

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Сайт для навчання сліпому методу друку»

Виконав студент групи ПЗк-21-1
ЛИТВИН Володимир Володимирович

Керівник роботи:
КОНЦИДАЙЛО Андрій Михайлович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2024 р.

**ЗАВДАННЯ
на кваліфікаційну роботу**

Здобувач вищої освіти: **ЛИТВИН Володимир Володимирович**

Керівник роботи: **КОНЦИДАЙЛО Андрій Михайлович**

Тема роботи: **«Сайт для навчання сліпому методу друку»,**

затверджена Наказом закладу вищої освіти від «26» лютого 2024 р., №74/с

Вихідні дані для роботи: **методи навчання сліпому методу друку та платформа для навчання сліпому методу друку, їх функціональні можливості та ефективність.**

РЕФЕРАТ

Випускна робота бакалавра складається з комплексної програмної реалізації сайту для навчання сліпому методу друку, а також пояснювальної записки.

Пояснювальна записка до дипломної роботи містить 55 сторінок, 41 ілюстрацію, 15 таблиць, 3 додатки та 13 літературних джерел.

Основною метою даної роботи є розробка алгоритмів та реалізація програмного продукту, що надаватиме широкий спектр інструментів для малювання, маніпуляції зображеннями, створення спеціальних ефектів, колажів.

В роботі визначено ключові завдання, що пов'язані з розробкою сайту, проведено аналіз існуючого інструментарію, подібного до розроблюваного, а також наведено сценарії роботи програмного продукту. Представлено огляд загальної структури додатку, описано алгоритми взаємодії між окремими компонентами, а також наведено опис об'єктної структури (діаграму класів) додатку.

КЛЮЧОВІ СЛОВА: КЛАВІАТУРА, СЛІПІЙ ДРУК, КУРСИ, НАВЧАННЯ, МЕТОДОЛОГІЯ, ПРАКТИКА, ШВИДКІСТЬ ДРУКУ, ЕФЕКТИВНІСТЬ, ТЕХНІКА, ВПРАВИ, СТАТИСТИКА.

					ПЗ.КР.Б – 121 – 24 – ПЗ								
Змн.	Арк.	№ докум.	Підпис	Дата									
Розроб.		Литвин В.В.			Сайт для навчання сліпому методу друку				Літ.	Арк.	Аркушів		
Керівник.		Концидайло А.М.									3	41	
									Житомирська політехніка, група ПЗк-21-1				
Рецензент		Єфремов Ю.М.											
Зав. каф.		Вакалюк Т.А.											

ABSTRACT

The bachelor's thesis consists of a comprehensive software implementation of a website for teaching the blind printing method, as well as an explanatory note. The explanatory note to the thesis contains 55 pages, 41 illustrations, 15 tables, 3 appendices, and 13 references.

The main purpose of this work is to develop algorithms and implement a software product that will provide a wide range of tools for drawing, image manipulation, creating special effects, and collages.

The paper identifies key tasks related to the development of the site, analyzes existing tools similar to the one being developed, and provides scenarios for the operation of the software product. An overview of the general structure of the application is presented, the algorithms of interaction between individual components are described, and a description of the object structure (class diagram) of the application is given.

KEYWORDS: KEYBOARD, BLIND PRINTING, COURSES, TRAINING, METHODOLOGY, PRACTICE, PRINTING SPEED, EFFICIENCY, TECHNIQUE, EXERCISES, STATISTICS.

		<i>Литвин В.В.</i>			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		<i>Концидайло А.М.</i>				
Змн.	Арк.	№ докум	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1 Технічне завдання на розробку системи	9
1.2 Аналіз аналогів програмного продукту	10
1.3. Вибір архітектури.....	13
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	15
1.5. Вибір методології навчання сліпого друкування	17
Висновки до розділу 1	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ ДЛЯ НАВЧАННЯ СЛІПОМУ МЕТОДУ ДРУКУ	20
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	20
2.2. Розробка бази даних системи.....	23
2.3. Проектування та реалізація алгоритмів роботи системи.....	28
2.4 Реалізація сайту для навчання сліпому методу друку.....	30
Висновки до розділу 2	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ САЙТА.....	41
3.1 Порядок встановлення та налаштування системи.....	41
3.2 Порядок встановлення та налаштування параметрів сайту	42
3.2.1 - Функціональні можливості користувача	43
3.2.2 - Функціональні можливості адміністратора	48
3.3. Тестування роботи програмного продукту	51
Висновки до розділу 3	54
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Інтерфейс програмування

БД – база даних

ПЗ – програмне забезпечення

ОС – операційна система

RAM - Random Access Memory (Пам'ять з довільним доступом);

UML – Unified Modeling Language (Уніфікована мова моделювання);

		<i>Литвин В.В.</i>			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		<i>Концидайло А.М.</i>				
Змн.	Арк.	№ докум	Підпис	Дата		6

ВСТУП

У сучасному цифровому світі, де комп'ютери та клавіатури стали необхідними інструментами для багатьох сфер життя, навички швидкого та точного набору тексту набувають особливого значення. Особливо це стосується людей з обмеженими можливостями, зокрема тих, хто користується сліпим методом друку.

Актуальність теми. Навчання сліпому методу друку є важливим навичкою, особливо у сучасному світі, де комп'ютери та клавіатурні навички важливі для багатьох професій. Створення сайту для навчання цієї навички може бути дуже корисним для людей, які хочуть вивчити цей метод або поліпшити свою швидкість та точність друку. Цей проект має на меті надати доступ до ефективних методів навчання, які допоможуть користувачам вдосконалити свої навички та забезпечити їх можливість участі у сучасному інформаційному середовищі.

Метою даної роботи є розробка інтерактивного сайту, який надасть можливість навчатися сліпому методу друку, забезпечуючи зручний доступ до навчального матеріалу та інструментів для практичних вправ.

Завдання роботи включають аналіз існуючих методик навчання сліпому методу друку. Розробка інтерактивного навчального контенту, включаючи вправи та тести. Реалізація інтерфейсу користувача, зручного для навчання та практики. Тестування та вдосконалення сайту на основі отриманих результатів.

Об'єктом дослідження є процес навчання сліпому методу друку, а предметом дослідження є розроблений в рамках роботи сайт та методи його використання.

Предметом дослідження є розробка та впровадження сайту для навчання сліпому методу друку. Це включає аналіз ефективних методик навчання, визначення необхідних функцій і інструментів, а також оцінку користувацького досвіду. Дослідження охоплює вивчення існуючих методик

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		7

і підходів до навчання сліпого методу друку, визначення ключових функцій, які повинні бути реалізовані на сайті, та аналіз технічних аспектів розробки.

Для досягнення поставлених цілей використовуються методи аналізу літературних джерел, педагогічного експерименту, тестування користувачів та статистичний аналіз отриманих даних. Інформаційна база дослідження включає наукові публікації, методичні матеріали з педагогіки та інформаційні технології.

Результати даної роботи мають практичне значення для осіб з візуальними обмеженнями, навчальних закладів та спеціалістів у сфері розробки інтерактивних навчальних платформ.

Робота складається з вступу, аналітичної частини, розділу з розробки сайту, експериментальної частини, висновків та списку використаних джерел.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Технічне завдання на розробку системи

Назва системи – Сайт для навчання сліпому методу друку.

Ціль – розробка та впровадження системи для навчання сліпому методу друку, яка надає інструменти для поліпшення сліпої печаті користувача.

Мета сайту для навчання сліпому методу друку - допомогти користувачам навчитися друкувати всіма пальцями, не дивлячись на клавіатуру. Цей метод друку дозволяє друкувати швидше, точніше та з меншою втратою.

Система повинна задовольняти зазначеним вище вимогам на комп'ютері в наступній мінімальній комплектації:

- Процесор: не менше 1 GHz
- Операційна система: Windows, Linux або macOS
- Пам'ять: 64 Mb
- Дисковий простір: не менше 5 Mb
- Інтернет з'єднання: з доступом до веб-браузера з підтримкою HTML5

Технічні вимоги:

- Масштабованість: система повинна бути здатна підтримувати зростання кількості користувачів.
- Продуктивність: час відгуку для типових задач повинен бути не більше 5 секунд, для складних завдань - не більше 15 секунд.

Інтерфейс користувача: система повинна відповідати стандартам інтерфейсу користувача операційних систем Windows або UNIX та сучасним нормам проектування веб-додатків.

Функціональні вимоги:

Сайт для навчання сліпому методу друку надає адміністраторам такі можливості:

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	9
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		

- Додавання, оновлення, видалення, сортування та пошук, курсів, уроків, завдань та тестів.
- Редагування віртуальної клавіатури.
- Отримання сповіщень про зміни.
- Управління користувачами, зміна ролі користувача, редагування даних.

Нефункціональні вимоги:

- Час навчання: для звичайних користувачів – 20-30 хвилин, для досвідчених - 10 хвилин.
- Швидкість відповіді: максимум 1 секунда на звичайні запити, 3 секунди на складніші запити.
- Інтуїтивний інтерфейс: легкий у використанні, не потребує додаткової підготовки.
- Надійність: стабільна та надійна робота системи без збоїв.
- Доступність: час обслуговування не перевищує 10% загального часу роботи.
- Неперервна робота: безперебійна робота протягом 30 робочих днів.
- Норма помилок: максимум 1 помилка на 1000 запитів користувача.
- Продуктивність: система повинна забезпечувати швидкість відповіді не більше 1-3 секунд, ефективну обробку даних, масштабованість, економне використання ресурсів, оптимізацію запитів, паралельну обробку, надійність та ефективне керування ресурсами.

1.2 Аналіз аналогів програмного продукту

На сьогоднішній день існує багато додатків для навчання сліпому методу друку. Вони пропонують різні підходи, вправи та інтерфейси, щоб допомогти користувачам опанувати цю навичку. Розглянемо найпопулярніші з них і проаналізуємо їхні особливості, переваги та недоліки.

Touch Typing Study [1].

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	10
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		

Touch Typing Study - це безкоштовна платформа, що допомагає освоїти сліпий метод друку. Вона пропонує 15 уроків, тести швидкості, ігри та статистику для відстеження прогресу. Підтримує різні розкладки клавіатури рис (1.1).

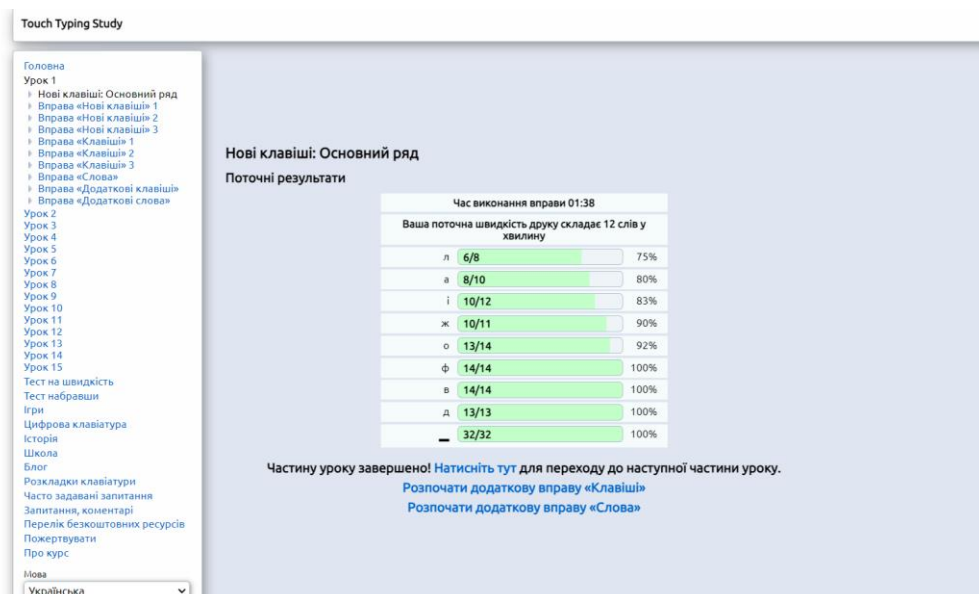


Рисунок 1.1 – Інтерфейс Touch Typing Study

Переваги:

- Поступове проходження уроків
- Вивід статистики після завершення проходження кожного уроку

Недоліки:

- Незручний інтерфейс
- Не зберігається прогрес проходження уроків

Teoset [2]

Платформа Teoset надає користувачам можливість вивчитися сліпий печаті, розвинути навички швидкісного читання, тренувати ментальну арифметику, вивчати іноземні мови та багато іншого. Teoset - це універсальний ресурс для тих, хто хоче поліпшити свої навички навчання та розвитку (рис 1.2).

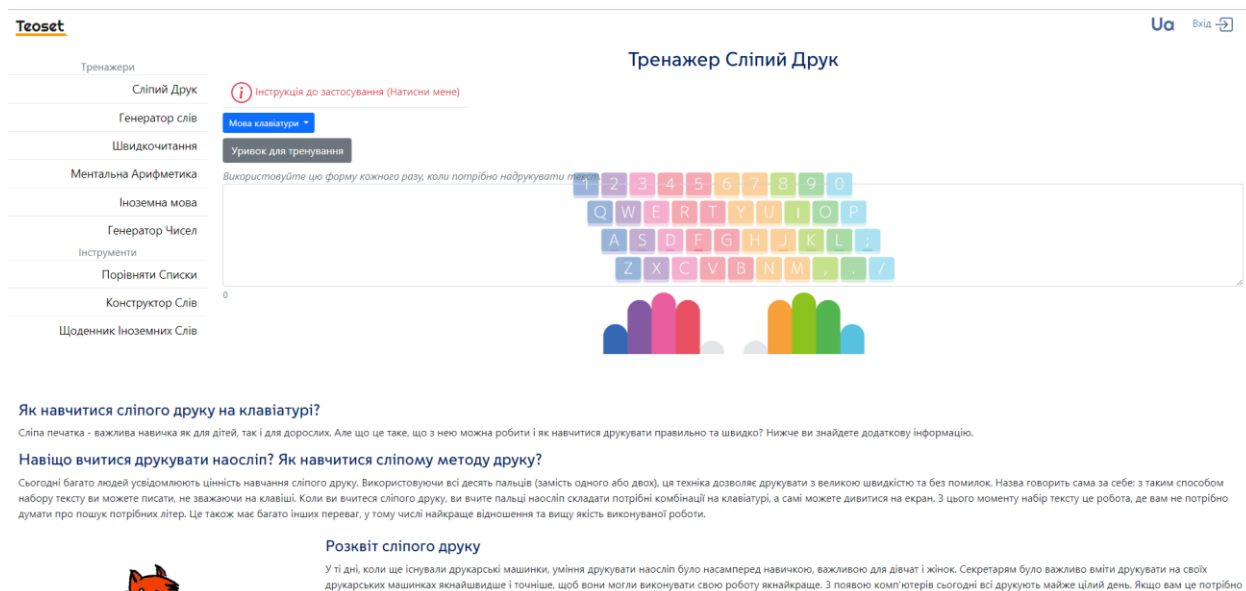


Рисунок 1.2 – Інтерфейс Teoset.

Переваги:

- Підказка правильного положення пальців на клавіатурі
- Можна друкувати згенерований текст
- Можна друкувати власний текст

Недоліки:

- Незручний інтерфейс
- Відсутність уроків та завдань
- Немає лічильника швидкості печаті
- Немає результатів після завершення друку

Ratatype [3]

Дана платформа допомагає освоїти сліпий метод друку. Інтерактивні курси, вправи, тести швидкості, ігри та статистика - все це для ефективного навчання сліпого друку. Підтримуються різні мови та розкладки клавіатури. Ratatype - простий, ефективний і доступний інструмент для освоєння сліпого методу друку та підвищення вашої продуктивності (рис 1.3).



