

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

факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

кафедра інформаційних систем та технологій
(повна назва кафедри)

Курсова робота

з дисципліни «Програмування-3»

на тему:

Виконав : студент __1__ курсу, групи ІА-33
(шифр групи)

Самойленко Анастасія Андріївна
(прізвище, ім'я, по батькові)

(підпис)

Науковий керівник асистент кафедри ІСТ Мягкий М. Ю.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Члени комісії

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Київ – 2020

ЗМІСТ

ВСТУП	3
1 ВИМОГИ ДО СИСТЕМИ	5
2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ.....	7
2.1 Діаграма прецедентів	7
2.2 Опис сценаріїв використання системи.....	9
3 АРХІТЕКТУРА СИСТЕМИ.....	25
4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ	27
4.1 Загальна структура проекту	27
4.2 Компоненти рівня доступу до даних.....	29
4.3 Компоненти рівня бізнес-логіки.....	32
4.4 Компоненти рівня інтерфейсу користувача	34
ВИСНОВКИ.....	36

ВСТУП

У епоху цифровізації та глобалізації, коли межі між країнами та культурами стають все менш помітними, важливість доступу до надійних та зручних інструментів для перекладу та вивчення мов зростає. Онлайн словники відіграють ключову роль у цьому процесі, надаючи користувачам можливість швидкого доступу до значень слів, їх вимови, етимології та перекладів. Розробка універсального онлайн словника, який би задовольняв потреби широкого кола користувачів, є актуальним завданням, яке вимагає глибокого аналізу та інноваційного підходу.

Метою даної роботи є створення системи онлайн-словника за допомогою JAVA бекенду та HTML фронтенду, яка надасть користувачам зручний та швидкий доступ до необхідної інформації про мови, слова та їх переклади. Система буде включати в себе дві основні функціональні частини: адміністративну та користувацьку.

Адміністративна частина системи буде відповідати за створення, редагування та видалення інформації про мови, слова та переклади. Адміністратор матиме можливість додавати нові мови, редагувати існуючі, а також видаляти неактуальні мови зі словника. Також адміністратор зможе додавати нові слова, редагувати вже існуючі та видаляти слова, які стали неактуальними. Крім того, адміністратор матиме можливість додавати нові переклади, редагувати вже існуючі та видаляти непотрібні переклади.

Користувацька частина системи надасть можливість користувачам вибирати мову, для якої вони бажають перекладу, а також знаходити переклад певного слова з обраної мови на іншу мову. Завдяки цьому користувачі зможуть швидко та зручно знайти переклад необхідного слова, що значно полегшить їхній процес навчання мови та комунікації з представниками інших країн.

Адміністраторська частина системи дозволить додавати нові мови, слова та переклади до системи, редагувати та видаляти їх. Це дозволить збільшувати кількість мов та словників у системі, що зробить її ще більш корисною та зручною для користувачів.

Окрім того, реалізація такої системи дає можливість зберігати та управляти великим обсягом інформації про різні мови та їх переклади. Це забезпечить більш точний та швидкий доступ до необхідної інформації, зменшить час та зусилля, потрібні для знаходження необхідного перекладу.

Така система є актуальною та корисною для різних груп користувачів, зокрема студентів, викладачів, перекладачів та всіх, хто цікавиться вивченням різних мов. Реалізація системи дозволить покращити якість навчання та сприятиме подальшому розвитку мовної освіти.

1 ВИМОГИ ДО СИСТЕМИ

2.1 Функціональні вимоги

Система повинна включати наступні функціональні вимоги:

1. Для адміністратора:

- додавання, редагування та видалення мов;
- додавання, редагування та видалення слів;
- додавання, редагування та видалення перекладів.

2. Для користувача:

- вибір мови для перекладу;
- перегляд словника;
- пошук слова в словнику;
- перегляд перекладу слова.

2.2 Нефункціональні вимоги

Система повинна включати наступні нефункціональні вимоги:

1. Безпека:

- система повинна мати механізми захисту від несанкціонованого доступу до даних;
- адміністратор повинен мати права доступу лише до відповідної функціональності системи.

2. Надійність:

- система повинна працювати безперервно 24 години на добу, 7 днів на тиждень;
- система повинна бути здатна дошкулюватися до збоїв та відновлюватися в короткий термін.

3. Швидкість та продуктивність:

- система повинна працювати швидко та ефективно, відповідаючи на запити користувачів у найкоротший термін.

