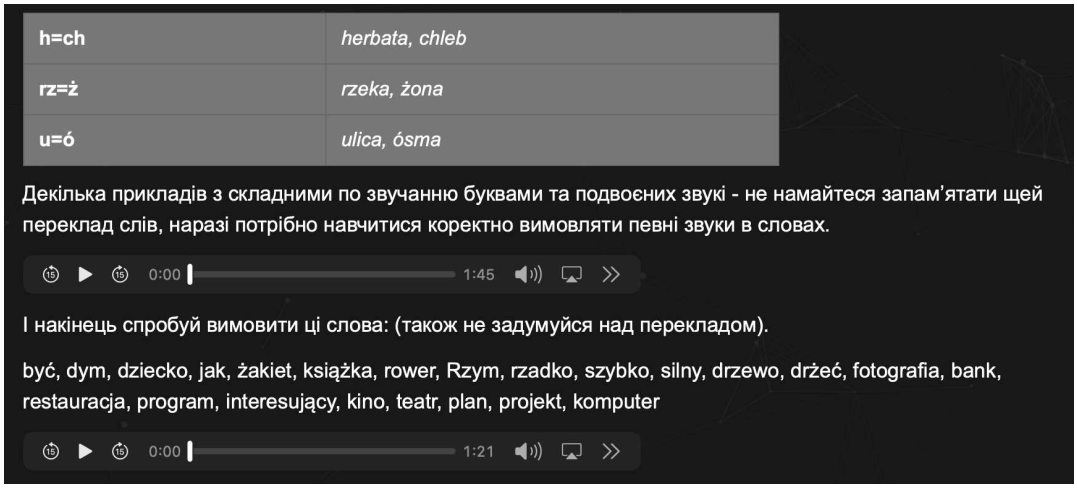


Рисунок 3.1 — Фрагмент інтерфейсу сторінки lecture1.html

Кожен навчальний блок є окремою секцією, яка складається з відповідного html файла, css файла і контент компонентів — рисунки, аудіо файли тощо.

Сторінка першого уроку (рисунок 3.1) з польської мови — це зручний і продуманий старт для тих, хто хоче познайомитися з основами. Вона створена з використанням Thymeleaf, Bootstrap і кастомних стилів із файлу lecture1.css, а для красивого візуального ефекту додано анімований фон через бібліотеку particles.js (шукайте блок із id="particles-js"). У центрі сторінки — напівпрозорий контейнер, де зібрано весь вміст уроку, що виглядає стильно і не відволікає від навчання.

Урок знайомить із польським алфавітом, пояснює, як працює наголос (зазвичай на передостанній склад), і розповідає про особливості вимови ненаголошених голосних та спрощення звуків. Наприклад, є аудіофайл із вимовою алфавіту, щоб одразу почути, як звучать літери.

Далі йдуть таблиці, які допомагають розібратися з польськими приголосними, особливо тими, що можуть бути складними для україномовних. У перших двох таблицях — літери та приклади слів із ними, а в третій — літери, які пишуться по-різному, але звучать однаково. Це зручно для розуміння тонкощів польської фонетики.

Потім пропонуються слова зі складними звуками чи подвоєними приголосними. Користувач може прослухати аудіо, повторити слова вголос і потренувати вимову, не заглиблюючись у переклад. Це допомагає звикнути до звучання мови.

Наприкінці є форма для самоперевірки з різними типами запитань: вибір однієї відповіді, правда/неправда та поле для відкритої відповіді. Після заповнення відповіді відправляються на сервер у форматі JSON через JavaScript (за допомогою fetch), а стандартне відправлення форми блокується. Для безпеки додано захист від CSRF через токен, який автоматично вставляє Thymeleaf.

Загалом, сторінка виглядає сучасно, інтерактивно і допомагає легко зробити перші кроки у вивченні польської мови.