Тест швидкості друку

Пройди тест за 1 хв і дізнайся, як швидко ти друкуєш в курсі
English for Beginners

China was the first country to
introduce paper money (in 812),
but it wasn't until 1661 that a
bank (Banco-Sedlar of Sweden)

🔄 01:00

+ Вимкнути рекламу

Рисунок 1.3 – Інтерфейс ratatype.ua

Переваги:

- Доступні різні курси для навчання
- До кожного курсу є тестування
- Зручний та зрозумілий інтерфейс
- Доступний перегляд статистики навчання

Недоліки:

- Не можна друкувати власний текст

Отже, основними характеристиками сайту для навчання сліпому методу друку повинні бути: зручний та зрозумілий інтерфейс, наявність різноманітних курсів поділених на уроки з завданнями та тестуванням до них, перегляд результатів навчання, підказка у вигляді клавіатури, де кожний палець відповідає певному кольору.

1.3. Вибір архітектури

Для реалізації сайту для навчання сліпому методу друку було обрано клієнт-серверну архітектуру. У обраній архітектурі один комп'ютер (клієнт)

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	13
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		

звертається до іншого комп'ютера (сервера) для отримання ресурсів або послуг.

У клієнт-серверній архітектурі, взаємодія з користувачем та збір введених даних відповідає клієнтській частині. Обробка та зберігання цих даних, а також виконання бізнес-логіки системи - завдання серверної частини (рис 1.4).

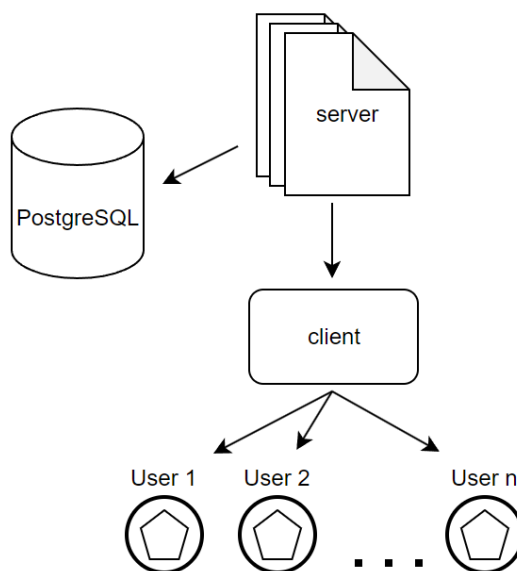


Рисунок 1.4 – Клієнт-серверна архітектура

Клієнт-серверна архітектура була обрана з кількох причин:

- Розділення відповідальності: Ця архітектура дозволяє чітко розділити функціональність між клієнтською та серверною частинами. Клієнтська частина забезпечує інтерфейс користувача та збирає введені дані, тоді як серверна частина відповідає за обробку цих даних та логіку навчання.
- Централізоване зберігання даних: Серверна частина може зберігати дані про прогрес користувачів, їх досягнення, результати навчання тощо. Це дозволяє зберігати та аналізувати дані для покращення ефективності навчання.
- Бізнес-логіка: Сайт включає в себе складну бізнес-логіку, наприклад, аналіз помилок користувачів, обрахунок результатів

користувачів, тощо. Цю логіку краще виконувати на серверній стороні для забезпечення швидкості та ефективності обробки даних.

Отже, клієнт-серверна архітектура відповідає потребам "сайту для навчання сліпому методу друку", забезпечуючи ефективну взаємодію з користувачами та оптимізовану обробку даних та логіки навчання.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення "сайту для навчання сліпому методу друку" були обрані такі інструментальні засоби:

- HTML 5: HTML 5 обрано як основну мову розмітки для створення користувацького інтерфейсу. Вона дозволить створити доступний та зручний інтерфейс для користувачів.
- SCSS: Для стилізації інтерфейсу використовується SCSS, що дозволяє зручно організувати стилі та забезпечити швидке розроблення.
- Сервер Node.js: Node.js обрано як серверну платформу для обробки запитів користувачів та забезпечення взаємодії клієнтської та серверної частин сайту.
- Клієнт React: React використовується для створення динамічного та інтерактивного інтерфейсу для користувачів. Він дозволяє зручно керувати станом додатку та швидко реагувати на події користувача.
- PostgreSQL: PostgreSQL обрано як систему керування базами даних для зберігання інформації про користувачів, їх прогрес у навчанні та інші важливі дані. Це надійна та потужна реляційна база даних.

Щодо вимог до апаратного забезпечення, для ефективної роботи системи рекомендується:

- Сервер Node.js: Достатньою може бути середня продуктивність сервера, яка забезпечить швидке оброблення запитів.

- Клієнтська частина (React): Користувачам рекомендується мати сучасний браузер, що підтримує HTML 5 та JavaScript для оптимального відображення та роботи інтерфейсу.
- База даних PostgreSQL: Необхідно мати сервер або хмарне рішення, яке забезпечить стабільну роботу бази даних та забезпечить необхідну продуктивність для зберігання та обробки даних.

Ці інструменти та вимоги до апаратного забезпечення допоможуть забезпечити стабільну та продуктивну роботу "сайту для навчання сліпому методу друку".

Таблиця 1.1

Порівняння максимального об'єму даних у БД

	БД	Таблиця	Рядок
MS SQL	524 258 TB	524 258 TB	∞
MySQL	∞	256 TB	64 TB
PostgreSQL	∞	32 TB	1.6 TB

Вони всі дозволяють використовувати тригери, процедури та функції.

Таблиця 1.2

Аналіз альтернатив, що підтримуються ОС

	Windows	Linux	Unix	Android
MS SQL	+	-	-	-
MySQL	+	+	+	+
PostgreSQL	+	+	+	-

Порівняємо заходи безпеки для захисту конфіденційної інформації в організаційній системі. Таблиця містить системи безпеки та їх описи.

Таблиця 1.3

Оцінка систем безпеки даних у різних

	Ідентифікація	Шифрування	brute-force	Сертифікація
MS SQL	+	+	-	+
MySQL	+	+	-	-
PostgreSQL	+	+	+	+

Враховуючи порівняльний аналіз різних систем у таблиці, для реалізації проекту було обрано PostgreSQL.

Для створення проекту використовувалися такі програмні засоби: редактор коду WebStorm 2021.3, інструмент для тестування Postman, графічний інтерфейс для керування баз даних PgAdmin 4, графічний редактор Figma, програмне забезпечення для створення діаграм draw.io.

1.5. Вибір методології навчання сліпого друкування

В процесі розробки сайту для навчання сліпого друкування, було проведено аналіз різних методологій, з метою визначення найбільш ефективного підходу для користувачів. Нижче подано порівняльну таблицю, де розглядаються основні методи навчання, їх переваги та недоліки. Цей аналіз допоможе визначити оптимальну методологію для досягнення успіху в навчанні сліпого друкуванню (табл 1.4).

Таблиця 1.4

Методологія	Особливості	Плюси	Мінуси
Традиційне сліпе друкування (10-пальцевий метод)	Використання всіх десяти пальців. Пальці на домашньому ряду (ASDF для лівої руки та JKL; для правої руки).	Висока швидкість друку. Універсальність. Широка підтримка навчальних програм і тренажерів.	Потребує значного часу і практики для освоєння. На початку можливі помилки та дискомфорт.
Метод Колемана (Dvorak Layout)	Альтернативна розкладка клавіатури. Оптимізована для зменшення рухів пальців.	Менше навантаження на пальці та зап'ястя. Потенційно вища швидкість друку.	Потребує переналаштування клавіатури. Менша популярність і підтримка. Переучування.
Програмні тренажери	Використання спеціалізованого ПЗ для навчання (наприклад, TypingClub, Typing.com).	Інтерактивні уроки та вправи. Прогрес відстежується автоматично. Можливість адаптації.	Залежність від доступу до комп'ютера та інтернету. У деяких випадках програми платні.
Ігрові підходи	Використання ігор для навчання (наприклад, TypeRacer, NitroType).	Захопливість і мотивація через ігровий процес. Можливість змагань з іншими користувачами.	Менш структуровані навчальні матеріали. Можуть відволікати від систематичного навчання.

Продовження таблиці 1.4

Методологія	Особливості	Плюси	Мінуси
Гібридні методи	Комбінація різних методів і підходів.	Гнучкість у навчанні. Можливість адаптації під індивідуальні потреби.	Необхідність ретельного підбору і комбінування методів. Можливо, більше часу на навчання.
Мнемонічні техніки	Використання асоціацій і візуальних підказок для запам'ятовування клавіш.	Допомагає швидше запам'ятати розташування клавіш. Підходить для візуальних учнів.	Може не підходити для всіх типів учнів. Обмежена підтримка в навчальних програмах.

Після аналізу різних методологій навчання сліпого друкування було обрано традиційний 10-пальцевий метод як оптимальний варіант для сайту. Його висока швидкість та точність друку, а також широка універсальність і доступність роблять його ефективним засобом навчання та практичного використання навичок сліпого друкування для користувачів будь-якого рівня досвіду та навичок.

Висновки до розділу 1

В даному розділі зазначено, що цільовою метою системи є поліпшення навичок сліпої печаті шляхом надання інструментів для ефективного навчання користувачів.

Аналіз аналогів програмного продукту підкреслює переваги та недоліки існуючих рішень у цій сфері. Для покращення навчального процесу важливо мати зручний інтерфейс, доступ до різноманітних курсів, можливість друкувати власний текст, а також зберігати та відстежувати прогрес користувачів.

У виборі архітектури була обрана клієнт-серверна модель через її можливість розділення відповідальності між клієнтом та сервером, централізоване зберігання даних та ефективне виконання бізнес-логіки.

Для реалізації проекту були обрані відповідні інструментальні засоби: HTML 5 для розмітки, SCSS для стилізації, Node.js як серверну платформу,

React для клієнтської частини, та PostgreSQL як систему керування базами даних.

Згідно з аналізом порівняльної таблиці та вимог до безпеки даних, було обрано PostgreSQL як оптимальне рішення для забезпечення потреб проекту.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		19

РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ ДЛЯ НАВЧАННЯ СЛІПОМУ МЕТОДУ ДРУКУ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Сайт для навчання сліпого методу друку має наступні основні цілі: забезпечити користувачам ефективні методи та інструменти для опанування навичок швидкого та точного друкування на клавіатурі без погляду на клавіші. Це включає інтерактивні вправи, що поступово збільшують складність, візуальні підказки, що орієнтують на правильне розташування пальців, та регулярні тести для відстеження прогресу. Метою є підвищення продуктивності та точності набору тексту, що може бути корисним як у професійній, так і в особистій діяльності користувачів.

Внутрішні користувачі

На сайті буде два типи внутрішніх користувачів: адміністратор і звичайний користувач.

Адміністратор відповідає за управління сайтом, включаючи додавання курсів, уроків, завдань, тестів та користувачів. Звичайний користувач має доступ до навчальних матеріалів і вправ, проходить курси, виконує завдання та тести, а також відстежує свій прогрес у навчанні.

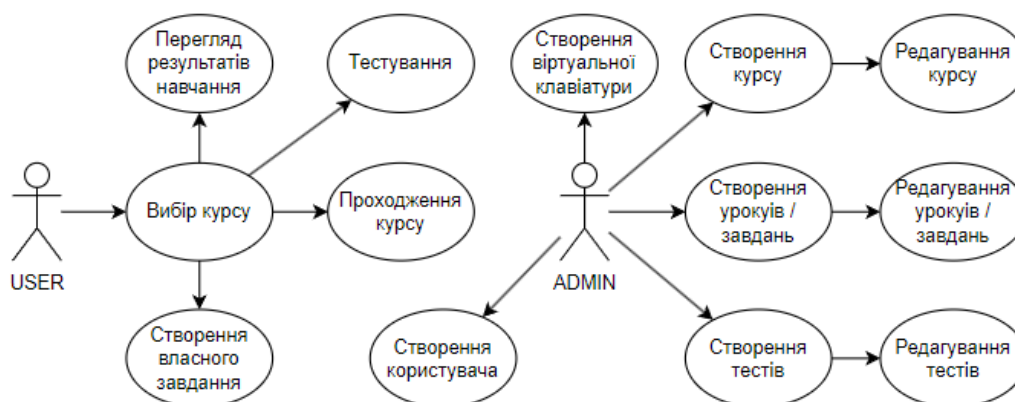


Рисунок 2.1 – Діаграма базових варіантів використання

Специфікація прецедентів для приведена в додатку А.

Адміністраторам платформи, які є внутрішніми користувачами, мають доступ до таких функцій:

1. Можливість створювати нові курси, вносити зміни до існуючих, а також видаляти ті курси, що вже не актуальні. Це дозволяє підтримувати навчальні матеріали в актуальному стані та забезпечувати високу якість освітнього контенту. Додавання нових курсів включає налаштування їх структури, завдань, уроків та тестів, що дозволяє гнучко реагувати на потреби користувачів.
2. Мають доступ до налаштування віртуальної клавіатури, яка відображається користувачам під час виконання завдань. Це включає можливість зміни розташування клавіш, кольору підсвічування та інших візуальних аспектів, що допомагають користувачам швидше орієнтуватися під час виконання вправ. Зручне налаштування віртуальної клавіатури сприяє покращенню навчального процесу та підвищенню ефективності освоєння матеріалу.
3. Можливість додавати нових користувачів, призначати їм ролі, а також змінювати чи видаляти наявні облікові записи. Це дозволяє ефективно контролювати доступ до платформи, забезпечувати належний рівень безпеки та організовувати користувачів відповідно до їхніх потреб та рівня доступу.

Користувачам платформи, які є внутрішніми користувачами, мають доступ до таких функцій:

1. Можливість обирати розкладку клавіатури, яка найбільше відповідає їхнім потребам та вподобанням. Це дозволяє їм адаптувати інтерфейс платформи під себе та створює комфортні умови для навчання та виконання завдань.
2. Можливість проходити різноманітні завдання та уроки, що представлені на платформі. Це дає можливість покращувати свої

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		21

навички, отримувати нові знання та закріплювати вже наявні уміння в зручному темпі та форматі.

3. Можливість проходження тестів, які допомагають внутрішнім користувачам оцінити свої знання та прогрес у навчанні. А також поліпшити вже існуючі здібності сліпої печаті. Тестування дозволяє виявляти сильні та слабкі сторони, а також сприяє підвищенню мотивації до навчання.
4. Можливість перегляду статистики своїх навчальних досягнень. Це включає дані про виконані завдання, результати тестів, загальний прогрес та інші важливі показники, що допомагають користувачам оцінювати власні успіхи та планувати подальше навчання.
5. Можливість створювати власні завдання на платформі. Це дозволяє їм не тільки виконувати запропоновані вправи, але й розробляти індивідуальні завдання, що відповідають їхнім навчальним цілям та інтересам. Це також сприяє творчому підходу до навчання та підвищенню мотивації.