4. Крос-платформність:

- система повинна бути доступна на будь-яких пристроях та операційних системах.

5. Інтуїтивний та зрозумілий інтерфейс:

- інтерфейс системи повинен бути легко зрозумілим та зручним для використання користувачами.

6. Масштабованість:

- система повинна бути здатна масштабуватися відповідно до зростання кількості користувачів та даних.

Отже, на основі вимог замовника до системи, можна визначити технічні вимоги до системи, які включають в себе вимоги до програмного забезпечення, до апаратного забезпечення та до взаємодії між ними. Також необхідно розробити архітектуру системи, забезпечити безпеку даних, розробити інтерфейс користувача, написати відповідний код програм та протестувати їх на відповідність вимогам замовника

2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Ця діаграма показує основні функціональні можливості системи, які доступні користувачам та адміністраторам [3]. Користувач може переглядати словник, використовуючи функції пошуку та перегляду перекладів, а також залишати коментарі. Адміністратор може додавати, видаляти та редагувати мови, слова та переклади. Усі користувачі повинні бути аутентифіковані, щоб мати доступ до системи.



Рисунок 2.1 – Діаграма прецедентів

2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.17.

Таблиця 2.1 – Сценарій використання «Реєстрація нового користувача»

Назва	Реєстрація нового користувача
ID	1
Опис	Користувач реєструється у системі
Актори	Користувач
Вигоди компанії	Можливість збільшення кількості користувачів, що підвищує популярність сервісу
Частота користування	Один раз
Тригери	Користувач відкриває сторінку реєстрації
Передумови	Користувач відкриває сторінку реєстрації
Постумови	Користувач успішно зареєстрований
Основний розвиток	<ol style="list-style-type: none"> 1. Користувач відкриває сторінку реєстрації 2. Користувач заповнює поля форми реєстрації 3. Користувач підтверджує реєстрацію 4. Система зберігає дані користувача
Альтернативні розвитку	<ol style="list-style-type: none"> 3а. Користувач невірно вводить дані у форму реєстрації 3а1. Система повідомляє користувача про помилку [4]
Виняткові ситуації	-

В таблиці 2.2 представлений сценарій використання «Авторизація користувача»

Таблиця 2.2 – Сценарій використання «Авторизація користувача»

Назва	Прослуховування обраних композицій
ID	2
Опис	Користувач вводить свій логін та пароль для входу в систему
Актори	Користувач
Вигоди компанії	Забезпечення безпеки та обмеження доступу до конфіденційної інформації
Частота користування	Кожного разу, коли користувач хоче увійти в систему
Тригери	Користувач натискає на кнопку «Увійти в систему»
Передумови	Користувач має створений обліковий запис
Постумови	Користувач потрапляє на особисту сторінку
Основний розвиток	Користувач вводить свій логін та пароль, натискає на кнопку «Увійти в систему»
Альтернативні розвитку	-
Виняткові ситуації	Користувач вводить неправильний логін або пароль, система повідомляє про помилку

В таблиці 2.3 представлений сценарій використання «Перегляд списку доступних мов»

Таблиця 2.3 – Сценарій використання «Перегляд списку доступних мов»

Назва	Перегляд списку доступних мов [5]
ID	3
Опис	Користувач може переглянути список мов, на які доступні переклади
Актори	Користувач
Вигоди компанії	Забезпечення можливості користувачам знаходити переклади на різні мови
Частота користування	Незалежно
Тригери	Користувач натискає на кнопку «Доступні мови»
Передумови	Користувач має створений обліковий запис
Постумови	Користувач бачить список доступних мов
Основний розвиток	Користувач натискає на кнопку «Доступні мови», система відображає список доступних мов
Альтернативні розвитку	-
Виняткові ситуації	-

В таблиці 2.4 представлений сценарій використання «Додавання нової мови»

Таблиця 2.4 – Сценарій використання «Додавання нової мови»

Назва	Пошук композицій по наданому користувачем аудіо із мікрофону або файлу [5]
ID	4
Опис	Адміністратор додає нову мову до списку доступних
Актори	Адміністратор
Вигоди компанії	Розширення можливостей сервісу
Частота користування	Іноді
Тригери	Адміністратор натискає кнопку "Додати мову"
Передумови	Адміністратор авторизувався в системі
Постумови	Нова мова додається до списку доступних
Основний розвиток	<ol style="list-style-type: none"> 1 Адміністратор вибирає опцію "Додати мову" 2. Адміністратор вводить назву мови 3. Адміністратор натискає кнопку "Додати мову" "
Альтернативні розвитку	Адміністратор може скасувати додавання мови
Виняткові ситуації	Інтерфейс збоїть або не працює коректно