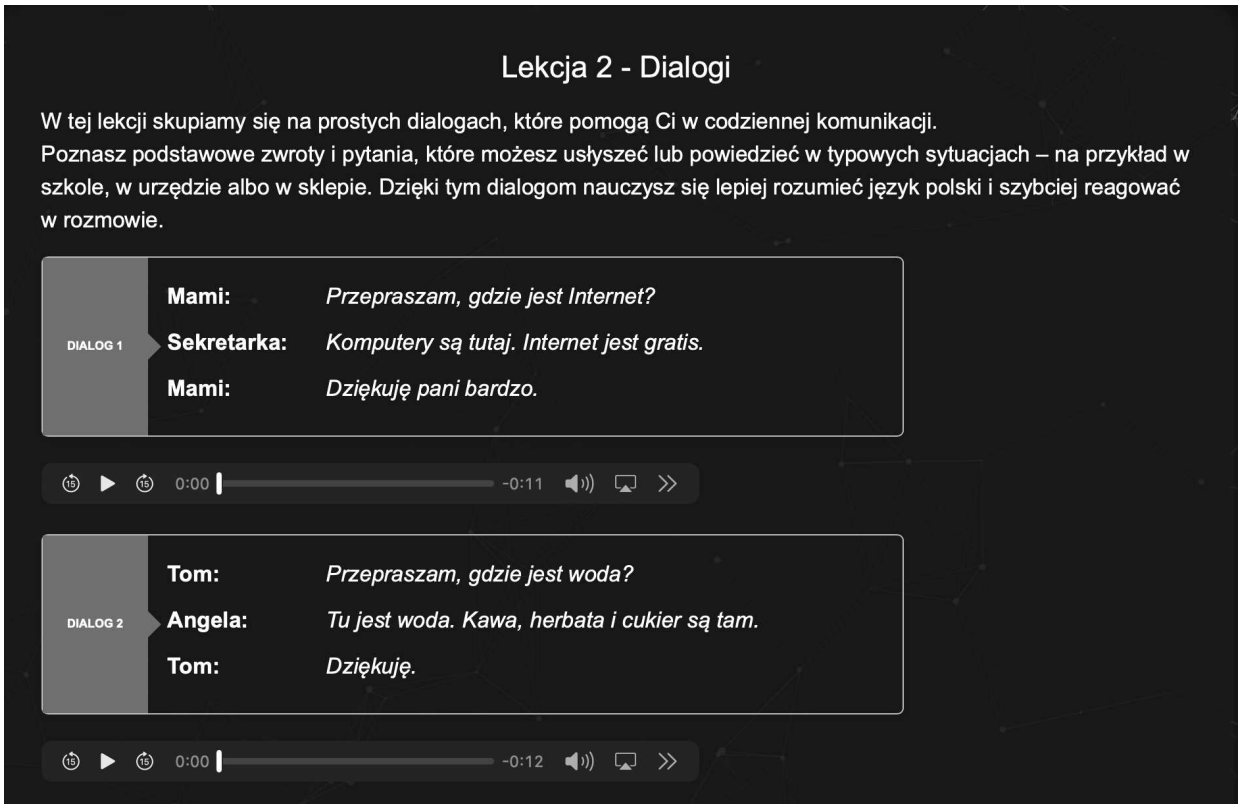


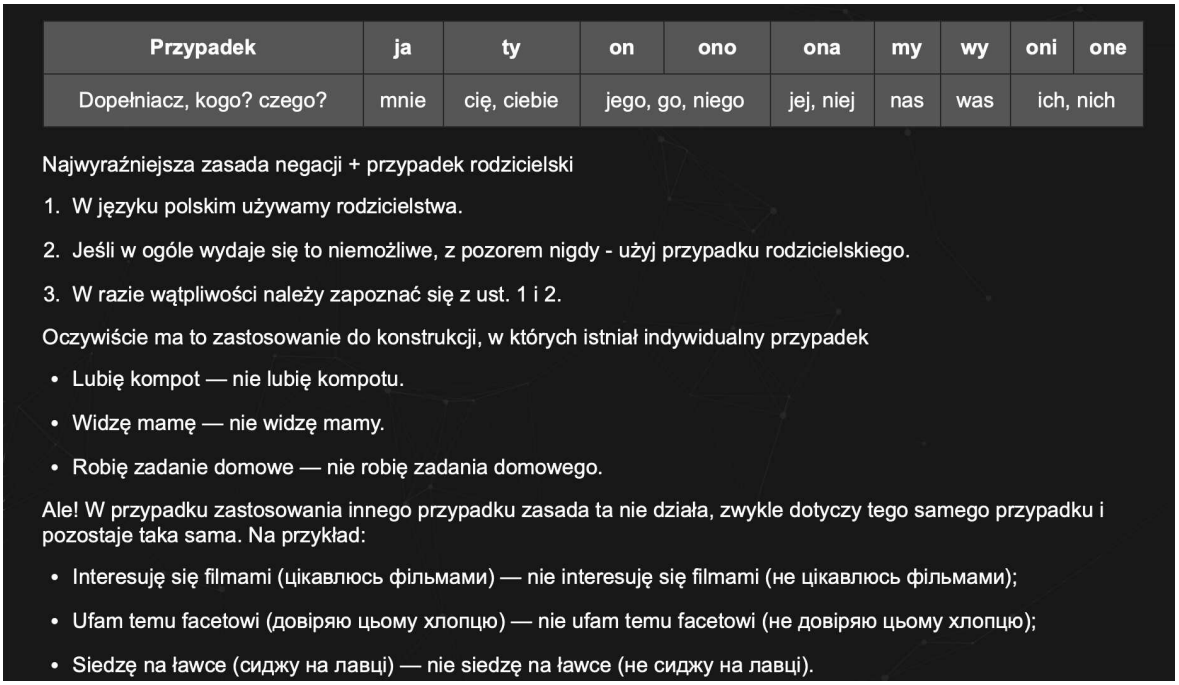
Рисунок 3.2 — Фрагмент інтерфейсу сторінки lecture2.html

Сторінка є частиною мовного курсу (рисунок 3.2) й представляє собою другий урок, присвячений вивченню простих діалогів польською мовою. Основною метою сторінки є інтеграція навчального матеріалу — коротких діалогів, озвучених носіями мови, а також створення простого інтерфейсу для перевірки розуміння через автотест. У шапці підключено Bootstrap для базової стилізації та кастомний CSS-файл, який, ймовірно, відповідає за візуальну частину, зокрема напівпрозорий фон і блок із частинками.

Вся сторінка вміщена в контейнер із фоном із анімацією (через particles.js), що створює атмосферу інтерактивного навчання. У центрі — головний блок із заголовком уроку та вступним текстом, який коротко пояснює зміст заняття та його користь.

Основна частина складається з послідовності діалогів. Кожен діалог оформлений в окремому контейнері, де зліва розташована бічна панель із заголовком "DIALOG X", а праворуч — власне контент із діалоговими репліками. Репліки чітко структуровані: є імена персонажів та їх висловлювання. Після

кожного діалогу йде аудіоплеєр із відповідним звуковим файлом, що дозволяє слухачу почути автентичну вимову. Таким чином, користувач може не лише читати, а й тренувати сприймання на слух.



Przypadek	ja	ty	on	ono	ona	my	wy	oni	one
Dopełniacz, kogo? czego?	mnie	cię, ciebie	jego, go, niego		jej, niej	nas	was	ich, nich	

Najwyraźniejsza zasada negacji + przypadek rodzicielski

1. W języku polskim używamy rodzicielstwa.
2. Jeśli w ogóle wydaje się to niemożliwe, z pozorem nigdy - użyj przypadku rodzicielskiego.
3. W razie wątpliwości należy zapoznać się z ust. 1 i 2.

Oczywiście ma to zastosowanie do konstrukcji, w których istniał indywidualny przypadek

- Lubię kompot — nie lubię kompotu.
- Widzę mamę — nie widzę mamy.
- Robię zadanie domowe — nie robię zadania domowego.

Ale! W przypadku zastosowania innego przypadku zasada ta nie działa, zwykle dotyczy tego samego przypadku i pozostaje taka sama. Na przykład:

- Interesuję się filmami (цікавлюсь фільмами) — nie interesuję się filmami (не цікавлюсь фільмами);
- Ufam temu facetowi (довіряю цьому хлопцю) — nie ufam temu facetowi (не довіряю цьому хлопцю);
- Siedzę na ławce (сиджу на лавці) — nie siedzę na ławce (не сиджу на лавці).

Рисунок 3.3 — Фрагмент інтерфейсу сторінки lecture3.html

HTML-файл (рисунок 3.3) є шаблоном сторінки, створеного з використанням Thymeleaf для динамічної генерації контенту на сервері Java. У <head> розміщено підключення до стилів Bootstrap, а також посилання на зовнішній CSS-файл lecture3.css. Уся сторінка оформлена в навчальному стилі: спочатку представлено теоретичний матеріал про граматичні правила, які пояснюють, як змінюється відмінок іменника в польській мові при запереченні. Матеріал подано польською мовою, частково з українськими перекладами, щоб зробити засвоєння матеріалу простішим для україномовного користувача.

Основна структура складається з декількох абзаців тексту та прикладів речень, що демонструють зміну відмінка. Наводяться правильні приклади заперечень із поясненням, коли потрібно використовувати родовий відмінок, і коли він не змінюється (як у конструкціях із прийменниками або дієсловами, що вимагають іншого відмінка).

У другій частині сторінки реалізована інтерактивна секція з тестами. Вона складається з форми, що надсилається через POST-запит за допомогою JavaScript. Всі питання є тестовими — частина з одним правильним варіантом відповіді (радіокнопки), частина з кількома можливими правильними варіантами (чекбокси). Для кожного питання подано інструкцію, запитання, і список відповідей. Наприкінці розміщено кнопку для надсилання відповідей.

Форма використовує атрибут th:action, що означає обробку URL сервером через Thymeleaf. Після натискання кнопки відбувається обробка форми за допомогою JavaScript: дані збираються в об'єкт FormData, перетворюються в JSON і надсилаються на сервер. Застосовано захист від CSRF-атак через токен, який Thymeleaf динамічно підставляє у заголовок запиту.

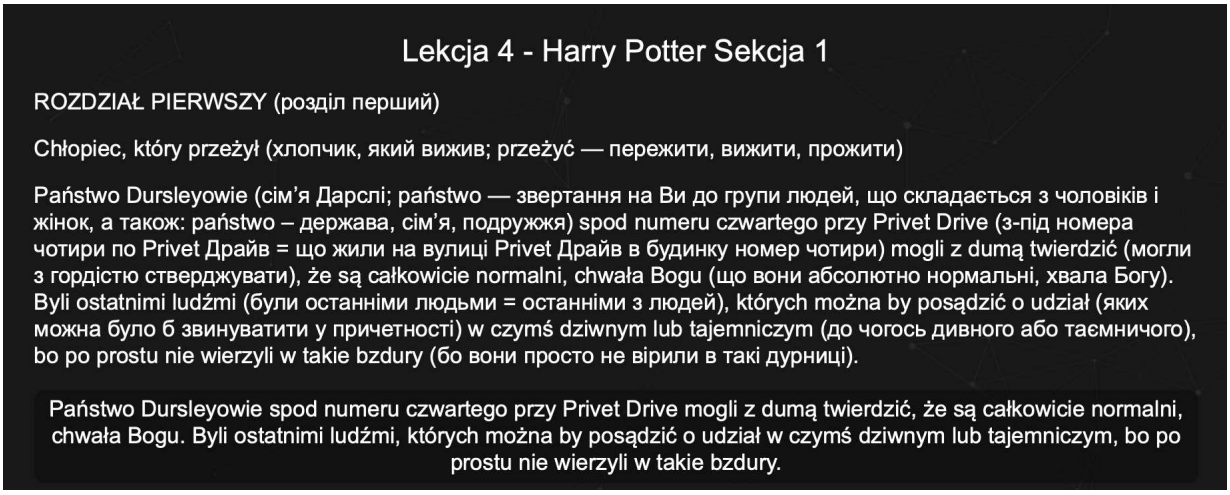


Рисунок 3.4 — Фрагмент інтерфейсу сторінки lecture4.html

У <head> містяться стандартні мета-теги для коректного відображення на різних пристроях, а також підключення до Bootstrap для базової стилізації та зовнішнього CSS-файлу lecture4.css.