Можливість експлуатації

Система повинна мати простий та зрозумілий інтерфейс, що не вимагає додаткового навчання користувачів. Звичайні користувачі можуть освоїти систему за 20-30 хвилин, а досвідчені - за 10 хвилин. У розробці проекту використовувалися технології Single Page Application (SPA), RESTful API, NodeJS та бібліотека React. Під час аналізу функціональних вимог були визначені основні сутності, необхідні для реалізації системи управління проектами. На діаграмі класів (рис. 2.1) відображено взаємозв'язки між цими сутностями. Детальніша модель бази даних системи буде наведена у розділі 2.2. У третьому розділі розглянута будова інтерфейсу системи, яка буде доступна користувачам.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		22

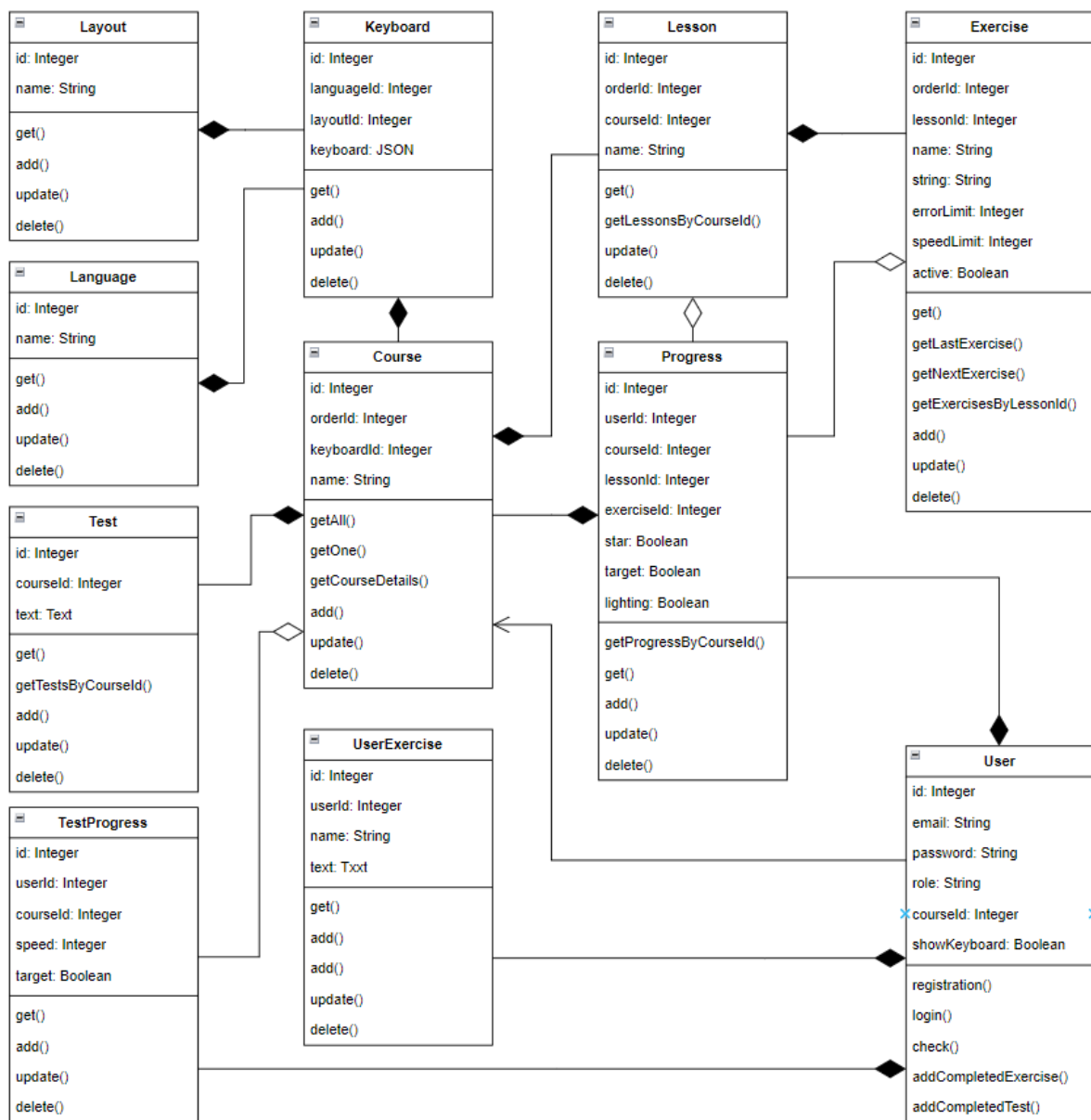


Рисунок 2.2 - Діаграма класів

2.2. Розробка бази даних системи

База даних PostgreSQL, яка була створена для сайту, включає 11 таблиць, що забезпечують зберігання всіх необхідних даних для її роботи.

Таблиця 2.2

Назва таблиці	Призначення
Layout	Таблиця розкладок
Language	Таблиця мов
Keyboard	Таблиця клавіатур

Продовження таблиці 2.2

Course	Таблиця курсів
Lesson	Таблиця уроків
Exercise	Таблиця завдань
Test	Таблиця тестів
Progress	Таблиця результатів проходження курсів
User	Таблиця користувачів
UserExercise	Таблиця завдань користувачів
TestProgress	Таблиця результатів тестів

Нижче подано опис основних таблиць бази даних.

Таблиця «Layout» призначення для зберігання назви розкладок. Структуру таблиці подано нижче.

Таблиця 2.3

Опис полів таблиці «Layout»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
name	STRING	-	-	Назва розкладки

Таблиця «Language» призначення для зберігання доступних мов. Структуру таблиці подано нижче.

Таблиця 2.4

Опис полів таблиці «Language»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
name	STRING	-	-	Назва мови

Таблиця «Keyboard» призначення для зберігання віртуальної клавіатури яку бачить користувач при виконанні завдань. Таблиця зв'язана з таблицями мови та розкладки, та містить об'єкт клавіатури, який складається з клавіш та їх характеристиками. Структуру таблиці подано нижче.

Таблиця 2.5

Опис полів таблиці «Keyboard»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
languageId	INTEGER	-	Language.id	Номер мови
layoutId	INTEGER	-	Layout.id	Номер розкладки
keyboard	JSON	-	-	Об'єкт клавіатури

Таблиця «Course» призначення для зберігання курсів. До кожного курсу прив'язана своя клавіатура. Курси складаються з уроків, уроки складаються з завдань. Структуру таблиці подано нижче.

Таблиця 2.6

Опис полів таблиці «Course»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
orderId	INTEGER	-	-	Порядок курсу
keyboardId	INTEGER	-	Keyboard.id	Номер клавіатури
name	STRING	-	-	Назва кур

Таблиця «Lesson» призначення для зберігання інформації про уроки у курсі. Курс поділений на уроки для полегшення засвоєння матеріалу. Кожен урок має свою тему та завдання. Структуру таблиці подано нижче.

Таблиця 2.7

Опис полів таблиці «Lesson»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
orderId	INTEGER	-	-	Порядок уроку
courseId	INTEGER	-	Course.id	Номер курсу
name	STRING	-	-	Назва уроку

Таблиця «Exercise» призначення для зберігання інформації про завдання. Кожне завдання зв'язане з уроком. Для успішного виконання

завдання, потрібно зробити менше помилок ніж зазначено у полі «errorLimit», та друкувати з більшою швидкістю ніж зазначено у полі «speedLimit».

Структуру таблиці подано нижче.

Таблиця 2.8

Опис полів таблиці «Exercise»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
orderId	INTEGER	-	-	Порядок уроку
lessonId	INTEGER	-	Lesson.id	Номер уроку
name	STRING	-	-	Назва завдання
string	STRING	-	-	Текст завдання
errorLimit	INTEGER	-	-	Ліміт помилок
speedLimit	INTEGER	-	-	Ліміт швидкості
active	BOOLEAN	-	-	Активність завдання

Таблиця «Test» призначення для зберігання тестів. У кожному курсі може бути декілька тестів, вони будуть відображатися по черзі. Користувач повинен повністю ввести текст з тесту, щоб отримати швидкість та точність свого друку. Структуру таблиці подано нижче.

Таблиця 2.9

Опис полів таблиці «Test»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
courseId	INTEGER	-	Course.id	Номер курсу
text	TEXT	-	-	Текст тесту

Таблиця «User» призначення для зберігання інформації про завдання. Кожне завдання зв'язане з уроком. Для успішного виконання завдання, потрібно зробити менше помилок ніж зазначено у полі «errorLimit», та

друкувати з більшою швидкістю ніж зазначено у полі «speedLimit». Структуру таблиці подано нижче.

Таблиця 2.10

Опис полів таблиці «User»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	+	-	Номер
email	STRING	-	-	Пошта
password	STRING	-	-	Пароль
role	STRING	-	-	Роль
courseId	INTEGER	-	Course.id	Номер поточного курсу
showKeyboard	BOOLEAN	-	-	Прапорець показу клавіатури

Таблиця «TestProgress» зберігає інформацію про виконанні тестування користувача, щоб потім користувач зміг подивитися історію своїх результатів. Структуру таблиці подано нижче.

Таблиця 2.11

Опис полів таблиці «TestProgress»

Назва	Тип даних	PK	FK	Опис поля
id	INTEGER	-	-	Номер
userId	INTEGER	-	User.id	Номер користувача
courseId	INTEGER	-	Course.id	Номер курсу
speed	INTEGER	-	-	Швидкість друку
target	INTEGER	-	-	Точність друку

Таким чином створена база даних відповідає вимогам реляційної моделі та забезпечує зручний та ефективний спосіб зберігання та спільного доступу до інформації. Використовуючи таблиці, такі як User, Language, Layout, Keyboard, Course, Lesson, Exercise, Progress, Test, TestProgress та UserExercise, можна керувати даними та організувати їх, надаючи зручний інструментарій для роботи з інформацією про користувачів, курси, уроки,

завдання, тестування та результати навчання. Всього база даних складається з 11 таблиць.

2.3. Проектування та реалізація алгоритмів роботи системи

Оскільки програмний додаток має односторінкову архітектуру (SPA), це передбачає його явне розподілення на серверну і клієнтську частини, які взаємодіють за допомогою RESTful API.

RESTful API, це набір конвенцій та принципів, які дозволяють додаткам ефективно спілкуватися між собою через мережу, зазвичай за допомогою Інтернету. Цей підхід дозволяє створювати зручні та масштабовані веб-сервіси, спрощуючи їхню розробку та інтеграцію з іншими системами.

Можна окреслити такі ключові принципи та характеристики RESTful API:

- ресурси: побудований навколо ресурсів, які зазвичай мають URL-адреси. Кожен ресурс ідентифікується унікальним URL, і клієнти взаємодіють з цими ресурсами за допомогою HTTP-запитів до відповідних URL.
- HTTP методи: RESTful API використовує стандартні HTTP методи для виконання різноманітних операцій над ресурсами. Методи, що застосовуються при розробці додатку:
 1. GET: Запитує дані ресурсу для отримання їхнього представлення.
 2. POST: Створює новий ресурс шляхом надсилання даних серверу.
 3. PUT: Оновлює існуючий ресурс або створює його, якщо він не існує.
 4. PATCH: Змінює лише певну частину існуючого ресурсу.
 5. DELETE: Видаляє вказаний ресурс з сервера.

У додатку, який створюється, з клієнтської частини відправляються запити до приватного API для отримання необхідних даних. Сервер на Node.js відповідає JSON-об'єктом на відповідний ендпоінт, на який з

клієнтської частини надходить запит через бібліотеку Http. На сервері здійснюється отримання даних з бази даних і виконуються необхідні маніпуляції з ними. Ці дані потім використовуються для створення користувацького інтерфейсу. Якщо користувач змінює запит сторінки, знову відправляється запит до серверу веб-додатку.

На рисунку 2.7 показано загальний процес функціонування додатку.

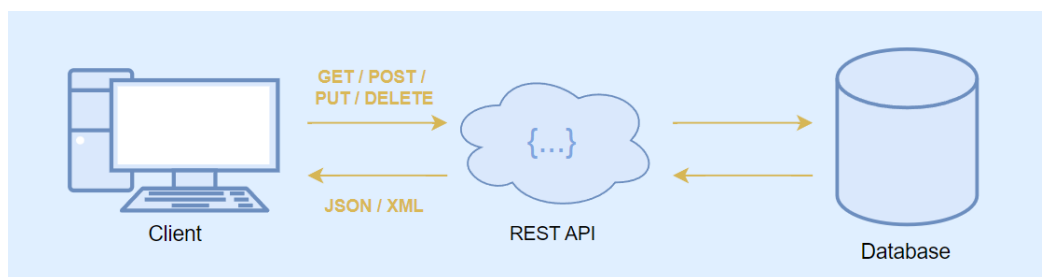


Рисунок 2.3 - Загальний процес функціонування сайту

Сам сайт можна умовно розділити на декілька модулів. Для наочності створимо діаграму компонентів додатку для цих модулів.

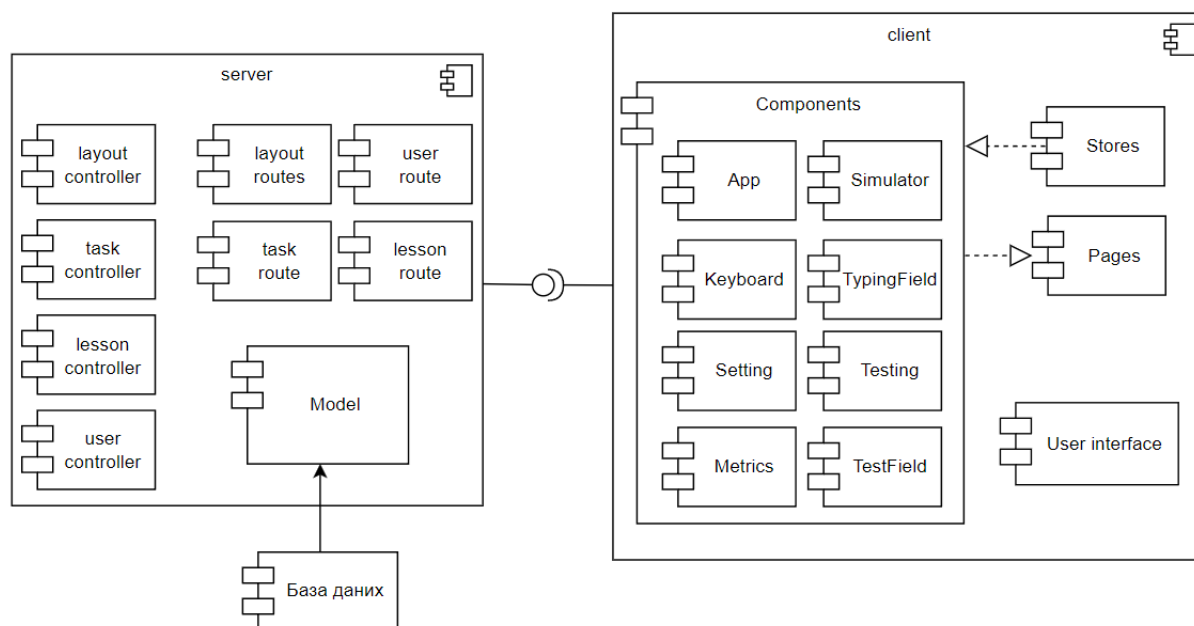


Рисунок 2.4 - Діаграма компонентів

Концепція діаграм компонентів у мові UML дозволяє візуалізувати взаємодію різних компонентів у системі. Ця структурна діаграма показує організацію та зв'язки між компонентами, такими як програмні модулі, бібліотеки, файли конфігурації, бази даних і сервіси.

Детально розглянемо алгоритм роботи модулів компаній і проектів. З цією метою створено діаграму послідовності.