В таблиці 2.5 представлений сценарій використання «Редагування існуючої мови адміністратором»

Таблиця 2.5 – Сценарій використання «Редагування існуючої мови адміністратором»

Назва	Редагування існуючої мови адміністратором [6]
ID	5
Опис	Адміністратор редагує наявний у списку мов
Актори	Адміністратор
Вигоди компанії	Збереження точності і релевантності інформації для користувачів, які використовують різні мови
Частота користування	Рідко
Тригери	Адміністратор вибирає наявну мову для редагування
Передумови	Адміністратор має право доступу до редагування мов
Постумови	Мова успішно змінена та збережена в системі
Основний розвиток	<ol style="list-style-type: none"> 1. Адміністратор входить до панелі адміністратора. 2. Адміністратор відкриває вкладку "Мови". 3. Адміністратор вибирає потрібну мову зі списку доступних мов. 4. Адміністратор вносить необхідні зміни. 5. Адміністратор зберігає зміни.
Альтернативні розвитку	-
Виняткові ситуації	-

В таблиці 2.6 представлений сценарій використання «Видалення мови»

Таблиця 2.6 – Сценарій використання «Видалення мови» [7]

Назва	Видалення мови
ID	6
Опис	Адміністратор видаляє існуючу мову зі списку мов сервісу
Актори	Адміністратор
Вигоди	Адміністратор може зменшити кількість мов в системі
Частота	Рідко
Тригери	Адміністратор обирає опцію видалення мови
Передумови	Адміністратор має права доступу до функції
Постумови	Мова більше не доступна в системі
Основний розвиток	<ol style="list-style-type: none"> 1. Адміністратор входить у систему 2. Адміністратор обирає опцію видалення мови 3. Система підтверджує намір адміністратора видалити мову 4. Система видаляє мову зі списку доступних мов
Альтернативні розвитку	-
Виняткові ситуації	1. Мова є офіційною мовою сервісу і не може бути видалена

В таблиці 2.7 представлений сценарій використання «Пошук по слову»

Таблиця 2.7 – Сценарій використання «Пошук по слову»

Назва	Пошук по слову
ID	7
Опис	Користувач переглядає список доступних слів
Актори	Користувач, Адміністратор
Вигоди компанії	Користувачі можуть швидко знайти необхідне слово, збільшується користування сервісом
Частота користування	Постійно
Тригери	Користувач відкриває сторінку зі списком слів
Передумови	Список доступних слів має бути завантажений
Постумови	Користувач бачить список доступних слів
Основний розвиток	Користувач відкриває сторінку зі списком слів
Альтернативні розвитку	-
Виняткові ситуації	-

В таблиці 2.8 представлений сценарій використання «Перегляд інформації про слово»

Таблиця 2.8 – Сценарій використання «Перегляд інформації про слово»

Назва	Перегляд інформації про слово [8]
ID	8
Опис	Користувач переглядає інформацію про конкретне слово
Актори	Користувач, Адміністратор
Вигоди компанії	Користувачі можуть детальніше ознайомитися зі словом, збільшується користування сервісом
Частота користування	Постійно
Тригери	Користувач клікає на слово в списку
Передумови	Список доступних слів має бути завантажений
Постумови	Користувач бачить детальну інформацію про слово
Основний розвиток	Користувач клікає на слово в списку
Альтернативні розвитку	-
Виняткові ситуації	Слово не існує в базі даних

В таблиці 2.9 представлений сценарій використання «Додавання нового слова адміністратором»

Таблиця 2.9 – Сценарій використання «Додавання нового слова адміністратором»

Назва	Додавання нового слова адміністратором
ID	9
Опис	Адміністратор додає нове слово до бази даних
Актори	Адміністратор
Вигоди компанії	Розширення словника сервісу, покращення користувацького досвіду
Частота користування	Рідко
Тригери	Адміністратор вибирає опцію додавання нового слова
Передумови	Адміністратор має доступ до системи управління словником
Постумови	Нове слово додається до бази даних
Основний розвиток	<ol style="list-style-type: none"> 1. Адміністратор вибирає опцію додавання нового слова 2. Система відкриває форму додавання нового слова. 3. Адміністратор вводить інформацію про нове слово (слово, опис, синоніми тощо). 4. Адміністратор натискає на кнопку «Додати». 5. Система перевіряє введені дані та додає нове слово до бази даних. [9]
Альтернативні розвитку	-