Основна частина сайту (рисунок 3.4) знаходиться в <body> і містить навчальний контент. Фоном виступає декоративна анімація particles.js, яка додає візуальної привабливості. Центральний блок сторінки — це контейнер з текстом уроку. Він поділений на дві основні частини: спочатку йде текст польською з перекладом на українську (атрибути класу intro), а потім подається той самий

текст лише польською (story). Це зроблено для поетапного ознайомлення користувача спершу з перекладом, а потім із суцільним текстом для практики.

Після основного тексту йде словничок, який пояснює нову лексику — подано польські слова та їхній переклад українською, що допомагає краще зрозуміти зміст прочитаного.

Нижче розташований блок з тестовими питаннями (форма з атрибутами method="post" і th:action для серверної обробки). Кожне питання містить текст і декілька варіантів відповіді у вигляді радіокнопок. Наприкінці розміщена кнопка для надсилання відповідей.

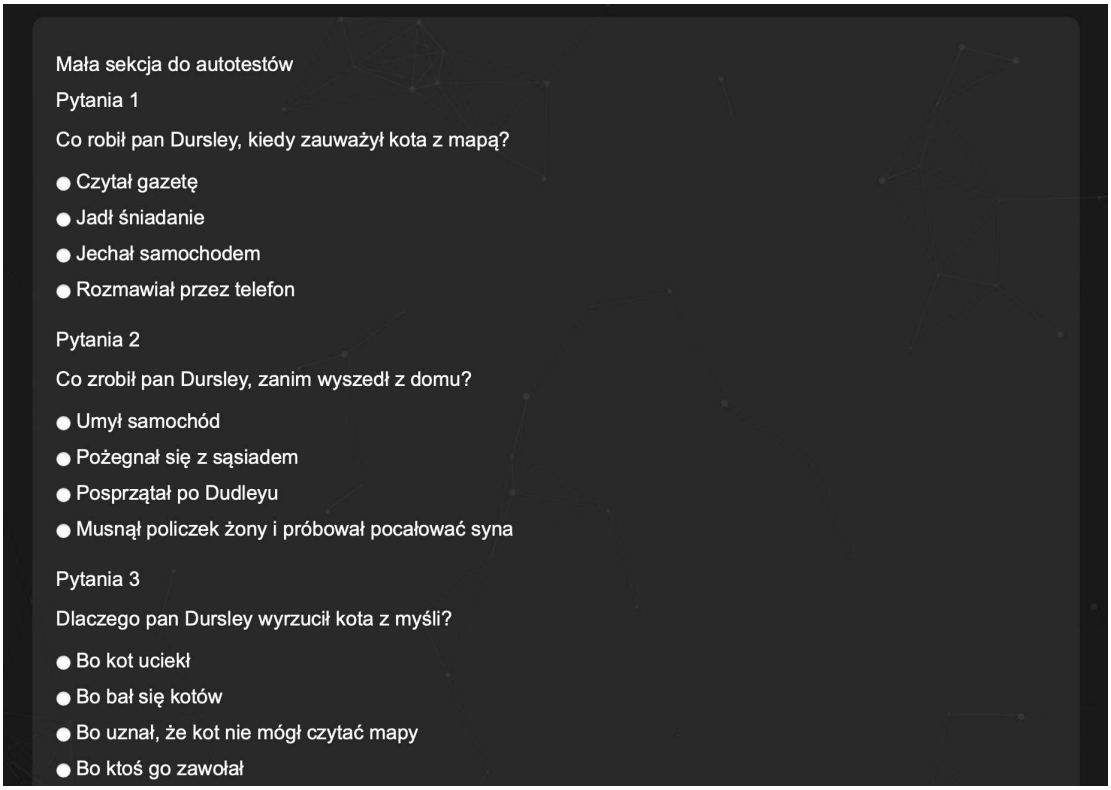


Рисунок 3.5 — Фрагмент інтерфейсу сторінки lecture5.html

У <head> підключено Bootstrap для стилізації інтерфейсу, а також кастомний CSS-файл, шлях до якого визначається динамічно через Thymeleaf. У тілі сторінки (рисунок 3.5) присутній контейнер з ідентифікатором level-container, який містить заголовки з текстами уроку, представленими парами: спочатку адаптований польський текст із коментарями й українським перекладом, а потім

оригінальний фрагмент без пояснень. Таке дублювання полегшує сприйняття матеріалу.

JavaScript у самому низу реалізує логіку відправлення форми: якщо відповідь успішно прийнята сервером, користувача перенаправляють на головну сторінку, інакше виводиться повідомлення про помилку. Також підключено бібліотеку particles.js, ймовірно, для анімованого фону.

3.5 Розробка основної логіки та стилізація

Значну частину логіки застосунку беруть на себе класи LessonController, LessonService в яких описаний функціонал до тестів, підключення файлів до відповідних баз даних та методика розрахування балів за них.

```
@GetMapping("/lessons/lecture1")
public String lesson1() {
   return "/lessons/lecture1";
}

@PostMapping("/lesson1/submit")
@ResponseBody
public ResponseEntity<?> submitLesson1Answers(@RequestBody Map<String, String> answers) {
   // Get current authenticated user
   Authentication auth = SecurityContextHolder.getContext().getAuthentication();
   String username = auth.getName();
   lessonService.processAndSaveAnswers(username, "lesson1", answers);
   return ResponseEntity.ok().build();
}
```

Фрагмент коду представляє собою Java-контролер у Spring Framework, який відповідає за обробку запитів, пов’язаних із лекціями в веб-додатку. Він використовує анотацію @Controller, що вказує на те, що клас є веб-контролером, який обробляє HTTP-запити та повертає представлення (шаблони HTML).

Клас має залежність від сервісу LessonsService, який інжектується за допомогою анотації @Autowired. Цей сервіс використовується для обробки та збереження відповідей користувача на тест після проходження уроку.

Методи з анотацією @GetMapping повертають HTML-сторінки відповідних лекцій: /lessons/lecture1, /lessons/lecture2 і так далі до лекції 5. Ці методи повертають шлях до відповідних шаблонів, які, відображаються у браузері за допомогою системи шаблонів Thymeleaf.

Методи з анотацією @PostMapping (від lesson1/submit до lesson5/submit) обробляють POST-запити, що надсилаються після проходження тестів користувачами. У кожному з цих методів:

— зчитуються дані відповідей з тіла запиту за допомогою @RequestBody, що дозволяє автоматично розпізнати JSON і конвертувати його у мапу Map<String, String>;

— отримується ім’я поточного автентифікованого користувача через SecurityContextHolder;

— викликається метод processAndSaveAnswers() сервісу lessonService, який обробляє та зберігає передані відповіді в контексті певного уроку (визначеного рядком "lesson1", "lesson2" тощо);

— повертається успішна HTTP-відповідь ResponseEntity.ok(), яка повідомляє, що обробка завершена без помилок.

Клас LessonsService відповідає за логіку обробки відповідей користувача в навчальній системі. Він зберігає у статичних мапах правильні відповіді на запитання (CORRECT_ANSWERS) та типи запитань (QUESTION_TYPES) для кожного уроку. Типи запитань визначають, як порівнювати відповіді (одиночний вибір, множинний вибір або текст).

```
    private static final Map<String, Map<String, String>> CORRECT_ANSWERS = new HashMap<>();
    private static final Map<String, Map<String, Integer>> QUESTION_TYPES = new HashMap<>();

    static {
        Map<String, String> lesson1Answers = new HashMap<>();
        lesson1Answers.put("question1", "A");
        lesson1Answers.put("question2", "D");
        lesson1Answers.put("question3", "false");
        lesson1Answers.put("question4", "rzeka,zona");
```

```
    CORRECT_ANSWERS.put("lesson1", lesson1Answers);
    Map<String, Integer> lesson1Types = new HashMap<>();
    lesson1Types.put("question1", 0);
    lesson1Types.put("question2", 0);
    lesson1Types.put("question3", 0);
    lesson1Types.put("question4", 2);
    QUESTION_TYPES.put("lesson1", lesson1Types);
}
```

Метод processAndSaveAnswers приймає ім’я користувача, ідентифікатор уроку та мапу відповідей користувача. Він видаляє попередні відповіді для цього уроку та користувача, обчислює кількість правильних відповідей і бал за кожне запитання з урахуванням типу запитання, а потім зберігає нові відповіді у базу даних. Також метод оновлює об’єкт LessonProgress, який відображає загальний прогрес користувача по уроку.