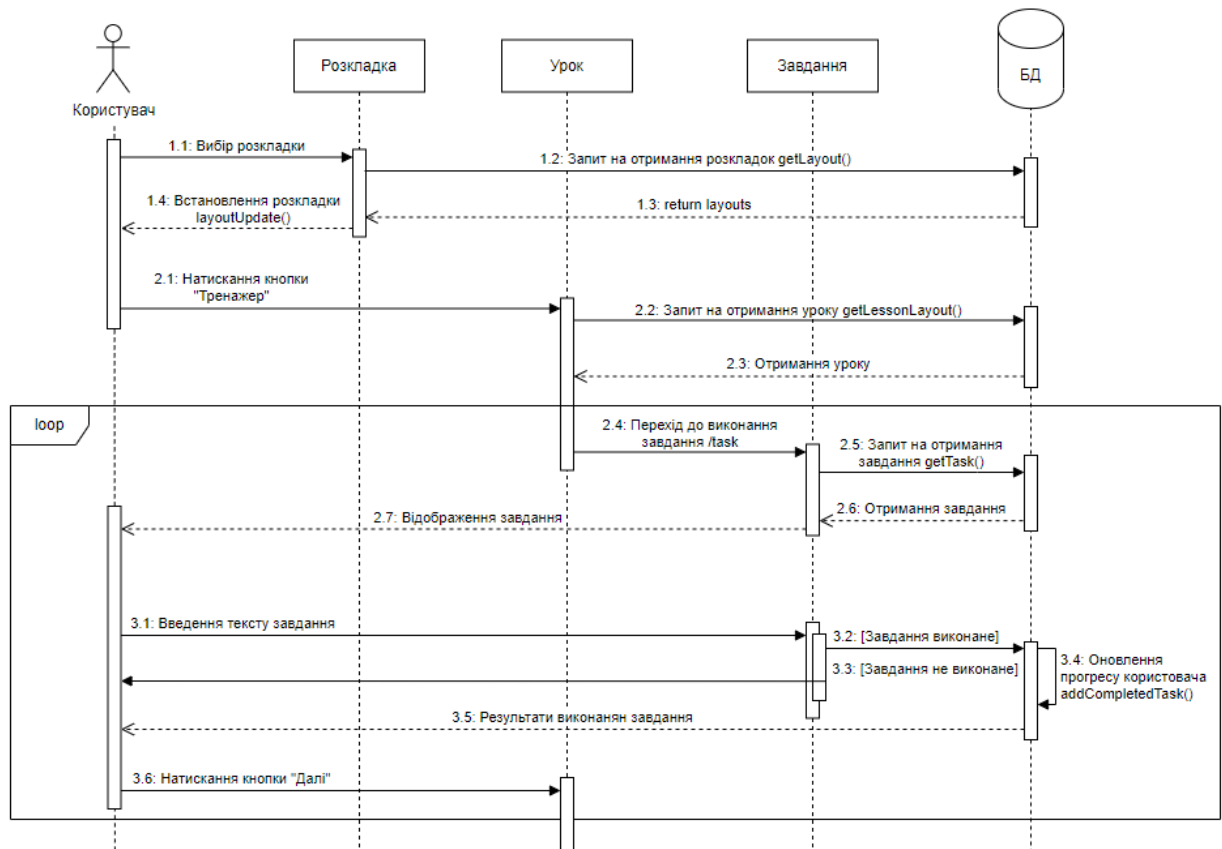


Рисунок 2.5 - Діаграма послідовності виконання завдання

Процес виконання завдань починається з вибору курсу. Після вибору курсу користувач може розпочати процес навчання. Натиснувши кнопку «Тренажер», відкриється сторінка з завданням. Відразу відкривається останнє завдання останнього уроку, щоб користувачеві не довелося шукати, де він зупинився. Щоб перейти до наступного завдання, потрібно пройти попереднє з необхідною швидкістю та точністю введення тексту. Після виконання завдання в базі даних зберігається прогрес користувача.

2.4 Реалізація сайту для навчання сліпому методу друку

Процес розробки сайту розпочався з впровадження механізму авторизації користувачів. Нижче детально описані методи registration (рис. 2.6) та login (рис 2.7), які забезпечують ці функції.

```

async registration(req, res, next) {
  try {
    const {username, email, password} = req.body

    if (!email || !password) {
      return next(ApiError.badRequest("Невірний email або пароль"))
    }

    const candidate = await Users.findOne({where: {email}})

    if (candidate) {
      return next(ApiError.internal("Користувач з таким email вже існує"))
    }
    const hashPassword = await bcrypt.hash(password, 3)
    const user = await Users.create({username, email, password: hashPassword})

    const token = generateJWT(user.id, user.username, user.email, user.role,
user.courseld, user.lastExerciseld, user.showKeyboard)

    return res.json({token})
  } catch (error) {
    return next(ApiError.internal(`Помилка про реєстрації ${error}`))
  }
}

```

Рисунок 2.6 - Метод registration класу UserController

Метод registration обробляє запити на реєстрацію нових користувачів.

Він працює за наступним алгоритмом:

1. Отримання даних з запиту: Зчитуються електронна адреса та пароль з тіла запиту.
2. Перевірка наявності даних: Якщо електронна адреса або пароль не задані, повертається помилка з відповідним повідомленням.
3. Перевірка існування користувача: Перевіряється, чи існує користувач з такою електронною адресою. Якщо так, повертається помилка.
4. Хешування пароля: Пароль хешується для збереження у базі даних у безпечному вигляді.
5. Створення нового користувача: У базі даних створюється новий запис користувача з хешованим паролем.
6. Генерація JWT токена: Генерується JWT токен, який містить ідентифікатор користувача, його електронну адресу, роль, ID курсу, ID останньої вправи та налаштування клавіатури. Токен використовується для автентифікації при подальших запитах.

7. Відправлення токена клієнту: Токен відправляється у відповіді на запит.

Авторизація користувача

```
async login(req, res, next) {
  try {
    const {email, password} = req.body

    if (!email || !password) {
      return next(ApiError.badRequest("Невірний email або пароль"))
    }

    const user = await Users.findOne({where: {email}})

    if (!user) {
      return next(ApiError.internal("Користувача з таким email не знайдено"))
    }

    let comparePassword = bcrypt.compareSync(password, user.password)
    if (!comparePassword) {
      return next(ApiError.internal("Невірний пароль"))
    }
    const token = generateJWT(user.id, user.username, user.email, user.role,
user.courseld, user.lastExerciseld, user.showKeyboard)
    return res.json({token})
  } catch (error) {
    return next(ApiError.internal("Помилка про авторизації"))
  }
}
```

Рисунок 2.7 - Метод login класу UserController

Метод login обробляє запити на авторизацію користувачів. Його алгоритм наступний:

1. Отримання даних з запиту: Зчитуються електронна адреса та пароль з тіла запиту.
2. Перевірка наявності даних: Якщо електронна адреса або пароль не задані, повертається помилка.
3. Перевірка існування користувача: Перевіряється, чи існує користувач з такою електронною адресою. Якщо ні, повертається помилка.
4. Перевірка пароля: Пароль перевіряється на відповідність збереженому хешу. Якщо паролі не співпадають, повертається помилка.

5. Генерація JWT токена: Генерується JWT токен, який містить ідентифікатор користувача, його електронну адресу, роль, ID курсу, ID останньої вправи та налаштування клавіатури.
6. Відправлення токена клієнту: Токен відправляється у відповіді на запит.

JWT (JSON Web Token) є стандартом для створення безпечних токенів, які можуть бути використані для автентифікації. Токен містить закодовану інформацію про користувача, що дозволяє серверам ідентифікувати його без необхідності зберігання стану на сервері.

Після створення контролерів для реєстрації та входу користувачів було налаштовано маршрутизацію за допомогою Express.js. Маршрути /registration та /login обробляють POST-запити, викликаючи відповідні методи контролера користувача (рис 2.8).

```
const Router = require('express')
const router = new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')
const checkRole = require("../middleware/checkRoleMiddleware");

router.post('/registration', userController.registration)
router.post('/login', userController.login)
```

Рисунок 2.8 – Маршрутизатор userRoutes

Маршрут /auth перевіряє аутентифікацію користувача за допомогою middleware authMiddleware який використовується для перевірки аутентифікації користувача перед виконанням певних дій або доступом до захищених ресурсів у системі. Цей middleware перевіряє наявність аутентифікаційних даних у запиті (наприклад, токен або сесійні дані), щоб підтвердити ідентифікацію користувача перед виконанням запиту (рис. 2.9).

```

const jwt = require("jsonwebtoken")

module.exports = function (req, res, next) {
  if (req.method === "OPTIONS") {
    next()
  }
  try {
    const token = req.headers.authorization.split(' ')[1]
    if (!token) {
      res.status(401).json({message: 'Не авторизований'})
    }
    const decoded = jwt.verify(token, process.env.SECRET_KEY)
    req.user = decoded
    next()
  } catch (e) {
    res.status(401).json({message: 'Не авторизований'})
  }
}

```

Рисунок 2.9 – Реалізація middleware authMiddleware

Маршрут `/delete` використовує middleware `checkRole('ADMIN')` для перевірки, що користувач є адміністратором, перед викликом методу `delete` з контролера користувача. Таким чином, ці маршрути дозволяють контролювати доступ до певних функцій системи на основі ролей та аутентифікації користувачів (рис. 2.10).

```

const jwt = require("jsonwebtoken")

module.exports = function (role) {
  return function (req, res, next) {
    if (req.method === "OPTIONS") {
      next()
    }
    try {
      const token = req.headers.authorization.split(' ')[1]
      if (!token) {
        res.status(401).json({message: 'Не авторизований'})
      }
      const decoded = jwt.verify(token, process.env.SECRET_KEY)
      if (decoded.role !== role) {
        return res.status(403).json({message: 'Немає доступу'})
      }
      req.user = decoded
      next()
    } catch (e) {
      res.status(401).json({message: 'Не авторизований'})
    }
  }
}

```

Рисунок 2.10 – Реалізація middleware CheckRoleMiddleware

Для обробки помилок у застосунку використовується спеціалізований клас, який дозволяє зручно створювати та управляти різними типами помилок (рис. 2.11).

```

class ApiError extends Error {
  constructor(status, message) {
    super();
    this.status = status
    this.message = message
  }

  static badRequest(message) {
    return new ApiError(404, message)
  }

  static internal(message) {
    return new ApiError(500, message)
  }

  static forbidden(message) {
    return new ApiError(403, message)
  }
}

module.exports = ApiError

```

Рисунок 2.11 – Клас ApiError

Клас ApiError є важливим компонентом мого сайту. Я використовую його в middleware та контролерах для обробки помилок, які виникають під час взаємодії з API. При виникненні помилки визивається конструкція `next(new ApiError(status, message))`, щоб створити та передати відповідну помилку далі по ланцюжку middleware. Також я використовую статичні методи `badRequest`, `internal` і `forbidden` для генерації конкретних типів помилок з попередньо визначеними статусами та повідомленнями. Це допомагає мені ефективно управляти обробкою помилок у моєму сайті та надає зручний механізм для їхнього створення та передачі на клієнтську сторону.

Далі розглянемо процес створення завдання, щоб зрозуміти його кроки та послідовність дій. Починаючи зі створення клавіатури, курсу та уроку, ми можемо ефективно підготувати й організувати завдання для користувачів. Початковим етапом є створення або додавання вже готової клавіатури для визначення мови та розкладки клавіш, яка буде використовуватися користувачами. Після цього створюється курс, який включає в себе певну послідовність уроків. Кожен урок, у свою чергу, має назву та пов'язаний із певним курсом. Після створення уроків, додаються конкретні завдання для

кожного уроку, які включають текст завдання, обмеження на швидкість та кількість помилок, а також активність самого завдання (рис. 2.12).

```

async add(req, res, next) {
  const {languageId, layoutId, keyboard} = req.body
  try {
    if (!languageId || !layoutId || !keyboard) {
      return next(ApiError.badRequest('Неверно задані параметри'));
    }
    const newKeyboard = await Keyboards.create({languageId, layoutId, keyboard})

    return res.json(newKeyboard)
  } catch (error) {
    return next(ApiError.internal('Помилка при додаванні нової мови'))
  }
}

```

Рисунок 2.12 – Додавання клавіатури

Метод add класу KeyboardController додає нову клавіатуру. Адміністратор створює клавіатуру, вказуючи мову, розкладку та об'єкт самої клавіатури, яку буде бачити користувач під час проходження завдання (рис. 2.13).

```

async add(req, res, next) {
  try {
    const {keyboardId, name} = req.body

    if (!keyboardId || !name) {
      return next(ApiError.badRequest('Неверно задані параметри'));
    }

    const course = await Courses.create({keyboardId, name})

    return res.json(course)
  } catch (error) {
    return next(ApiError.internal('Помилка при створенні курсу'))
  }
}

```

Рисунок 2.13 – Створення курсу

У кожного курсу буде своя клавіатура з визначеною мовою та розкладкою, яка буде використовуватися під час проходження завдань у цьому курсі. Такий підхід дозволяє налаштовувати клавіатуру для кожного курсу з урахуванням специфіки та потреб користувачів у відповідній мові та розкладці клавіш (рис. 2.14).

```

async add(req, res, next) {
  try {
    const {courseId, name} = req.body

    if (!courseId || !name) {
      return next(ApiError.badRequest('Неверно задані параметри'));
    }

    const lesson = await Lessons.create({courseId, name})
    return res.json(lesson)
  } catch (error) {
    return next(ApiError.internal('Помилка при додаванні нового уроку'))
  }
}

```

Рисунок 2.14 – Створення уроку

Метод add класу LessonController відповідає за додавання нового уроку до бази даних. Уроки в курсах потрібні для зручного розподілення тем та організації навчального процесу для користувачів. Кожен урок може містити в собі певну тему чи завдання, що дозволяє структурувати та послідовно вивчати матеріал в межах курсу (рис. 2.15).

```

async add(req, res, next) {
  try {
    const {lessonId, name, string, speedLimit, errorLimit, active} = req.body

    const exercise = await Exercises.create({lessonId, name, string, speedLimit, errorLimit,
    active})
    return res.json(exercise)
  } catch (error) {
    return next(ApiError.internal('Помилка при додаванні нового завдання'))
  }
}

```

Рисунок 2.15 – Створення завдання

У методі add класу ExerciseController виконується додавання нового завдання до бази даних. Вказуються такі параметри як номер уроку, до якого відноситься завдання (lessonId), його назва (name), текст завдання (string), обмеження на швидкість набору тексту (speedLimit), обмеження на кількість помилок (errorLimit), а також активність завдання (active). Цей процес дозволяє створювати та організовувати різноманітні завдання для навчальних уроків у курсі.

Після додавання завдання користувач може його виконати. Для цього він вибирає відповідний курс та починає виконувати завдання. Після виконання завдання користувачу відображаються три іконки: зірочка (якщо

кількість помилок менше, ніж вказано в параметрі `errorLimit`), ціль (якщо завдання виконане без помилок) та блискавка (якщо швидкість набору тексту вища, ніж вказано в `speedLimit`).

На рисунку 2.16 зображено метод для додавання виконаного завдання.

```
async addCompletedExercise(req, res, next) {
  const {id} = req.params
  const {courseId, lessonId, exerciseId, star, target, lightning} = req.body

  try {
    const progress = await Progress.findOne({
      where: {
        userId: id,
        courseId,
        lessonId,
        exerciseId
      }
    })

    if (progress) {
      if (!progress.star && star) progress.star = true;
      if (!progress.target && target) progress.target = true;
      if (!progress.lightning && lightning) progress.lightning = true;
      await progress.save();
      return res.json(progress);
    }

    const newProgress = await Progress.create({
      userId: id, courseId, lessonId, exerciseId, star, target, lightning
    })
    return res.json(newProgress)
  } catch (err) {
    return next(ApiError.internal('Помилка при додаванні виконаного звдання'))
  }
}
```

Рисунок 2.16– Метод `addCompletedExercise` класу `UserController`

Для того щоб завдання було визнане як виконане, досить отримати зірочку. Якщо користувач бажає покращити свій результат, він може пройти завдання знову. При повторному проходженні завдання минулі результати не погіршуються, це зроблено для мотивації користувача і надихає його на постійне покращення своїх навичок. Кожена спроба може стати кроком у напрямку досягнення більш високого рівня майстерності.