Виняткові ситуації	1. Адміністратор вводить невірні дані. 2. Система відображає повідомлення про помилку. 3. Адміністратор виправляє дані та спробує знову додати слово.
--------------------	---

В таблиці 2.10 представлений сценарій використання «Редагування існуючого слова адміністратором»

Таблиця 2.10 – Сценарій використання «Редагування існуючого слова адміністратором»

Назва	Редагування існуючого слова адміністратором
ID	10
Опис	Адміністратор редагує інформацію про існуюче слово в базі даних
Актори	Адміністратор
Вигоди компанії	Забезпечується актуальність і правильність інформації про слова в базі даних
Частота користування	Залежить від необхідності редагування
Тригери	Адміністратор обрав слово для редагування
Передумови	Слово вже існує в базі даних
Постумови	Інформація про слово змінена
Основний розвиток	1. Адміністратор входить у систему та переходить до списку доступних слів. 2. Адміністратор обирає слово для редагування.

	<p>3. Адміністратор змінює необхідну інформацію.</p> <p>4. Адміністратор зберігає зміни.</p>
Альтернативні розвитки	–
Виняткові ситуації	<p>1. Адміністратор не може відредагувати слово, якщо він не має відповідних прав доступу до бази даних.</p> <p>2. Адміністратор не може відредагувати слово, якщо воно вже було видалено з бази даних.</p>

В таблиці 2.11 представлений сценарій використання «Видалення слова адміністратором»

Таблиця 2.11 – Сценарій використання «Видалення слова адміністратором»

Назва	Видалення слова адміністратором
ID	11
Опис	Адміністратор видаляє слово з бази даних
Актори	Адміністратор
Вигоди компанії	Зменшення обсягу бази даних і поліпшення швидкості роботи системи
Частота користування	Рідко
Тригери	Адміністратор вибирає слово для видалення
Передумови	Адміністратор має доступ до бази даних

Постумови	Слово більше не міститься в базі даних
Основний розвиток	Адміністратор вибирає слово, натискає на кнопку видалення і підтверджує видалення
Альтернативні розвитки	Адміністратор може скасувати видалення, натиснувши на кнопку "Скасувати"
Виняткові ситуації	Якщо слово не існує в базі даних, адміністратор отримає повідомлення про помилку

В таблиці 2.12 представлений сценарій використання «Переклад слова»

Таблиця 2.12 – Сценарій використання «Переклад слова»

Назва	Розмістити аудіорекламу
ID	12
Опис	Користувач може переглянути переклад слова на обраний мовний рівень
Актори	Користувач
Вигоди компанії	Збільшення користувацької бази завдяки можливості використання сервісу користувачами з різних мовних середовищ
Частота користування	Рідко
Тригери	Користувач натискає кнопку "Переклад" для відображення перекладу слова

Передумови	Користувач увійшов в систему та має доступ до словника
Постумови	Користувач бачить переклад слова на обраний мовний рівень
Основний розвиток	Користувач вводить слово в поле пошуку, натискає кнопку "Переклад", вибирає мовний рівень зі списку, бачить переклад слова
Альтернативні розвитку	Відображення перекладу слова без вибору мовного рівня
Виняткові ситуації	Система не має перекладу слова на обраний мовний рівень

В таблиці 2.13 представлений сценарій використання «Додавання нового перекладу адміністратором»

Таблиця 2.13 – Сценарій використання «Додавання нового перекладу адміністратором»

Назва	Додавання нового перекладу адміністратором
ID	13
Опис	Адміністратор може додати новий переклад для існуючого слова
Актори	Адміністратор
Вигоди компанії	Розширення мовної бази сервісу

Частота користування	Рідко
Тригери	Адміністратор обирає опцію додавання нового перекладу
Передумови	Адміністратор увійшов в систему та має доступ до словника
Постумови	Новий переклад слова додано до бази даних
Основний розвиток	Рекламодавець із особистого кабінету вибирає аудіо, який хоче видалити, натискає відповідну кнопку
Альтернативні розвитки	—
Виняткові ситуації	—

В таблиці 2.14 представлений сценарій використання «Видалення слова»

Таблиця 2.14 – Сценарій використання

Назва	Видалення слова
ID	14
Опис	Адміністратор видаляє переклад слова зі списку доступних перекладів.
Актори	Адміністратор
Вигоди компанії	Збереження точності і якості перекладів.