Метод calculateStringSimilarity обчислює ступінь схожості між двома рядками на основі нормалізованої відстані Левенштейна — це потрібно для текстових відповідей, де враховується часткова відповідність. Метод levenshteinDistance реалізує сам алгоритм обчислення відстані Левенштейна між двома рядками, тобто кількість змін, які потрібно внести, щоб один рядок перетворити на інший. Метод min — допоміжна функція для обчислення мінімуму з трьох значень, яка використовується в алгоритмі.

Метод getLessonProgressPercentage повертає лише відсоток завершення уроку користувачем, витягуючи його з об'єкта LessonProgress.

Метод getLessonProgress повертає повну інформацію про прогрес користувача в уроці, включно з кількістю правильних відповідей і загальною кількістю запитань.

Метод getUserAnswers дозволяє отримати всі відповіді користувача на певний урок, що може бути корисним для подальшого аналізу, перегляду результатів або повторного відображення відповідей у UI.

3.6 Тестування та перевірка коректності роботи

Задля правильної роботи веб застосунку потрібно перевірити функціонал деяких компонентів, а саме: — слайдер, який демонструє картинки з інформацією,

—каталог з лекціями та результатами проходження тестових блоків, — реєстраційна форма та форма логіну, — вихід із акаунту та коректність витягу інформації з бази даних для відповідного користувача.

Для компонентів логіна та реєстрації (рисунок 3.6) були обрані схожі стильові наповнення, а саме — темний колір фону, білий текст та яскраві кнопки для запису інформації.

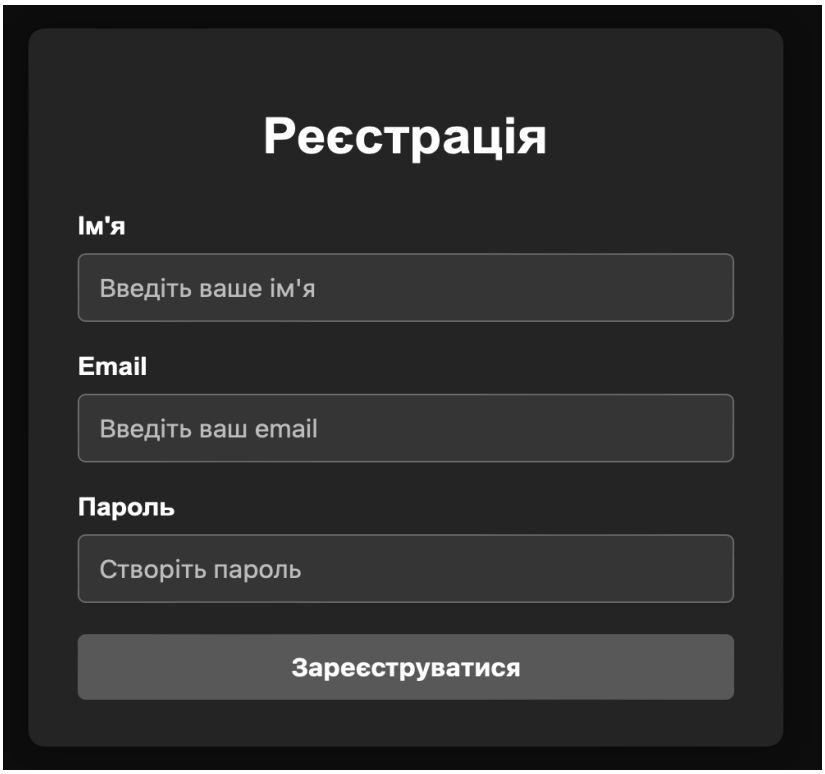


Рисунок 3.6 — Фрагмент реєстраційної форми

Код для реєстрації та логіна створює сучасний веб-інтерфейс для автентифікації користувачів з приємним темно-синім фоном та напівпрозорими елементами. Стиль body задає основний колір фону #0a1929 та центрує весь контент через flexbox. Контейнер .container має напівпрозорий фон з ефектом розмиття та використовує сучасний blur-ефект для глибини. Текстові елементи відображаються білим кольором для контрасту з темним фоном. Поля вводу стилізовані з напівпрозорим темним фоном, світлими межами та білим текстом, а placeholder-текст має знижену непрозорість для кращої UX. Кнопки відправки форм мають яскраво-синій фон #4361ee зі зміною кольору при наведенні для

інтерактивності. Для логіна додано стилізовані повідомлення про помилки через клас .alert, червоні для помилок та зелені для успішних дій. Посилання на реєстрацію внизу логін-форми має білий колір із красивим ефектом підкреслення при наведенні. Весь дизайн адаптивний завдяки відносним розмірам та max-width. У коді для логіна використовуються Thymeleaf умови для відображення станів помилок та виходу із системи. Обидві сторінки мають єдину візуальну концепцію з мінімалістичним оформленням, що фокусує увагу користувача на функціях автентифікації.

Рисунок 3.7 — Слайдер головної сторінки

Після логіну користувача зустрічає головна сторінка на якій одразу коректно протестовано показ інформації (рисунок 3.7) — username, якій користувач вводив при реєстрації був занесений до бази даних та успішно взятий звідти після заходу саме цього користувача в акаунт. Також слайдер із інформацію, як і попередній компонент веде себе коректно та правильно відображає всі блоки.