На рисунку 2.17 зображений метод для `getProgressByCourse`

```

async getProgressByCourse(req, res, next) {
  const {id, courseId} = req.body

  if (!id) {
    return next(ApiError.badRequest("Не заданий id користувача"))
  }

  const progress = await Progress.findAll({where: {userId: id, courseId}})

  if (!progress) {
    return next(ApiError.internal("Користувач не почав проходити курс"))
  }

  return res.json(progress)
}

```

Рисунок 2.17– Метод getProgressByCourse класу userController

Метод getProgressByCourse призначений для обробки запиту на отримання прогресу користувача для конкретного курсу. Він перевіряє наявність ідентифікатора користувача у вхідних даних. Якщо ідентифікатор відсутній, метод відправляє помилку про неправильний запит. Далі метод шукає прогрес користувача для вказаного курсу та ідентифікатора користувача. Якщо прогрес знайдено, він повертається у відповідь. Якщо ж прогрес не знайдено, метод відправляє помилку, вказуючи на відсутність прогресу для цього курсу та користувача.

Висновки до розділу 2

У цьому розділі описується проектування, розробка та реалізація сайту для навчання сліпому методу друку.

Сайт має односторінкову архітектуру (SPA) та поділений на серверну і клієнтську частини, які взаємодіють за допомогою RESTful API. Для зберігання інформації використовується PostgreSQL база даних, яка містить 11 таблиць. Сайт пропонує широкий спектр функцій для користувачів та адміністраторів, включаючи реєстрацію, авторизацію, вибір курсів та уроків, виконання завдань, проходження тестів, перегляд статистики прогресу, створення та редагування контенту.

Сайт реалізований на Node.js, React, Express.js та PostgreSQL. Описано алгоритми роботи модулів сайту, включаючи процес виконання завдань, авторизацію користувачів, створення та редагування контенту.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		40

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ САЙТА

3.1 Порядок встановлення та налаштування системи

Діаграма розгортання наочно показує, як розміщені та налаштовані обробні вузли й компоненти системи під час її розгортання. Цей тип структурних діаграм використовується для моделювання фізичних аспектів об'єктно-орієнтованих систем, включаючи топологію апаратних ресурсів. Діаграма розгортання показує розміщення компонентів системи на обчислювальних вузлах і їх взаємодію (рис. 3.1).

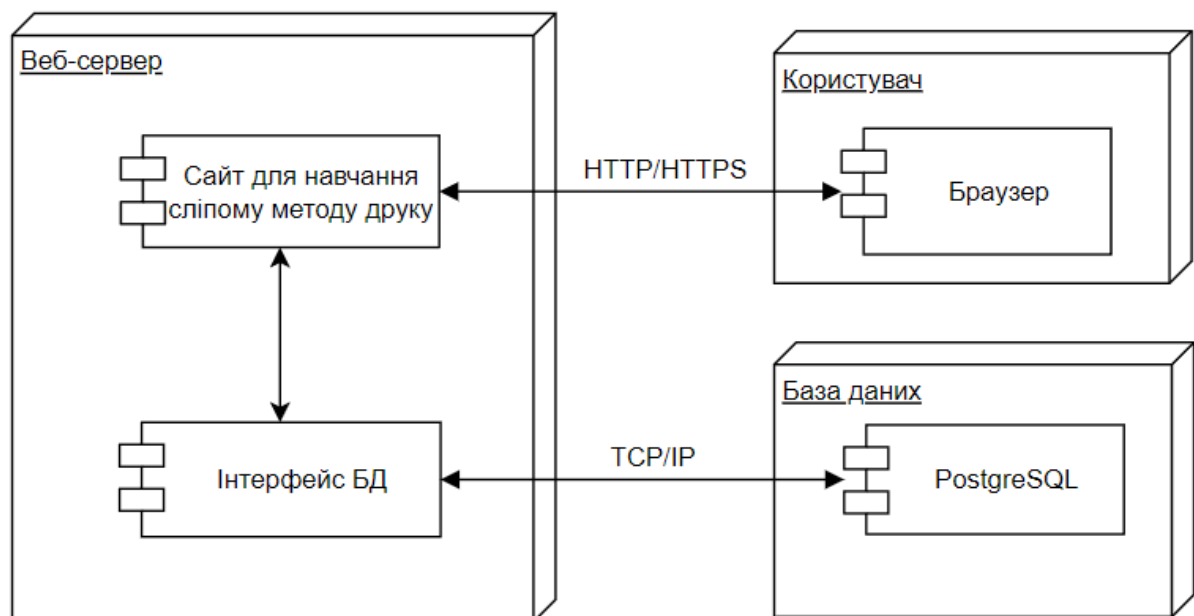


Рисунок 3.1– Діаграма розгортання сайту

Ця діаграма розгортання зображує конфігурацію системи навчання сліпому методу друку, яка складається з трьох основних компонентів: веб-сервера, бази даних та користувача. Веб-сервер містить два компоненти: сайт для навчання сліпому методу друку, який забезпечує основний функціонал навчання для користувачів, та інтерфейс БД, що взаємодіє з базою даних для отримання та зберігання необхідної інформації. База даних складається з одного компонента – PostgreSQL, який використовується для зберігання даних, необхідних для функціонування системи навчання. Користувач

використовує браузер для доступу до сайту навчання сліпому методу друку через HTTP/HTTPS протоколи. Взаємодія між компонентами здійснюється таким чином: користувач взаємодіє з сайтом для навчання через браузер за допомогою HTTP/HTTPS, а сайт для навчання взаємодіє з інтерфейсом бази даних, який у свою чергу обмінюється даними з PostgreSQL через протокол TCP/IP. Ця діаграма показує, як різні частини системи взаємодіють між собою для забезпечення функціональності навчання сліпому методу друку.

3.2 Порядок встановлення та налаштування параметрів сайту

При відкритті за стосунку користувач бачить головну сторінку.

На рисунку 3.2 зображено головну сторінку

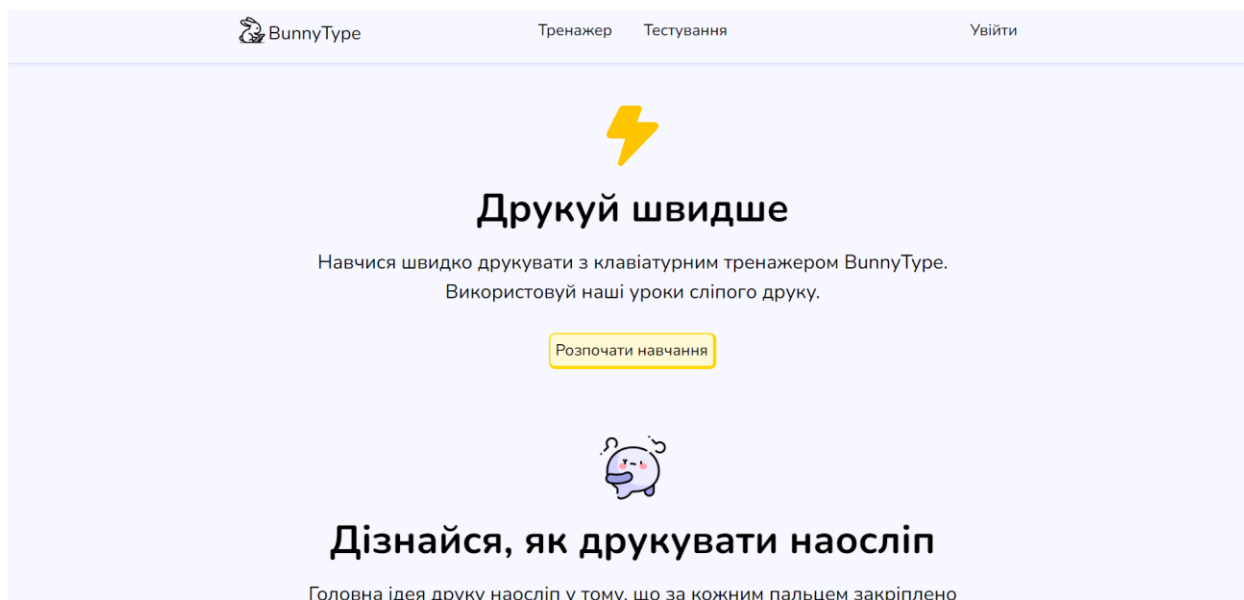


Рисунок 3.2 – Головна сторінка

Для того щоб розпочати навчання сліпому друку, необхідно пройти авторизацію. Після входу в веб-застосунок користувач бачить сторінку авторизації, де потрібно ввести свої облікові дані. Авторизація відкриває доступ до всіх навчальних матеріалів та вправ.

На рисунку 3.3.1 та 3.3.2 можна побачити форму авторизації.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		42

Рисунок 3.3.1 – Форма входу

Рисунок 3.3.2 – Форма реєстрації

Після авторизації користувач автоматично перенаправляється на сторінку вибору курсу. Якщо користувач вже зареєстрований і раніше проходив навчання, система автоматично перекидає його на сторінку з завданнями вибраного курсу, щоб він міг продовжити навчання з того місця, де зупинився.

3.2.1 - Функціональні можливості користувача

Після входу в систему користувач має можливість обрати курс. На сторінці вибору курсу будуть відображені інформація про назву курсу, мову,

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		43

розкладку та прогрес проходження, якщо користувач вже раніше пройшов цей курс або зараз проходить його.

На рисунку 3.4 зображено сторінку вибору курсу

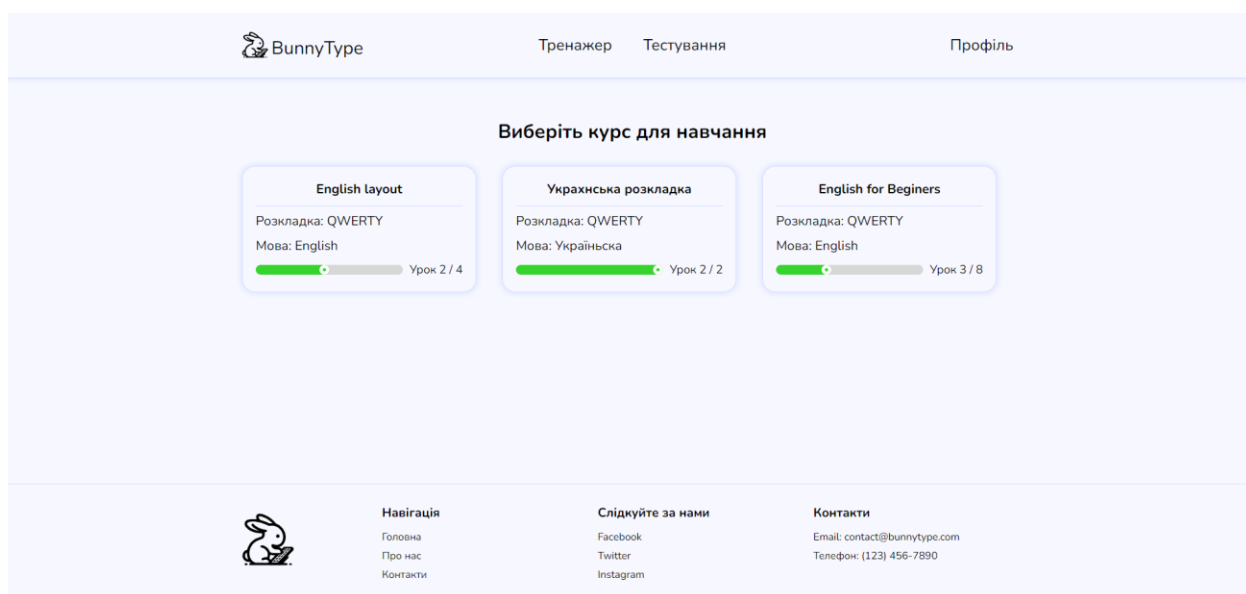


Рисунок 3.4 – Сторінка вибору курсу

Після вибору курсу користувач буде перенаправлений на сторінку з завданням, яке він зможе одразу почати виконувати. На сторінці з завданням буде присутнє поле для введення тексту завдання та допоміжна клавіатура, яка допоможе користувачеві легше вводити текст.

На рисунку 3.5.1, 3.5.2 зображено сторінку завдання та його виконання.

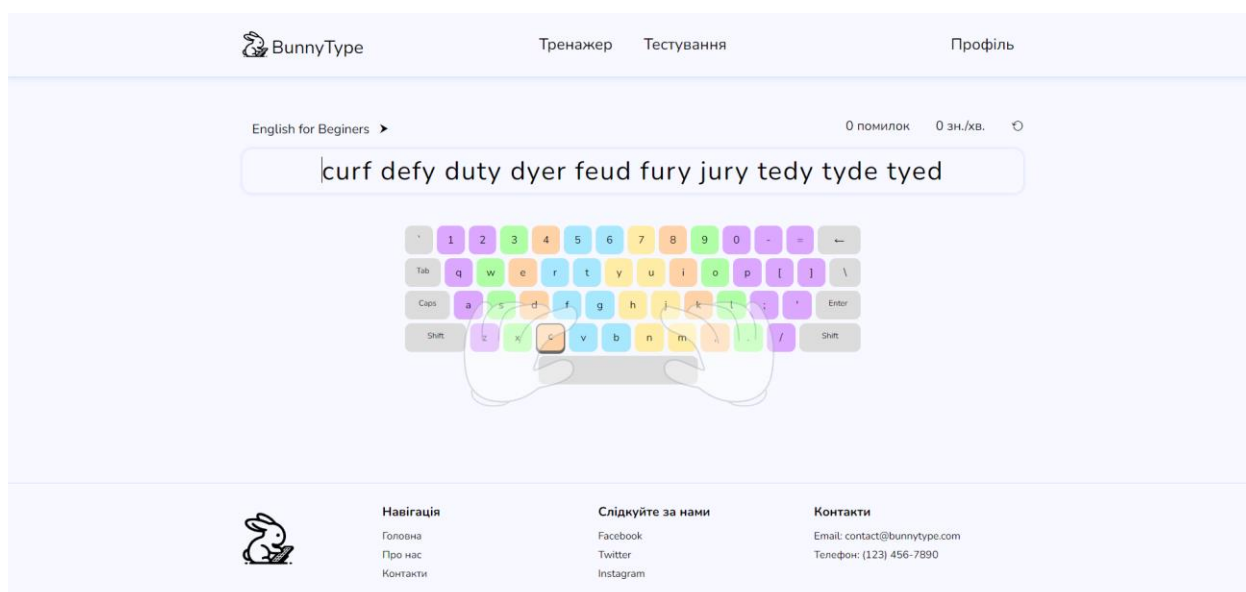


Рисунок 3.5.1 – Сторінка завдання

Під час проходження завдання, якщо користувач правильно натискає на літеру, курсор зміщується, а літера стає зеленою. Над полем введення буде показана статистика виконання завдання, така як точність та швидкість друку.



Рисунок 3.5.2 – Процес проходження завдання

Після завершення завдання користувач побачить свій результат. Якщо він виконав умову для переходу далі, він зможе продовжити; в іншому випадку йому доведеться виконувати завдання ще раз.