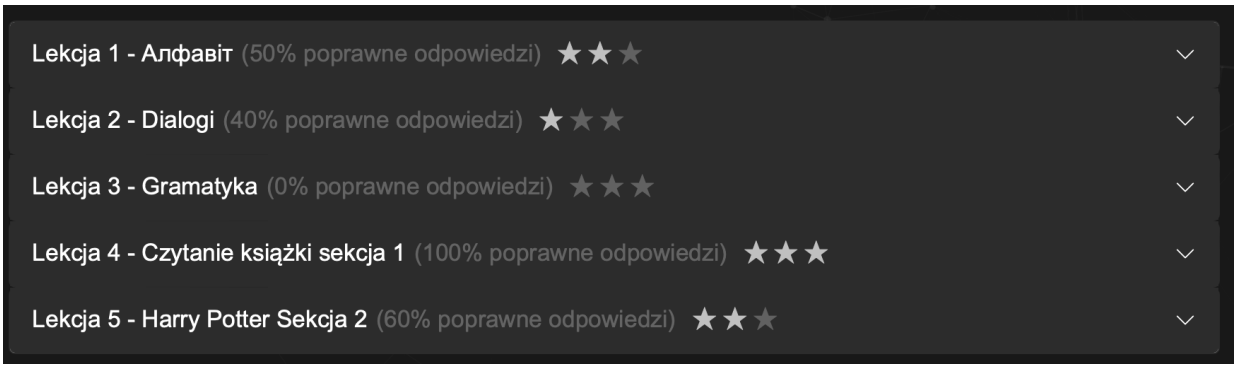


Рисунок 3.8 — Каталог головної сторінки

На представленому рисунку блок з навчальними лекціями працює коректно, кожен його елемент виконує очікувані функції. Кожна лекція відображається окремим рядком із чіткою назвою, яка відповідає тематиці уроку, наприклад "Lekcja 1 – Алфавіт", "Lekcja 2 – Dialogi", що свідчить про правильну генерацію назв з урахуванням локалізації. Поруч із назвою відображається відсоток правильних відповідей у дужках, наприклад "50% poprawne odpowiedzi", "100%", що демонструє успішне зчитування результатів користувача з бази або іншого джерела збереження прогресу. Ці результати відповідають очікуваному рівню складності: наприклад, при 0% правильних відповідей у "Gramatyka" бачимо нуль зірок, а при 100% — повний набір з трьох зірок у "Czytanie książki sekcja 1". Таким чином, відображення зірок динамічно реагує на показники тесту, тобто система правильно виконує перерахунок і візуалізацію нагород за результатами. Візуально видно, що елементи з однаковою структурою мають однакове розміщення: назва, відсоток правильних відповідей у зеленому кольорі, потім блок зі зірками та стрілка розкриття в правому куті. Стрілки свідчать про наявність прихованого контенту, — детального опису завдань, тобто блок інтерактивний. Структура інтерфейсу збережена, немає зміщень або порушення верстки, а стилі для активних елементів (зірок) відповідають очікуваним станам: жовті зірки — активні, сірі — неактивні. Це означає, що логіка конвертації результатів у кількість зірок працює правильно. Таким чином, весь блок виконує свої функції: відображає назви лекцій, інформує про успішність проходження тестів, візуалізує досягнення за допомогою зірок і передбачає можливість відкриття кожного

розділу, що свідчить про успішне проходження інтеграційного, візуального й функціонального тестування інтерфейсу.

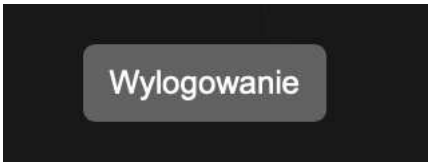


Рисунок 3.10 — Кнопка виходу з акаунту

Кнопка «Wylogowanie» (рисунок 3.10), розташована в нижній частині сторінки, також працює коректно, відповідно до стандартів взаємодії з користувачем. Її позиціонування є логічним і звичним для більшості додатків — вона знаходиться внизу інтерфейсу, що дозволяє легко завершити сеанс після перегляду всіх навчальних розділів. Кнопка має зрозуміле текстове позначення, яке чітко сигналізує про її призначення, а також, візуально відокремлена або стилізована так, щоб не змішувалась із навчальним контентом, чим підвищує зручність користування. Натискання цієї кнопки викликає очікувану дію — завершення сеансу користувача з очищенням сесії або токена автентифікації та перенаправленням на вхідну сторінку чи головну сторінку додатка. Це свідчить про коректну реалізацію як клієнтської логіки, так і взаємодії з бекендом, зокрема API виходу. Кнопка не створює плутанини в інтерфейсі, не перешкоджає навігації та не дублюється, що говорить про продумане UX-рішення. Таким чином, функціональність виходу з облікового запису реалізована відповідно до очікувань користувача, забезпечуючи логічне завершення роботи в додатку та гарантує захищеність особистих даних після завершення сесії.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ В КРОСПЛАТФОРМНОМУ СЕРЕДОВИЩІ

4.1 Тестування системи на телефонних пристроях

У процесі реалізації вебзастосунку для вивчення іноземної мови було обґрунтовано вибір апаратно-програмного середовища, що забезпечує ефективну, стабільну та кросплатформну роботу системи. Оскільки основною платформою є веб, доцільно було використовувати інструменти, які гарантують сумісність із більшістю сучасних браузерів і операційних систем, включно з Windows, macOS, Linux, Android та iOS. Для цього було обрано мову програмування Java для реалізації серверної логіки, що забезпечує масштабованість, безпечність обробки даних та зручну інтеграцію з базами даних. У якості клієнтської частини було використано HTML, CSS і JavaScript із сучасними технологіями стилізації та валідації, що дозволило створити адаптивний, динамічний та естетично привабливий інтерфейс. Для зберігання і обробки навчального контенту, результатів тестування та даних користувачів застосовано реляційну базу даних PostgreSQL, яка забезпечує швидкий доступ до інформації, надійність та підтримку багатокористувацького середовища. Розробка відбувалась у середовищі IntelliJ IDEA з використанням фреймворку Spring Boot, що значно спростило конфігурацію і дало змогу зосередитись на реалізації бізнес-логіки застосунку. Для забезпечення кросбраузерної та кросплатформної сумісності регулярно проводилось тестування в Google Chrome, Firefox і Safari, а також на різних мобільних пристроях. Окрім того, під час розробки враховувались вимоги адаптивності для різних розмірів екранів, що дозволяє користуватись системою як на десктопних, так і на мобільних пристроях без втрати функціональності чи зручності. Таке середовище забезпечує високу якість виконання застосунку, зручність подальшої підтримки і розширення функціоналу.

Рисунок 4.1 — Відображення застосунку на телефоні

Розробка вебзастосунку з урахуванням коректного відображення на мобільних пристроях (рисунок 4.1) здійснювалась відповідно до принципів адаптивного дизайну, що передбачає автоматичне підлаштування інтерфейсу під різні розміри екранів. На початковому етапі було визначено основні точки перегляду (breakpoints) для найпоширеніших пристроїв: смартфонів, планшетів і десктопів. Було обрано mobile-first підхід, при якому верстка спочатку оптимізується під найменший розмір екрана, а потім поступово розширюється до більших. Для побудови гнучких макетів використовувались сучасні технології CSS — Flexbox і Grid, які дозволяють легко компонувати елементи в різних орієнтаціях, не втрачаючи зручності сприйняття.

Також були застосовані медіа-запити (media queries), які адаптують розміри шрифтів, відступів, кнопок, блоків та інших елементів інтерфейсу. Кнопки для мобільних пристроїв мають збільшену зону натискання, а текст адаптується під вертикальну прокрутку, яка є основною для смартфонів. Додатково було оптимізовано зображення — завантажуються полегшені версії файлів для пришвидшення роботи на мобільному інтернеті.

4.2 Тестування системи на планшетних пристріях

Для забезпечення цілісного користувацького досвіду на планшетних пристроях (рисунок 4.2) було протестовано кожну сторінку застосунку в різних браузерах (Chrome, Safari, Firefox) і на реальних пристроях із різними роздільностями екрана. Планшети займають проміжне положення між мобільними телефонами та повноцінними комп’ютерами, тому важливо було зберегти баланс між компактністю та повною функціональністю елементів інтерфейсу.

Рисунок 4.2 — відображення застосунку на планшеті

Для цього в рамках верстки застосовувались медіа-запити CSS, які дозволяють перебудовувати компонування сторінки відповідно до роздільної здатності та орієнтації екрана.

Одним із ключових рішень стала зміна структури макету: багатоколонкові блоки трансформуються у вигляд з однією або двома колонками, що забезпечує комфортне сприйняття інформації без необхідності масштабування чи горизонтального прокручування. Наприклад, блоки завдань у тестах, лекційні картки чи елементи профілю користувача автоматично змінюють свої пропорції, щоби залишатися читабельними та зручними для натискання пальцем. Інтерактивні елементи, як-от кнопки, поля для введення, перемикачі, були збільшені до мінімального розміру, рекомендованого для сенсорних пристроїв, згідно з вимогами доступності.

Окрему увагу було приділено адаптації верхньої навігаційної панелі. На широких планшетах панель залишається у горизонтальному вигляді, але на вузьких — перетворюється на компактне випадаюче меню (бургер-меню), що економить простір на екрані й зберігає естетику інтерфейсу. Також були враховані відмінності в поведінці сенсорного вводу: усі функціональні елементи протестовані на відповідність гештам торкання, скролінгу та свайпам, щоб виключити можливість випадкових натискань.

Під час тестування інтерфейс перевірявся як на емульованих пристроях у браузерних інструментах розробника, так і на реальних планшетах з різними операційними системами, включаючи iPadOS та Android. Це дозволило виявити потенційні проблеми з розташуванням або масштабуванням окремих блоків і своєчасно їх усунути. У результаті вебзастосунок зберігає свою повну функціональність і візуальну привабливість на планшетах, дозволяючи користувачеві комфортно працювати з навчальним матеріалом, проходити тести та взаємодіяти з усіма розділами системи.

4.3 Тестування системи на пк та ноутбуках

Для забезпечення коректного відображення вебзастосунку на персональних комп’ютерах і ноутбуках (рисунок 4.3) під час розробки було враховано специфіку великих екранів і застосовано адаптивний підхід до верстки. Основу інтерфейсу складали гнучкі сітки, реалізовані за допомогою CSS-технологій Flexbox та Grid, що дали змогу рівномірно розподіляти елементи по ширині екрана, адаптувати компоненти до змін розміру вікна браузера та зберігати цілісність візуального стилю. Особливу увагу було приділено створенню зручної навігаційної панелі, яка на великих екранах відображається горизонтально з доступом до всіх основних розділів без необхідності згортання в бургер-меню. Також було протестовано відображення в різних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge), щоб унеможливити розриви верстки, некоректні шрифти або зсуви елементів, і виявлені розбіжності усувались через медіа-запити та уточнення CSS-властивостей. У результаті користувачі, що відкривають застосунок на комп’ютерах або ноутбуках, бачать стабільний, зручний і гармонійно оформлений інтерфейс.

Рисунок 4.3 — відображення застосунку на ПК

ВИСНОВКИ

Бакалаврську дипломну роботу було присвячено створенню та впровадженню інформаційної системи для онлайн-платформи, що допомагає користувачам у вивченні іноземної мови. Основна мета полягала в реалізації інтерактивного веб застосунку, який забезпечує доступ до навчальних матеріалів, полегшує засвоєння граматики, лексики, конструкцій діалогів і дозволяє перевіряти знання за допомогою тестування. Розроблена система орієнтована на широке коло користувачів, які прагнуть покращити рівень володіння мовою, отримати чітку структуру навчання й можливість практичного закріплення знань через взаємодію з контентом.

У процесі реалізації було проаналізовано сучасні вебтехнології, що дозволяють створити зручний, функціональний і адаптивний інтерфейс, який підходить як для десктопних, так і мобільних пристроїв. Для клієнтської частини використовувались HTML, CSS і JavaScript, а для серверної частини мова — Java та її бібліотеки Spring web, boot, thymeleaf, security, data-jpa та loombook. Враховувались особливості кольорової гами — обрані кольори сприяють зменшенню втоми при тривалому навчанні й підвищують загальну привабливість застосунку.

Структура вебзастосунку побудована на основі навчальних розділів, які включають лекції, вправи, тести та підсумкові завдання. Кожен розділ містить окрему навчальну тему, яку можна проходити у зручному темпі, повертатись до складних місць, повторювати тестування для поглиблення розуміння, що особливо важливо у разі труднощів або неповного засвоєння матеріалу. Такий підхід дозволяє користувачу самостійно регулювати ритм навчання й досягати кращих результатів через повторення та самоперевірку.

Особливістю застосунку є логічно вибудувана траєкторія вивчення мови — від знайомства з алфавітом і простими словами до побудови діалогів і вивчення граматичних правил, що згодом веде до практичного читання оригінальних текстів, зокрема уривків із культових книжок, як-от «Гаррі Поттер». Це дозволяє

не лише вивчати мову, а й відразу застосовувати її у змістовному контексті, розвиваючи навички читання, розуміння, перекладу й аналізу.

Додатковим функціональним блоком стала система мотивації — реалізовано заохочувальні елементи, включно зі збиранням зірок за правильно виконані тести, що позначають рівень засвоєння розділу. Прогрес візуалізується через підсвічування зірок, що стимулює користувача прагнути до повного засвоєння матеріалу. Такий механізм позитивно впливає на регулярність навчання і зацікавлення.

Усі поставлені завдання були виконані повною мірою. Розроблена система відповідає вимогам щодо структури, дизайну, зручності використання та ефективності подачі навчального матеріалу. Вона має потенціал до масштабування, інтеграції з базами даних, додавання облікових записів користувачів, персоналізації навчання, а також подальшого розвитку в повноцінну навчальну платформу для вивчення різних мов.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загубін О. В., Крупельницький Л. В. Кросплатформна інформаційна система для допомоги у вивченні іноземної мови // Матеріали всеукраїнської науково-практичної Інтернет-конференції студентів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (15листопада 2024 р . - 14 червня 2025 р., Вінниця) : — Режим доступу : https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25305

2. Java Documentation. [Електронний ресурс]. — Режим доступу URL: https://docs.oracle.com/en/java/

3. PostgreSQL Documentation. [Електронний ресурс]. — Режим доступу: URL: https://www.postgresql.org/docs/

4. Бахрушин В.Є. Методи аналізу даних : навчальний посібник для студентів /В.Є. Бахрушин. – Запоріжжя : КПУ, 2011. – 268 с.

5. Linq. [Електронний ресурс]. — Режим доступу URL: https://www.lingq.com/uk/

6. Babbel. [Електронний ресурс]. — Режим доступу URL: https://www.babbel.com

7. Memrise. [Електронний ресурс]. — Режим доступу URL: https://www.memrise.com

8. Азаров, О. Д. Комп’ютерна схемотехніка : навчальний підручник / [Азаров О. Д., Гарнага В. А., Клятченко Я. М., Тарасенко В. П.] – Вінниця: ВНТУ, 2018. – 230 с.

9. Duolingo. [Електронний ресурс]. — Режим доступу URL: https://www.duolingo.com/learn

10. Spring Documentation. [Електронний ресурс]. — Режим доступу URL: https://docs.spring.io/spring-framework/reference/index.html

11. Вальвано Д. В. Вбудовані системи: Вступ до ARM Cortex-мікроконтролерів / Д. В. Вальвано. — 2012. — 656 с.

12. Bootstrap Documentation. [Електронний ресурс]. — Режим доступу URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/

13. HTML.[Електронний ресурс]. — Режим доступу URL: https://uk.wikipedia.org/wiki/HTML

14. Edward L.Robinson Data Analysis for Scientists and Engineers // Pricenton University Press, 2016, – P.408.

15. Joshua Bloch. Effective Java (2nd Edition). – Addison-Wesley, 2008 – 346 p.

16. Завадський І.О. Основи баз даних. Навчальний посібник. К., 2011. 192 с.

17. CSS. [Електронний ресурс]. — Режим доступу URL: https://uk.wikipedia.org/wiki/CSS

18. JavaScript. [Електронний ресурс]. — Режим доступу URL: https://uk.wikipedia.org/wiki/JavaScript

19. Michael R. Brzustowicz Data Science with Java Practical Method for scientists and engineers /Michael R. Brzustowicz. – O'REILLY, 2017. – 311p.

20. Oracle Documentation. [Електронний ресурс]. — Режим доступу URL: https://www.oracle.com

21. Krok po kroru Polish workbook. URL: https://e-polish.eu/pdf-view/kurs/PKPK1/

22. Lombok Documentation. [Електронний ресурс]. — Режим доступу URL: https://www.baeldung.com/intro-to-project-lombok

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний

університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., проф.. Азаров О.Д.

10.03.2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Кросплатформна інформаційна система для допомоги у вивченні іноземної мови »

08-54.БКР.007.00.000 ТЗ

Керівник роботи к.т.н. доц. каф. ОТ
Крупельницький Л. В.

Студент групи 1КІ–21б
Загубін О.В

ВНТУ 2025

1 Підстава для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність теми полягає в тому, що веб застосунок для домоги у вивченні мови стане у нагоді для багатьох людей з різних сфер — від домогосподарок до представників важливих бізнесів.

1.2 Наказ ВНТУ №97 про затвердження теми БКР від 20.03.2025.

2 Мета БКР і призначення розробки

2.1 Мета роботи — реалізація та аналіз веб застосунку, що сприяє вивченню іноземної мови шляхом інтерактивної подачі навчального матеріалу, структурування інформації за логікою поступового ускладнення тем.