На рисунку 3.6 показано результати виконання завдання.

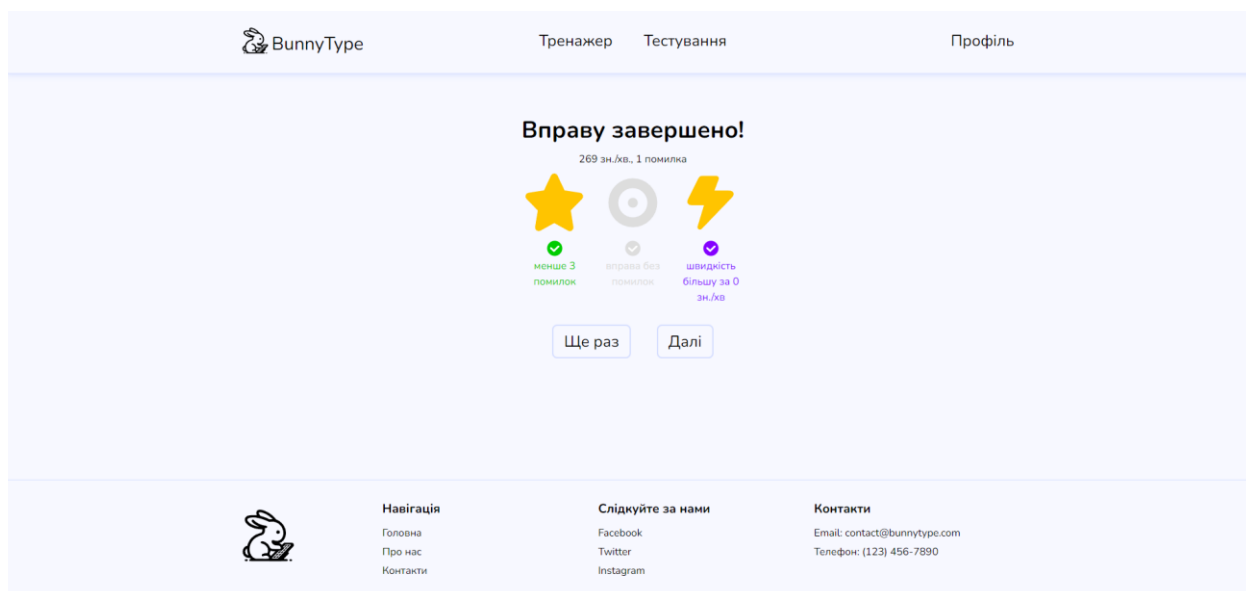


Рисунок 3.6 – Результати виконання завдання

Користувач також може переглядати завдання, які він виконав, і якщо він захоче покращити результат, він зможе пройти завдання ще раз. Зберігається лише кращий результат.

На рисунку 3.7 зображено список всіх завдань курсу.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		45



Рисунок 3.7 – Список завдань курсу

У кожному курсі є вкладка "Тестування", де користувач може пройти тести. Після переходу на цю вкладку, йому потрібно буде ввести текст для отримання результату. У кожному курсі може бути багато текстів для тестування, вони будуть відображатися по черзі.

На рисунку 3.8 зображено сторінку тестування.

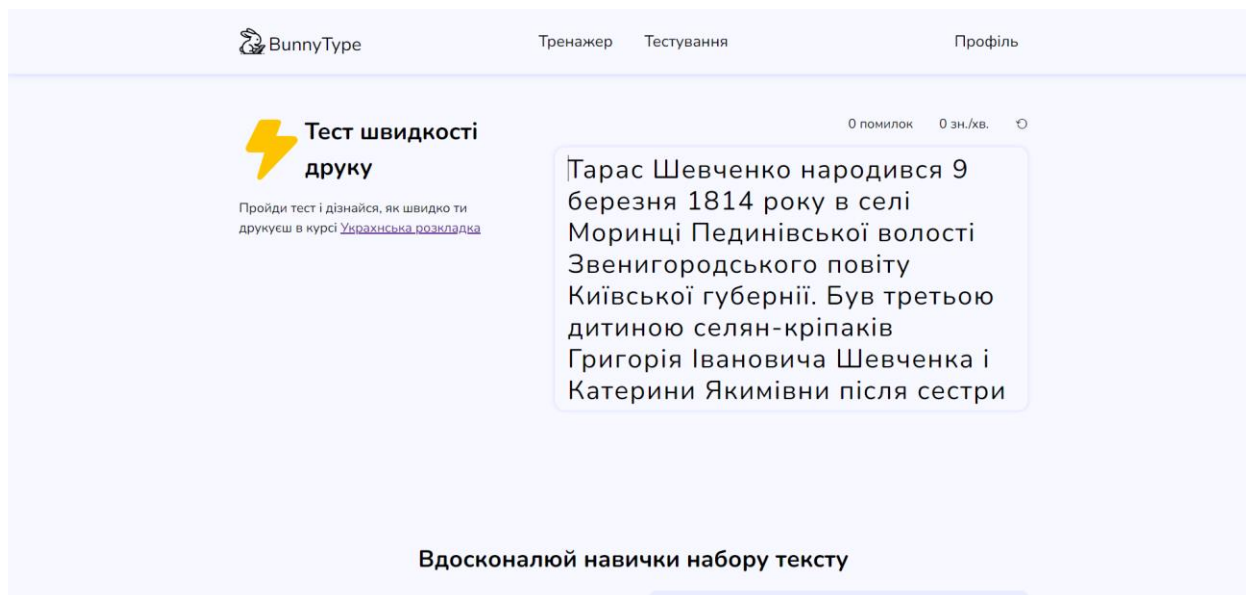


Рисунок 3.8 – Сторінка тестування

Після проходження тесту користувачу буде показано результат, включаючи точність друку та швидкість друку.

На рисунку 3.9 зображено результат виконання тесту.

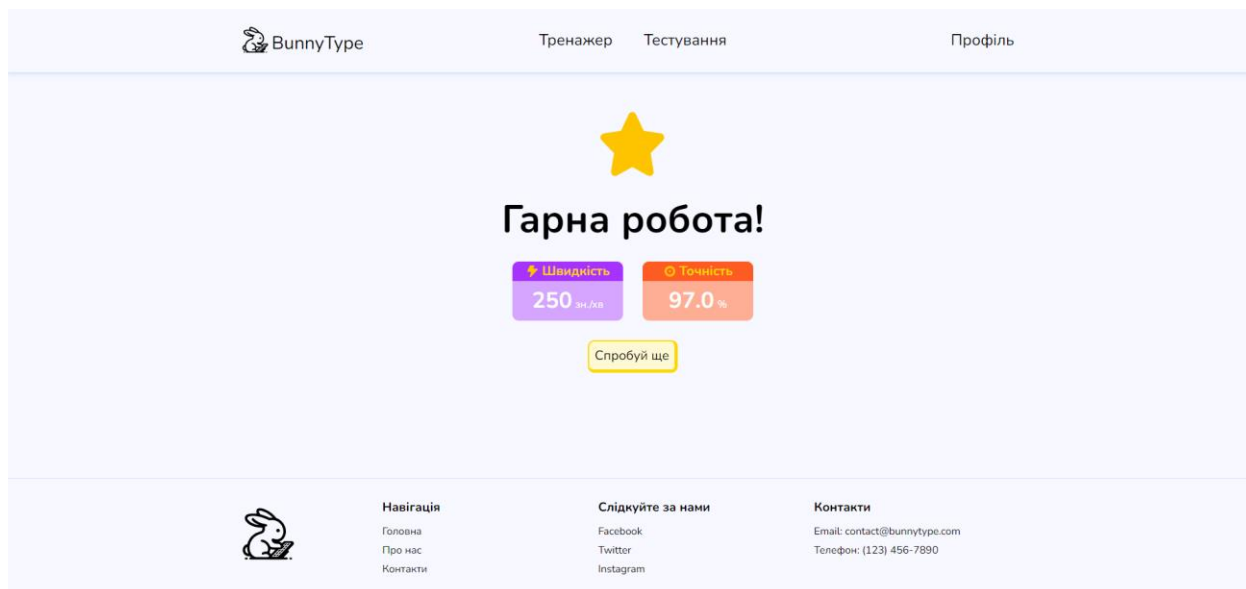


Рисунок 3.9 – Результат виконання тесту

Після завершення завдань або тестів користувач може переглядати свої результати, перейшовши у свій профіль.

На сторінці результатів тестів користувач може побачити графік, на якому відображено його прогрес за кожен період часу, а також таблицю з результатами. Біля результату швидкості виводиться медаль, яка відображає рівень досягнення користувача: срібло, золото, платина. Чим кращий результат, тим вище рівень медалі.

На рисунку 3.10 зображені результати тестування.

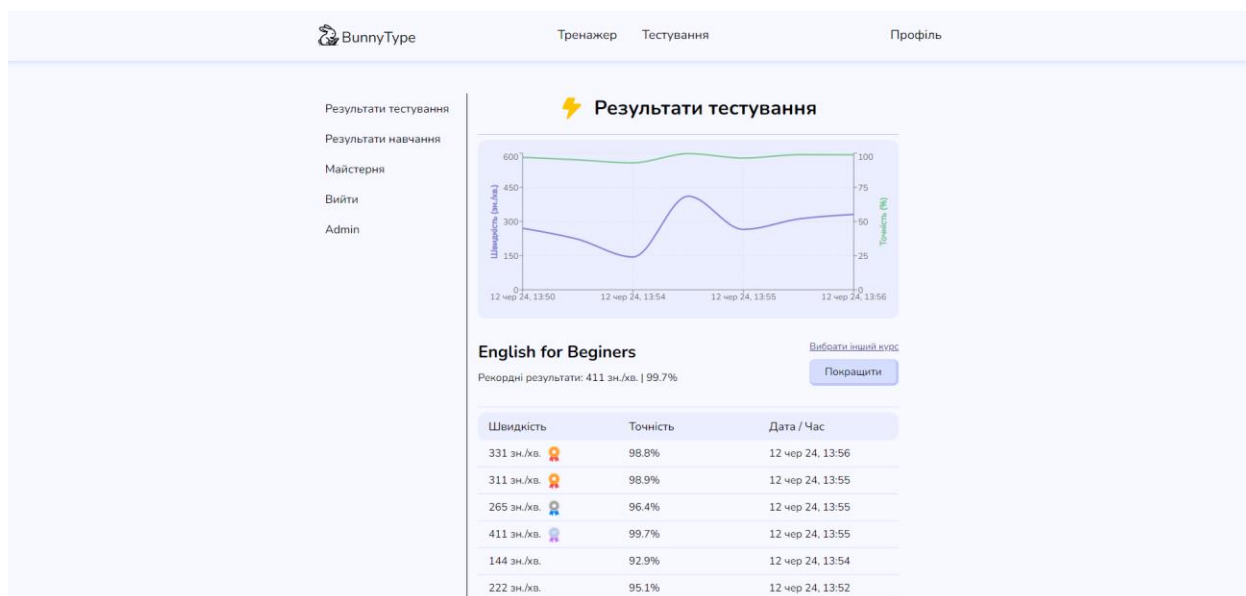


Рисунок 3.10 – Результати тестування

На сторінці результатів навчання користувач може побачити прогрес виконання курсу, а також результати виконання кожного завдання в цьому курсі.

На рисунку 3.10 зображені результати навчання.

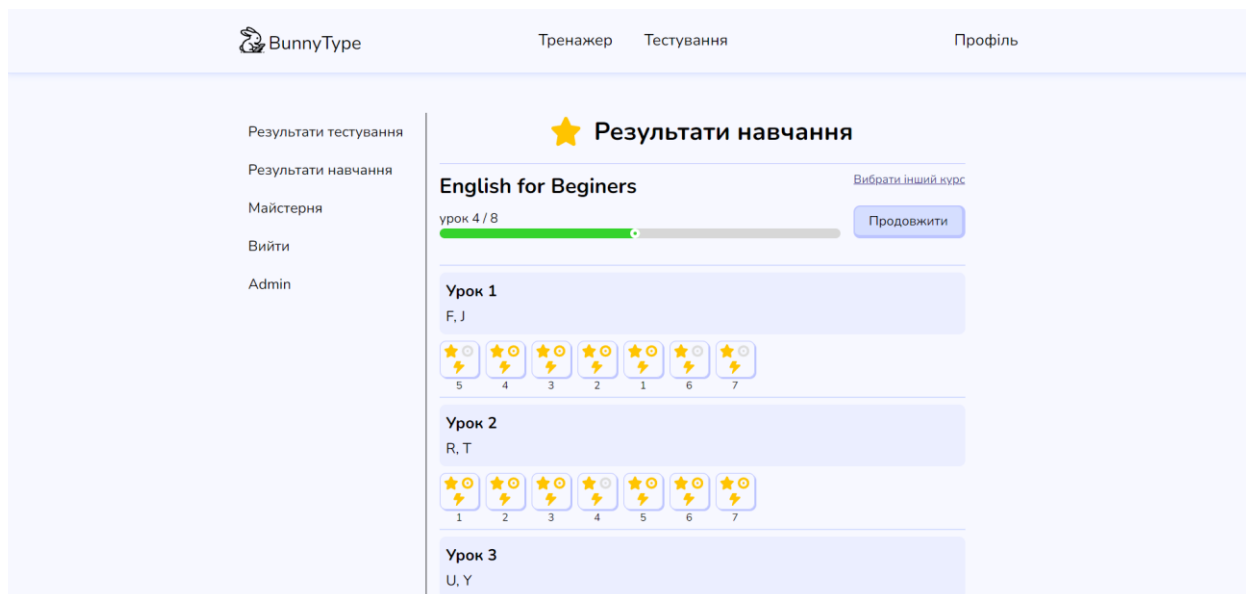


Рисунок 3.11 – Результати навчання

Також на сторінці з результатами користувач може змінити курс, щоб переглянути результати інших курсів.

3.2.2 - Функціональні можливості адміністратора

Адміністратори мають додаткове поле "Admin" у своєму профілі. Перейшовши до цього поля, вони отримують доступ до адміністративної панелі, яка відкриває можливості для керування контентом і користувачами на сайті. У цій панелі адміністратори можуть редагувати мови, що використовуються на платформі, та налаштовувати розкладки клавіатур. Крім того, адміністративна панель дозволяє створювати, редагувати та видаляти курси, підтримуючи актуальність навчальних матеріалів. Адміністратори також можуть керувати профілями користувачів, налаштовувати права доступу модерації контенту.

На рисунку 3.12 зображена панель адміністрування сайту.