2.2 Призначення розробки — навчання студентів і не тільки основам обраної іноземної мови

3 Вихідні дані для виконання БКР

3.1 Набір базової інформації;
3.2 Принципи технічних вимог;
3.3 Методи підключення бібліотек та фреймворків;
3.4 Середовище розробки програмного забезпечення;

4 Вимоги до виконання БКР

4.1 Провести аналіз предметної області для інформаційної системи
4.2 Розробити веб застосунок;
4.3 Провести тестування системи в кросплатформному середовищі;

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БКР

№ з/п	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз задачі	22.03.25	25.03.25	Огляд джерел, висновки із взаємодії викладачів та студентів
2	Огляд предметної області	28703.25	01.04.25	Розділ 1
3	Обгрунтування обраних інструментів для розробки	04.04.25	15.04.25	Розділ 2
4	Особливості розробки веб застосунку, принцип роботи та підключення баз даних	16.04.25	28.04.25	Розділ 3
5	Тестування системи в різноманітних середовищах	30.04.25	06.05.25	Розділ 4
6	Оформлення пояснювальної записки та презентації	9.05.24	13.05.24	ПЗ, презентація

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКП, графічні і ілюстративні матеріали, протокол попереднього захисту БКП на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БКП українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКП контролюється науковим керівником згідно зі встановленими термінами. Захист БКП відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02- П.001.01:21.

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Кросплатформна інформаційна система для допомоги у вивченні іноземної мови

Тип роботи: бакалаврська кваліфікаційна дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності StrikePlagiarism

Оригінальність 99 Схожість 1

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

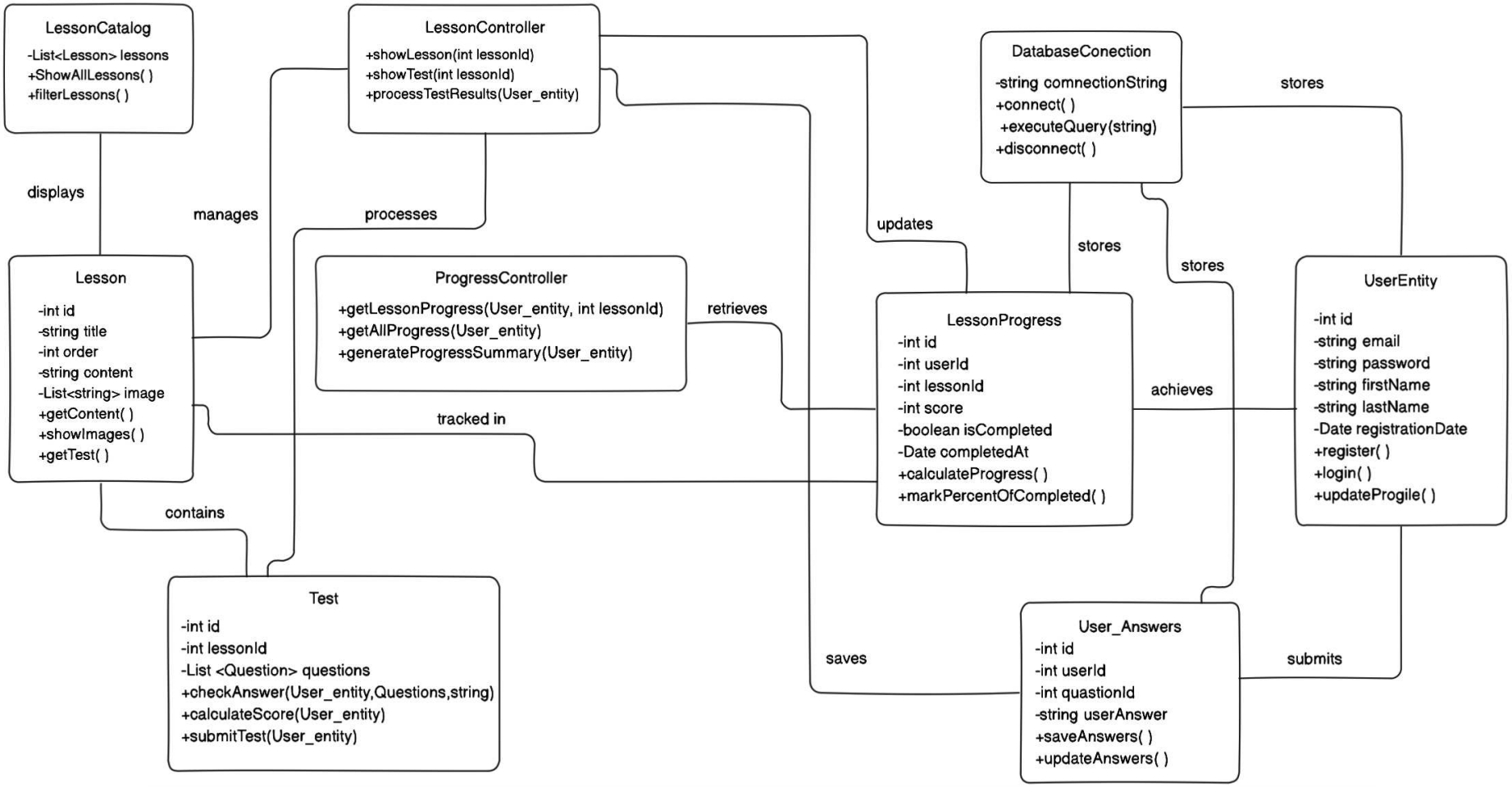
Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism щодо роботи.

Автор роботи _____ Загубін О.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Крупельницький Л. В.
(підпис) (прізвище, ініціали)

ДОДАТОК В
ГРАФІЧНА ЧАСТИНА

КРОСПЛАТФОРМНА ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ДОПОМОГИ У ВИВЧЕННІ ІНОЗЕМНОЇ МОВИ



Додаток В1 — UML діаграма

ДОДАТОК Г
ІЛЮСТРАТИВНА ЧАСТИНА

КРОСПЛАТФОРМНА ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ДОПОМОГИ У ВИВЧЕННІ ІНОЗЕМНОЇ МОВИ

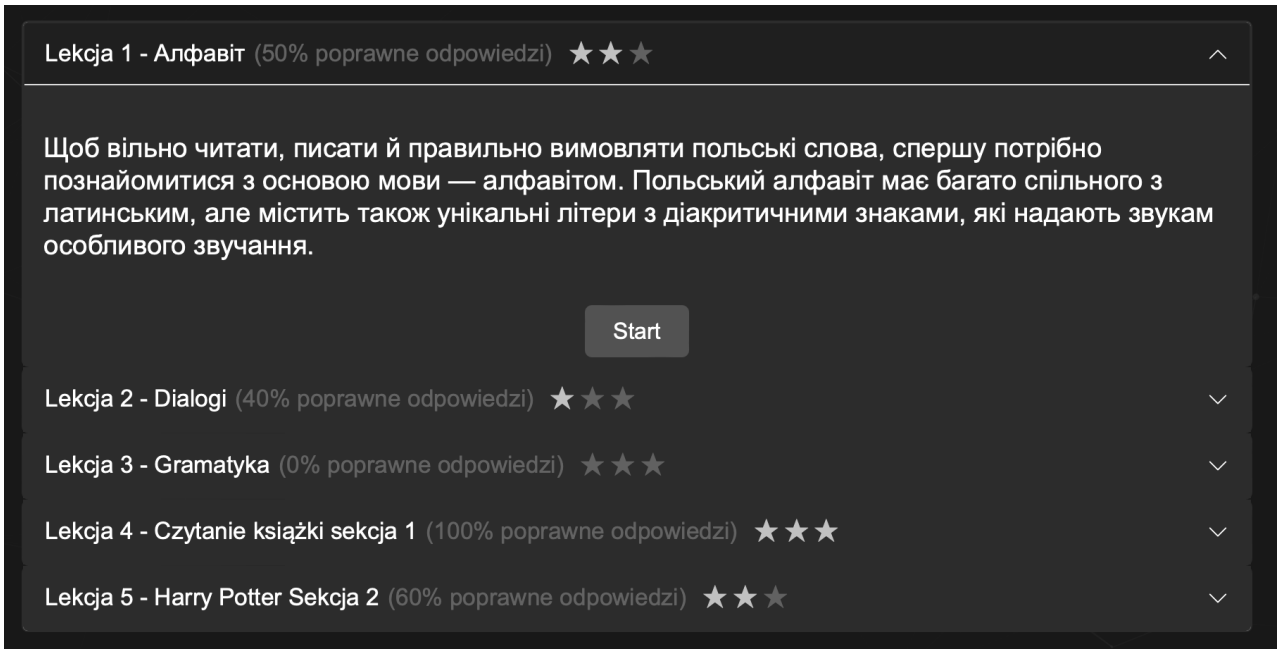
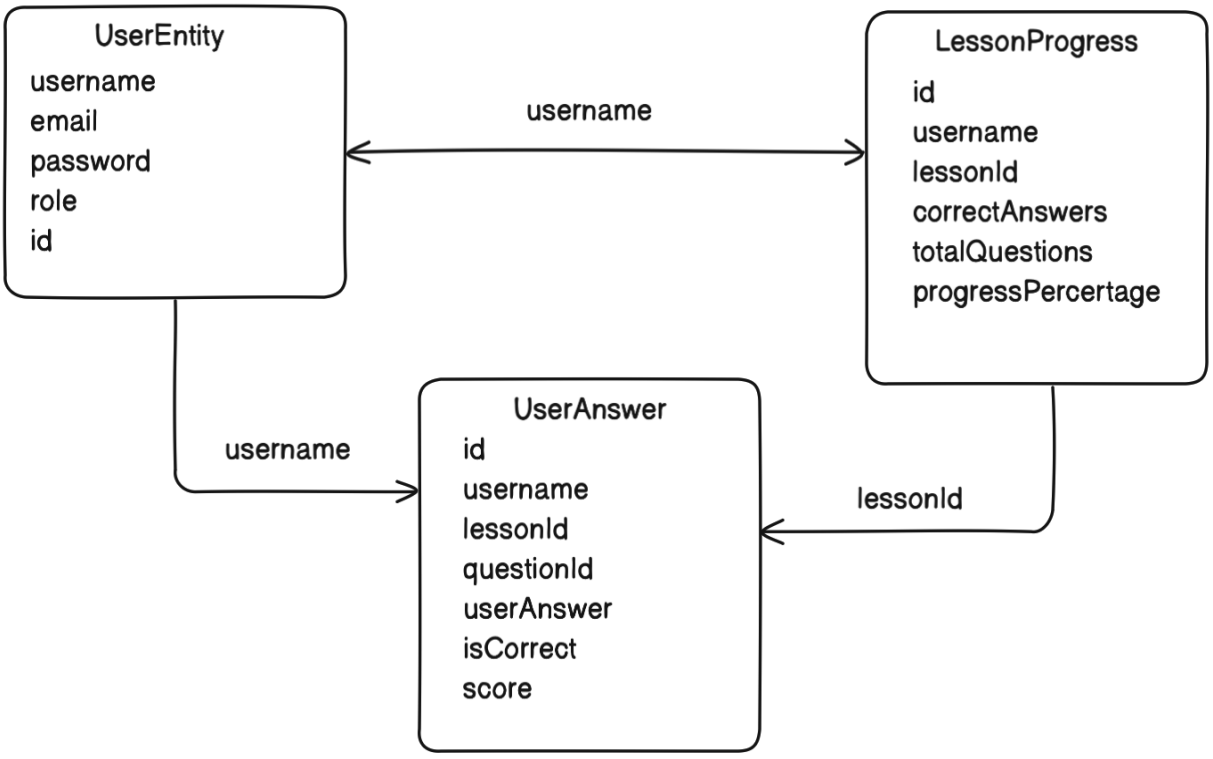


Рисунок Г1 — приклад відображення інтерфейсу користувача



Додаток Г2 — ER-діаграма бази даних

ДОДАТОК Д — ЛІСТИНГ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

MainController.java

```
public class MainController {
    LessonsService lessonsService;
    UserService userService;
    @GetMapping({"/", "/main"})
    public String main(Model model, Principal principal ){
        String username = principal.getName();
        Integer progress1 = lessonsService.getLessonProgressPercentage(username,
"lesson1");
        Integer progress2 = lessonsService.getLessonProgressPercentage(username,
"lesson2");
        Integer progress3 = lessonsService.getLessonProgressPercentage(username,
"lesson3");
        Integer progress4 = lessonsService.getLessonProgressPercentage(username,
"lesson4");
        Integer progress5 = lessonsService.getLessonProgressPercentage(username,
"lesson5");
        model.addAttribute("progress1", progress1);
        model.addAttribute("progress2", progress2);
        model.addAttribute("progress3", progress3);
        model.addAttribute("progress4", progress4);
        model.addAttribute("progress5", progress5);
        model.addAttribute("users", userService.getAll());
        return "/main";
    }
    @GetMapping("/login")
    public String login(){
        return "login";
    }
    @GetMapping("/logout")
    public String logoutPage(HttpServletRequest request, HttpServletResponse
response) {
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (auth != null) {
            new SecurityContextLogoutHandler().logout(request, response, auth);
        }
        return "redirect:/login?logout";
    }
}
```

LessonController.java

```
public class LessonsController {
```

```
        @Autowired
        private LessonsService lessonService;
        @GetMapping("/lessons/lecture1")
        public String lesson1() {
           return "/lessons/lecture1";
        }
        @GetMapping("/lessons/lecture2")
        public String lesson2() {
           return "/lessons/lecture2";
        }
        @GetMapping("/lessons/lecture3")
        public String lesson3() {
           return "/lessons/lecture3";
        }
        @GetMapping("/lessons/lecture4")
        public String lesson4() {
           return "/lessons/lecture4";
        }
        @GetMapping("/lessons/lecture5")
        public String lesson5() {
           return "/lessons/lecture5";}
        @PostMapping("/lesson1/submit")
        @ResponseBody
        public ResponseEntity<?> submitLesson1Answers(@RequestBody
Map<String, String> answers) {
           Authentication SecurityContextHolder.getContext().getAuthentication();
           String username = auth.getName();
           lessonService.processAndSaveAnswers(username, "lesson1", answers);
           return ResponseEntity.ok().build();
        }
        @PostMapping("/lesson2/submit")
        @ResponseBody
        public ResponseEntity<?> submitLesson2Answers(@RequestBody
Map<String, String> answers) {
           Authentication auth =
SecurityContextHolder.getContext().getAuthentication();
           String username = auth.getName();
           lessonService.processAndSaveAnswers(username, "lesson2", answers);
           return ResponseEntity.ok().build();
        }
        @PostMapping("/lesson3/submit")
        @ResponseBody
        public ResponseEntity<?> submitLesson3Answers(@RequestBody
Map<String, String> answers) {
           Authentication SecurityContextHolder.getContext().getAuthentication();
```

```
            String username = auth.getName();
            lessonService.processAndSaveAnswers(username, "lesson3", answers);
            return ResponseEntity.ok().build();
        }
        @PostMapping("/lesson4/submit")
        @ResponseBody
public ResponseEntity<?> submitLesson4Answers(@RequestBody Map<String, String> answers) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
            String username = auth.getName();
            lessonService.processAndSaveAnswers(username, "lesson4", answers);
            return ResponseEntity.ok().build();
        }
        @PostMapping("/lesson5/submit")
        @ResponseBody
public ResponseEntity<?> submitLesson5Answers(@RequestBody Map<String, String> answers) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
            String username = auth.getName();
            lessonService.processAndSaveAnswers(username, "lesson5", answers);
            return ResponseEntity.ok().build();
        }
    }
```

RegistrationController.java

```
    public class RegistrationController {
        UserService userService;
        @GetMapping("/registration")
        public String registration(Model model){
            model.addAttribute("userEntity", new UserEntity());
            return "registration";
        }
        @PostMapping("/registration")
        public String registration(@ModelAttribute("userEntity") UserEntity userEntity,Model model){
            userService.saveUser(userEntity);
            return "redirect:/login";
        }
    }
```