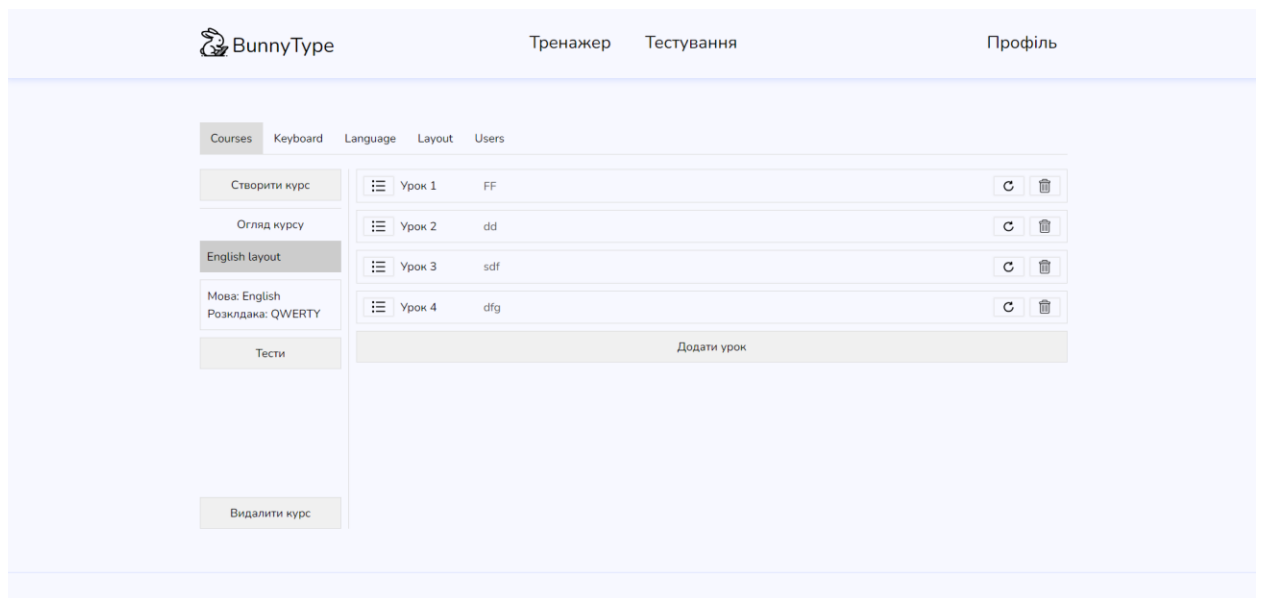


Рисунок 3.12 – Панель адміністрування сайту

Для додавання завдань до курсу потрібно створити нову клавіатуру або підключити вже існуючу. Клавіатура складається з мови та розкладки.

На рисунку 3.13 зображено налаштування клавіатури.

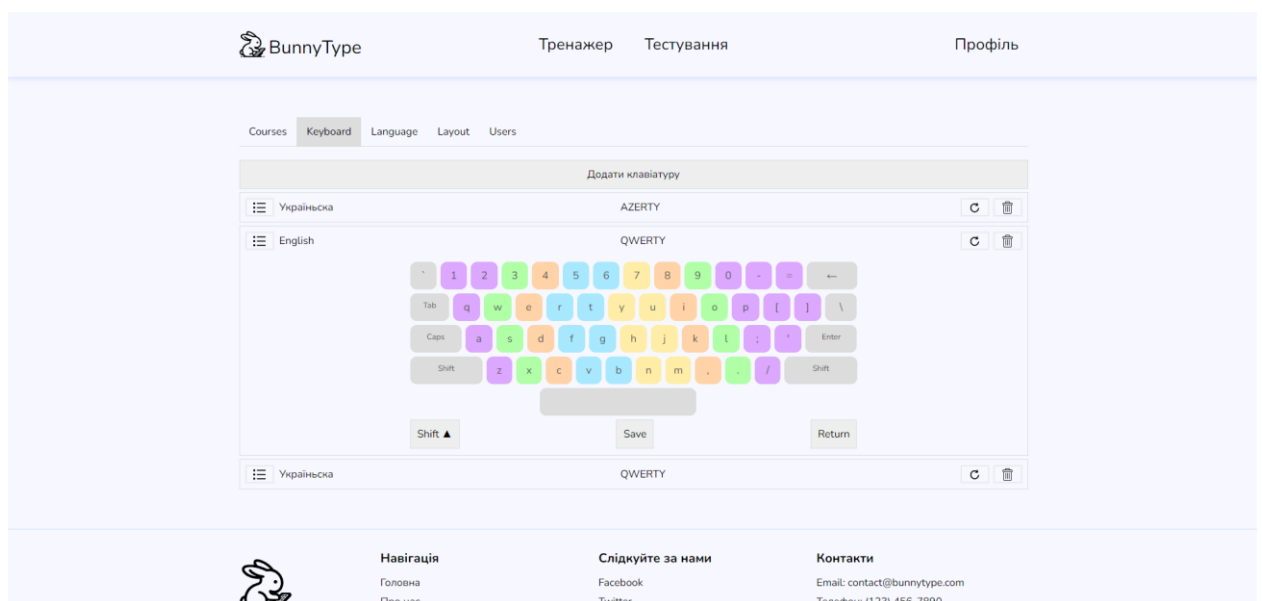


Рисунок 3.12 – Налаштування клавіатури

Для створення курсу потрібно лише вказати назву курсу та вибрати клавіатуру, яка містить певну мову та розкладку.

На рисунку 3.13 зображено створення курсу.

Рисунок 3.13 – Створення курсу

Після створення курсу адміністратор може додавати уроки. Для цього потрібно натиснути кнопку "Додати урок" і вказати назву уроку.

На рисунку 3.14 зображено додавання уроку.

Рисунок 3.14 – Додавання уроку

Коли урок створено, до нього можна додати завдання. Для цього потрібно розгорнути урок, натиснувши на кнопку перед його назвою, і натиснути на "+", щоб додати завдання. При додаванні завдання необхідно вказати назву, текст завдання, ліміт помилок та швидкість виконання завдання.

На рисунку 3.15 зображено додавання завдання

Рисунок 3.15 – Додавання завдання

3.3. Тестування роботи програмного продукту

Розробка будь-якого програмного продукту неминуче супроводжується виникненням недоліків, тому без обов'язкового тестування неможливо забезпечити високу якість додатка. Тестування включає перевірку різних аспектів програми, таких як правильність обробки вхідних і вихідних даних, адекватність роботи функцій та алгоритмів, а також відновлення роботи після помилок. Основна мета тестування полягає у виявленні та усуненні недоліків програмного продукту перед його випуском на ринок або в експлуатацію. Це забезпечує високу якість і надійність програмного забезпечення, знижуючи ризики виникнення помилок у реальних умовах.

Спочатку, створимо набір тестових сценаріїв. Тестовий сценарій - це документ, який містить послідовність дій або умов для перевірки роботи певної частини програмного продукту. Ці сценарії допоможуть перевірити різні аспекти розроблюваної системи. Результати тестування будуть представлені у формі таблиці (табл. 3.1).

Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка авторизація з коректними даними	1. Перейти на сторінку http://localhost:8080/login 2. Ввести email 3. Ввести пароль 4. Натиснути “Увійти”	Email: q@gmail.com Пароль: Bunny1	Відбудеться вхід в систему	Як і очікувалося	Pass
Перевірка авторизація з некоректними даними	1. Перейти на сторінку http://localhost:8080/login 2. Ввести email 3. Ввести пароль 4. Натиснути “Увійти”	Email: q@gmail.com Пароль: NotBunny1	Не відбудеться входу в систему	Як і очікувалося	Pass
Перевірка вибору курсу	1. Увійти в систему 2. Перейти до сторінки з курсами 3. Вибрати курс	Наявний список курсів	Курс зміниться та зберігеться у системі	Як і очікувалося	Pass
Перевірка виконання завдання	1. Увійти в систему 2. Натиснути “Тренажер” 3. Виконати завдання 4. Натиснути “Профіль” 5. Натиснути “Результати навчання”	Наявний заповнений курс	Виконане завдання з’явиться у профілі	Як і очікувалося	Pass
Перевірка виконання тесту	1. Увійти в систему 2. Натиснути “Тестування” 3. Пройти тест 4. Натиснути “Профіль” Натиснути “Результати тестування”	Наявні тести у курсі	Результат виконаного тесту з’явиться у профілі	Як і очікувалося	Pass
Перевірка створення курсу	1. Увійти в систему у ролі адміністратора 2. Натиснути “Профіль” 3. Натиснути “Admin” 4. Натиснути “Створити курс” 5. Ввести назву курсу 6. Вибрати клавіатуру 7. Натиснути “Створити”	Назва курсу: Test Клавіатура: English Layout	Створений курс з’явиться у списку з курсами	Як і очікувалося	Pass

Після аналізу таблиці можна зробити висновок, що всі наведені тестові випадки відповідають заданим очікуванням. Також потрібно протестувати обмеження деяких функцій для користувачів. Для цього скористаємося Postman, який дозволяє перевіряти наші API.

На рисунку 3.16 зображено спробу створення курсу без надання необхідних параметрів.

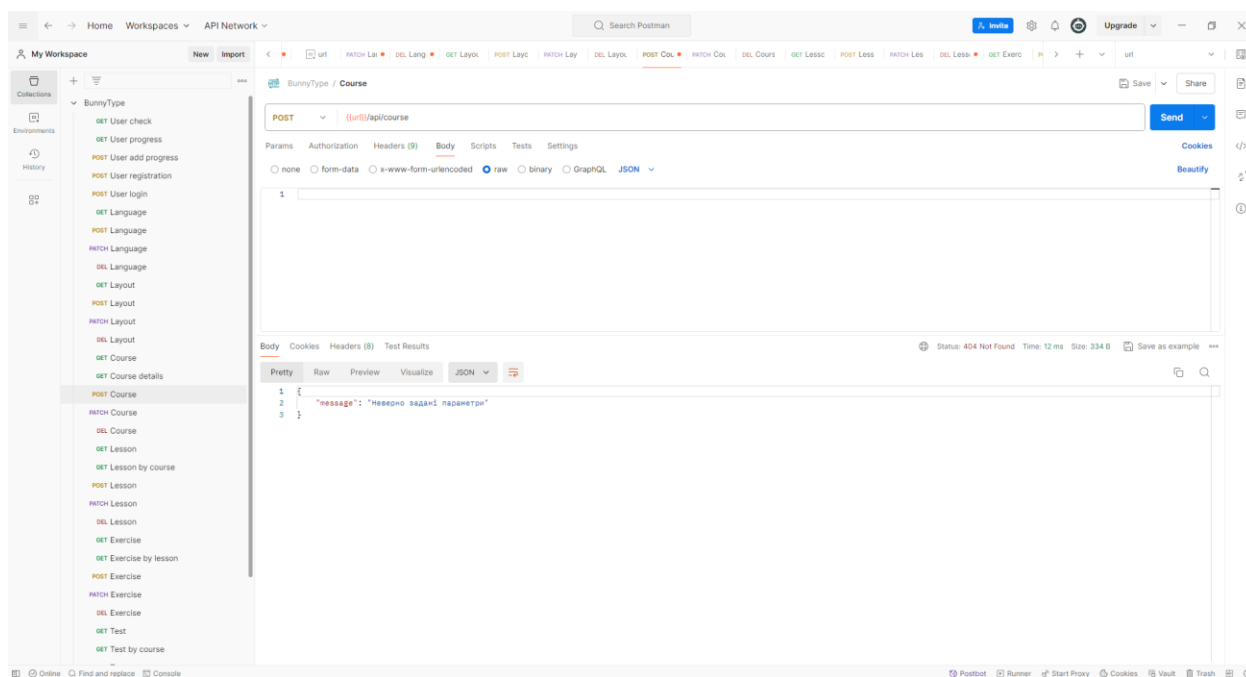


Рисунок 3.16 – Спроба створення курсу

Як і передбачалося, з'явилося повідомлення про неправильно задані параметри.

Перевіримо спробу створення завдання без доступу адміністратора.

На рисунку 3.17 зображено спробу створення завдання без доступу адміністратора.

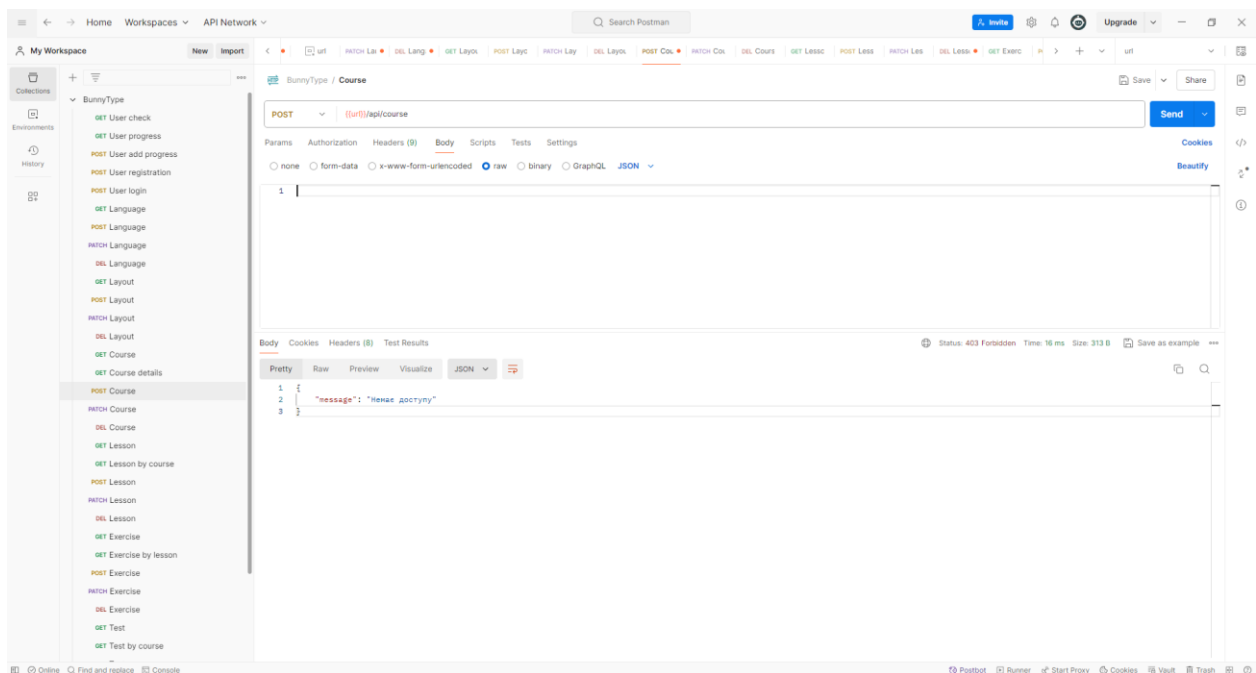


Рисунок 3.18 – Спроба створення курсу без прав адміністратора

Без прав адміністратора створити завдання не вдалося, як і очікувалося. Було отримано повідомлення про відсутність доступу.

Під час тестування сайт було перевірено, виявлені помилки виправлено. В результаті отримано додаток високої якості.

Висновки до розділу 3

У цьому розділі було створено діаграму розгортання та зібрано дані про порядок встановлення та налаштування параметрів системи. Також було показано структуру інтерфейсу та порядок взаємодії з сервісом. Детально описано функціональні можливості додатку в залежності від ролі користувача.

Було проведено тестування додатка за допомогою тестових сценаріїв і Postman. Під час тестування перевірено додаток та виправлено знайдені помилки. У підсумку отримано програму високої якості.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	54
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		

ВИСНОВКИ

В процесі виконання бакалаврської роботи був створений сайт для навчання сліпого методу друку, з наступними завданнями:

- 1) Був проведений аналіз популярних платформ для навчання сліпому методу друку. Кожна платформа має свої переваги та недоліки, але для нової системи ключовими були зручний веб-інтерфейс, модуль для збереження інформації про курси, уроки та завдання, а також модуль для відстеження прогресу.
- 2) Були визначені архітектурні компоненти для реалізації додатку: база даних PostgreSQL, мова програмування JavaScript, серверна частина на Node.js з використанням Express, клієнтська частина на React з використанням Sass. Середовище розробки включало в себе pgAdmin4 та WebStorm.
- 3) Проведено аналіз функціональних та нефункціональних вимог, створено діаграму варіантів використання та діаграму класів.
- 4) Було реалізовано базу даних з 11 таблицями та зручним збереженням даних.
- 5) Був реалізований інтерфейс для адміністратора та користувача. Адміністратор може створювати мови, розкладки клавіатури, курси, уроки, завдання та тести.

Результатом є програмний продукт готовий для використання як сайт, який спрямований на навчання методу сліпого друку. У майбутньому можливе його впровадження для широкого кола користувачів, що зацікавлені в опануванні цього методу або покращенні своїх навичок у цій області.

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		55

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Atom [Електронний ресурс] – Режим доступу до ресурсу: <https://flight-manual.atom.io/>
2. Документація CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
3. Документація Express для Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/>
4. Документація HTML [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
5. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
6. Документація Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/>
7. Документація pgAdmin4 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.pgadmin.org/docs/pgadmin4/latest/index.html>
8. Документація PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/>
9. Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://reactjs.org/>
10. Документація Recharts [Електронний ресурс] – Режим доступу до ресурсу: <https://recharts.org/en-US/>
11. Документація Sass [Електронний ресурс] – Режим доступу до ресурсу: <https://sass-lang.com/documentation>
12. Postman [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postman.com>
13. Використання drawio.com [Електронний ресурс] – Режим доступу до ресурсу: <https://www.drawio.com/blog/features>

ДОДАТКИ

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		57

Специфікація прецедентів

Таблиця А.1

Ім'я варіанту використання: Додавання завдання		Рівень важливості: Високий
Основний актор: Адміністратор		
Короткий опис:	Додавання адміністратором нового завдання на платформі.	
Передумови:	Наявність прав адміністратора для додавання завдань та активний стан платформи.	
Пост-умови:	Успішне додане завдання до списку доступних завдань.	
Відносини:		
Асоціація:	Додане завдання пов'язане з обраним курсом або темою навчання.	
Включення:	Процес включає в себе введення тексту завдання та визначення параметрів.	
Розширення:	Можливість редагування, видалення та додавання завдань у майбутньому.	
Узагальнення:	Функціонал додавання завдань адміністратором доступний лише для спеціальних ролей на платформі.	
Основний потік:	Вхід в режим адміністрування -> Введення курсу -> Вибір уроку -> Введення параметрів завдання -> Додавання завдання.	
Альтернативні/Виключні потоки:	Відміна створення завдання -> Повернення до панелі адміністратора або попереднього кроку.	

Таблиця А.2

Ім'я варіанту використання: Додавання власного завдання		Рівень важливості: Низький
Основний актор: Користувач		
Короткий опис:	Додавання користувачем власного завдання на платформі.	
Передумови:	Наявність можливості додавання завдань та активний стан користувача.	
Пост-умови:	Успішне додане завдання до списку доступних завдань.	
Відносини:		
Асоціація:	Додане завдання пов'язане з користувачем.	
Включення:	Процес включає в себе введення тексту та назви завдання.	
Розширення:	Можливість редагування, видалення та додавання завдань у майбутньому.	

Продовження Таблиці А.2

Узагальнення:	Функціонал додавання власних завдань доступний для всіх користувачів платформи.
Основний потік:	Введення назви та тексту завдання -> Додавання завдання.
Альтернативні/Виключні потоки:	

Таблиця А.3

Ім'я варіанту використання: Вибір курсу		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:	Вибір користувачем курсу для навчання на платформі.	
Передумови:	Наявність доступних курсів та активний стан користувача.	
Пост-умови:	Обраний курс успішно вибраний і доступний для навчання.	
Відносини:		
Асоціація:	Вибір курсу пов'язаний з вибором тематики та напрямку навчання.	
Включення:	Вибір курсу включає в себе перегляд доступних опцій та вибір конкретного курсу.	
Розширення:	Можливість зміни курсу під час навчання та перегляд інших доступних курсів.	
Узагальнення:	Процес вибору курсу є загальним для всіх користувачів платформи.	
Основний потік:	Перегляд доступних курсів -> Вибір курсу -> Підтвердження вибору.	
Альтернативні/Виключні потоки:		

Таблиця А.3

Ім'я варіанту використання: Проходження тесту		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:	Проходження користувачем тесту з обраного курсу на платформі.	
Передумови:	Наявність тесту для проходження та активний стан користувача.	
Пост-умови:	Успішне завершення тесту.	
Відносини:		
Асоціація:	Завдання пов'язане з курсами та рівнями участі.	
Включення:	Виконання завдання включає в себе введення тексту, роботу з клавіатурою та перевірку результатів.	

Продовження Таблиці А.3

Розширення:	Можливість вдосконалення результату, перегляд результатів та повторення завдання.
Узагальнення:	Процес проходження завдання є загальним для всіх користувачів платформи.
Основний потік:	Вибір завдання -> Виконання завдання -> Перевірка результату -> Показ результату.
Альтернативні/Виключні потоки:	Некоректний ввід тексту -> Відображення помилки -> Повторне введення тексту.

Таблиця А.3

Ім'я варіанту використання: Проходження завдання		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:	Проходження завдання користувачем на платформі.	
Передумови:	Наявність завдання для проходження та активний стан користувача.	
Пост-умови:	Успішне завершення завдання.	
Відносини:		
Асоціація:	Завдання пов'язане з курсами та рівнями участі.	
Включення:	Виконання завдання включає в себе введення тексту, роботу з клавіатурою та перевірку результатів.	
Розширення:	Можливість вдосконалення результату, перегляд результатів та повторення завдання.	
Узагальнення:	Процес проходження завдання є загальним для всіх користувачів платформи.	
Основний потік:	Вибір завдання -> Виконання завдання -> Перевірка результату -> Показ результату.	
Альтернативні/Виключні потоки:	Некоректний ввід тексту -> Відображення помилки -> Повторне введення тексту.	

ДОДАТОК Б

Вміст файлу «index.js»

```
require('dotenv').config()
const express = require('express')
const sequelize = require('./db')
const cors = require('cors')
const router = require('./routes/index')
const errorHandler = require('./middleware/ErrorHandlerMiddleware')

const PORT = process.env.PORT || 5000
```

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	60
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		

```

const app = express()
app.use(cors())
app.use(express.json())
app.use('/api', router)

app.use(errorHandler)

app.get('/', (req, res) => {
  res.status(200).json({message: "WORKING!"})
});

const start = async () => {
  try {
    await sequelize.authenticate()
    await sequelize.sync()
    app.listen(PORT, () => console.log(`Server started on port ${PORT}`))
  } catch (e) {
    console.log(e)
  }
}

start()

```

Вміст файлу «db.js»

```

const {Sequelize} = require('sequelize')

module.exports = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    dialect: 'postgres',
    host: process.env.DB_HOST,
    port: process.env.DB_PORT
  }
)

```

Вміст файлу «models.js»

```

const sequelize = require('./db')
const {DataTypes, Op} = require('sequelize')

const Users = sequelize.define('Users', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  username: { type: DataTypes.STRING, defaultValue: 'Bunny' },

```

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		61

```

    email: { type: DataTypes.STRING, unique: true },
    password: { type: DataTypes.STRING },
    role: { type: DataTypes.STRING, defaultValue: 'USER' },
    courseId: { type: DataTypes.INTEGER },
    lastExerciseId: { type: DataTypes.INTEGER },
    showKeyboard: { type: DataTypes.BOOLEAN, defaultValue: true },
  });

const UserExercises = sequelize.define('UserExercises', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  userId: { type: DataTypes.INTEGER },
  name: { type: DataTypes.STRING },
  text: { type: DataTypes.TEXT },
});

Users.hasMany(UserExercises, { foreignKey: 'userId' });
UserExercises.belongsTo(Users, { foreignKey: 'userId' });

const Progress = sequelize.define('Progress', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  userId: { type: DataTypes.INTEGER },
  courseId: { type: DataTypes.INTEGER },
  lessonId: { type: DataTypes.INTEGER },
  exerciseId: { type: DataTypes.INTEGER },
  star: { type: DataTypes.BOOLEAN },
  target: { type: DataTypes.BOOLEAN },
  lightning: { type: DataTypes.BOOLEAN },
});

Users.hasMany(Progress, { foreignKey: 'userId' });
Progress.belongsTo(Users, { foreignKey: 'userId' });

const Languages = sequelize.define('Languages', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true },
});

```

```
const Layouts = sequelize.define('Layouts', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true },
});
```

```
const Keyboards = sequelize.define('Keyboards', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  languageId: { type: DataTypes.INTEGER },
  layoutId: { type: DataTypes.INTEGER },
  keyboard: { type: DataTypes.JSON },
});
```

```
Languages.hasMany(Keyboards, { foreignKey: 'languageId' });
Keyboards.belongsTo(Languages, { as: 'language', foreignKey: 'languageId' });
```

```
Layouts.hasMany(Keyboards, { foreignKey: 'layoutId' });
Keyboards.belongsTo(Layouts, { as: 'layout', foreignKey: 'layoutId' });
```

```
const Courses = sequelize.define('Courses', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  orderId: { type: DataTypes.INTEGER },
  keyboardId: { type: DataTypes.INTEGER },
  name: { type: DataTypes.STRING },
}, {
  hooks: {
    async beforeCreate(course) {
      const maxOrderId = await Courses.max('orderId');
      course.orderId = maxOrderId ? maxOrderId + 1 : 1;
    },
    async beforeDestroy(course) {
      await Courses.update({ orderId: sequelize.literal('"orderId" - 1') }, { where: { orderId: { [Op.gt]: course.orderId } } });
      await Courses.destroy({ where: { id: course.id } });
    }
  }
});
```

```
Courses.belongsTo(Keyboards, { foreignKey: 'keyboardId', as: 'keyboard' });
Keyboards.hasMany(Courses, { foreignKey: 'keyboardId', as: 'courses' });
```

```
const Lessons = sequelize.define('Lessons', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  orderId: { type: DataTypes.INTEGER },
  courseId: { type: DataTypes.INTEGER },
  name: { type: DataTypes.STRING },
}, {
  hooks: {
    async beforeCreate(lesson) {
      const maxOrderId = await Lessons.max('orderId', { where: { courseId: lesson.courseId } });
      lesson.orderId = maxOrderId ? maxOrderId + 1 : 1;
    },
    async beforeDestroy(lesson) {
      await Lessons.update({ orderId: sequelize.literal('"orderId" - 1') }, {
        where: {
          courseId: lesson.courseId,
          orderId: { [Op.gt]: lesson.orderId }
        }
      });
    }
  }
});
```

```
Courses.hasMany(Lessons, { foreignKey: 'courseId' });
Lessons.belongsTo(Courses, { foreignKey: 'courseId' });
```

```
const Exercises = sequelize.define('Exercises', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  orderId: { type: DataTypes.INTEGER },
  lessonId: { type: DataTypes.INTEGER },
  name: { type: DataTypes.STRING },
  string: { type: DataTypes.STRING },
  errorLimit: { type: DataTypes.INTEGER },
  speedLimit: { type: DataTypes.INTEGER },
```

```

    active: { type: DataTypes.BOOLEAN },
  }, {
    hooks: {
      async beforeCreate(exercise) {
        const maxOrderId = await Exercises.max('orderId', { where: { lessonId: exercise.lessonId } });
        exercise.orderId = maxOrderId ? maxOrderId + 1 : 1;
      },
      async beforeDestroy(exercise) {
        await Exercises.update({ orderId: sequelize.literal('"orderId" - 1') }, {
          where: {
            lessonId: exercise.lessonId,
            orderId: { [Op.gt]: exercise.orderId }
          }
        });
      }
    }
  });

```

```

Lessons.hasMany(Exercises, { as: 'exercises', foreignKey: 'lessonId' });
Exercises.belongsTo(Lessons, { as: 'lesson', foreignKey: 'lessonId' });

```

```

const Tests = sequelize.define('Tests', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  courseId: { type: DataTypes.INTEGER },
  text: { type: DataTypes.TEXT },
});

```

```

const TestProgress = sequelize.define('TestProgress', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  userId: { type: DataTypes.INTEGER },
  courseId: { type: DataTypes.INTEGER },
  speed: { type: DataTypes.INTEGER },
  target: { type: DataTypes.FLOAT },
});

```

```
Courses.hasMany(Tests, { foreignKey: 'courseId' });
Tests.belongsTo(Courses, { foreignKey: 'courseId' });
```

```
module.exports = {
  Users,
  Progress,
  Languages,
  Layouts,
  Keyboards,
  Courses,
  Lessons,
  Exercises,
  Tests,
  TestProgress,
  UserExercises
}
```

ДОДАТОК В

Вміст файлу «routers.js»

```
import Admin from "./pages/Admin/Admin";
import {ADMIN_ROUTE, LOGIN_ROUTE, REGISTRATION_ROUTE} from "./utils/consts";
import Auth from "./pages/Auth/Auth";
import Exercise from "./pages/Exercise/Exercise";
import UserProfile from "./pages/UserProfile/UserProfile";
import App from "./components/App";
import Test from "./pages/Test/Test";
import CourseSelect from "./pages/CourseSelect/CourseSelect";
import Home from "./pages/Home/Home";
import UserExercise from "./pages/UserExercise/UserExercise";

export const authRouters = [
  {
    path: ADMIN_ROUTE,
    Component: Admin,
  },
  {
    path: '/typing',
```

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		66

```

        Component: Exercise,
    },
    {
        path: '/test',
        Component: Test,
    },
    {
        path: '/user',
        Component: UserProfile,
    },
    {
        path: '/user-exercise',
        Component: UserExercise,
    }
]

export const publicRouters = [
    {
        path: '/',
        Component: Home,
    },
    {
        path: LOGIN_ROUTE,
        Component: Auth,
    },
    {
        path: REGISTRATION_ROUTE,
        Component: Auth,
    },
    {
        path: '/course-select',
        Component: CourseSelect,
    },
]

```

Вміст файлу «App.jsx»

```
import './App.module.scss';
```

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		67

```

import {BrowserRouter} from "react-router-dom";
import AppRouter from "../AppRouter";
import {useRecoilState} from "recoil"
import Header from "../Header/Header";
import {keyboardState, userState} from "../store/atoms";
import {useEffect, useState} from "react";
import {check, login} from "../http/userAPI";
import Container from "../Container/Container";
import '../index.scss'
import Footer from "../Footer/Footer";
import styles from '../App.module.scss'

```

```

const App = () => {
  const [user, setUser] = useRecoilState(userState);
  const [loading, setLoading] = useState(true);

```

```

  useEffect(() => {
    check().then(data => {
      setUser(data)
    }).finally(() => setLoading(false))
  }, [])

```

```

  if (loading) {
    return <span>Loading ...</span>
  }

```

```

  return (
    <BrowserRouter>
      <Header/>
      <main className={styles.main}>
        <AppRouter />
      </main>
      <Footer/>
    </BrowserRouter>
  );
};

```

```
export default App;
```

Вміст файлу «AppRouter.js»

```
import React from 'react';
```

```
import {Navigate, Route, Routes} from "react-router-dom";
```

```
import {authRouters, publicRouters} from "../routers";
```

```
import {useRecoilState, useRecoilValue} from "recoil";
```

```
import {userState} from "../store/atoms";
```

```
const AppRouter = () => {
```

```
  const [user, setUser] = useRecoilState(userState);
```

```
  return (
```

```
    <Routes>
```

```
      {user && authRouters.map((( {path, Component}) =>
```

```
        <Route key={path} path={path} Component={Component}/>
```

```
      )}}
```

```
      {publicRouters.map((( {path, Component}) =>
```

```
        <Route key={path} path={path} Component={Component}/>
```

```
      )}}
```

```
      <Route path="*" element={<Navigate to="/404" replace />} />
```

```
    </Routes>
```

```
  );
```

```
};
```

```
export default AppRouter;
```

		Литвин В.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	
		Концидайло А.М.				
Змн.	Арк.	№ докум	Підпис	Дата		69