Частота користування	Іноді
Тригери	Адміністратор вирішує видалити переклад слова.
Передумови	Переклад слова вже існує в системі.
Постумови	Переклад слова більше не доступний для користувачів.
Основний розвиток	1. Адміністратор входить у систему. 2. Адміністратор відкриває список доступних перекладів. 3. Адміністратор вибирає переклад, який потрібно видалити. 4. Адміністратор підтверджує видалення перекладу.
Альтернативні розвитку	-
Виняткові ситуації	-

В таблиці 2.15 представлений сценарій використання «Пошук за параметрами»

Таблиця 2.15 – Сценарій використання «Пошук за параметрами»

Назва	Пошук за параметрами
ID	15
Опис	Користувач шукає слова, які відповідають певним параметрам
Актори	Користувач

Вигоди компанії	Збільшення задоволення користувачів від використання сервісу
Частота користування	Постійно
Тригери	Користувач вводить параметри пошуку у полі для пошуку
Передумови	Поле пошуку доступне у будь-якому вікні
Постумови	Користувач потрапляє на вікно з результатами пошуку
Основний розвиток	Користувач вводить параметри пошуку у полі для пошуку, натискає на кнопку пошуку чи Enter
Альтернативні розвитку	-
Виняткові ситуації	-

3 АРХІТЕКТУРА СИСТЕМИ

Архітектура системи, яку було згадано вище, використовує різні технології для забезпечення роботи веб-сервісу. На початку стоїть база даних SQLite, яка використовується для збереження даних, пов'язаних з користувачами, перекладами та іншими сутностями в системі. Далі, використовуючи JSP (JavaServer Pages), система генерує веб-сторінки для взаємодії з користувачем.

На серверному боці Java використовується для розробки бізнес-логіки, що включає в себе обробку запитів користувачів, роботу з базою даних та логіку аутентифікації та авторизації. Для дизайну веб-сторінок використовується мова розмітки HTML.

Ця архітектура системи дозволяє забезпечити швидку та надійну роботу веб-сервісу, збереження даних у базі даних та надання користувачам зручного інтерфейсу для взаємодії з системою.

Для створення діаграми общей архітектуры системи було використано PlantUML. Діаграма показує компоненти системи та їх взаємозв'язки, включаючи базу даних SQLite, бізнес-логіку на Java, генерацію веб-сторінок з використанням JSP та дизайн веб-сторінок з використанням HTML. Загальна архітектура системи наведена на рис. 3.1



Рисунок 3.1 – Загальна архітектура системи

Система складається з наступних елементів:

- Клієнтське застосування: включає в себе компоненти HTML та JavaScript, які забезпечують користувачам інтерфейс взаємодії з системою;
- Серверне застосування: включає в себе компоненти JSP та Java, які забезпечують бізнес-логіку системи та взаємодію з базою даних SQLite;
- База даних SQLite: використовується для зберігання даних про слова та їх переклади;

Клієнтське застосування надає можливість користувачам шукати та переглядати слова та їх переклади, додавати нові переклади, а також редагувати та видаляти існуючі переклади. Серверне застосування забезпечує перевірку даних, обробку запитів користувачів, а також зберігання даних у базі даних.

Взаємодія між компонентами системи здійснюється за допомогою протоколу HTTP, а база даних SQLite використовується для зберігання даних. Для реалізації функцій пошуку та фільтрації даних використовуються SQL запити до бази даних.

4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

4.1 Загальна структура проекту

Проект містить наступні складові:

1. Клієнтське застосування

- HTML - мова розмітки гіпертексту для створення інтерфейсу користувача;
- JavaScript - скриптова мова програмування для динамічного змінення вмісту HTML-сторінки без перезавантаження сторінки;

2. Серверне застосування

- JSP - технологія, що дозволяє створювати динамічні веб-сторінки, що включають в себе як HTML-код, так і Java-код;
- Java - об'єктно-орієнтована мова програмування, яка використовується для розробки серверної частини застосування;
- SQLite - реляційна база даних, яка використовується для збереження даних про мови, слова та їх переклади;

Ці складові взаємодіють між собою за допомогою HTTP-запитів. Клієнтське застосування запитує сервер за допомогою HTTP-запитів, а сервер відповідає на запити, оброблюючи їх і відправляючи відповіді, що містять відповідні дані. База даних SQLite використовується для збереження інформації про мови, слова та їх переклади.

Загальна структура проекту представлена на рис.4.1

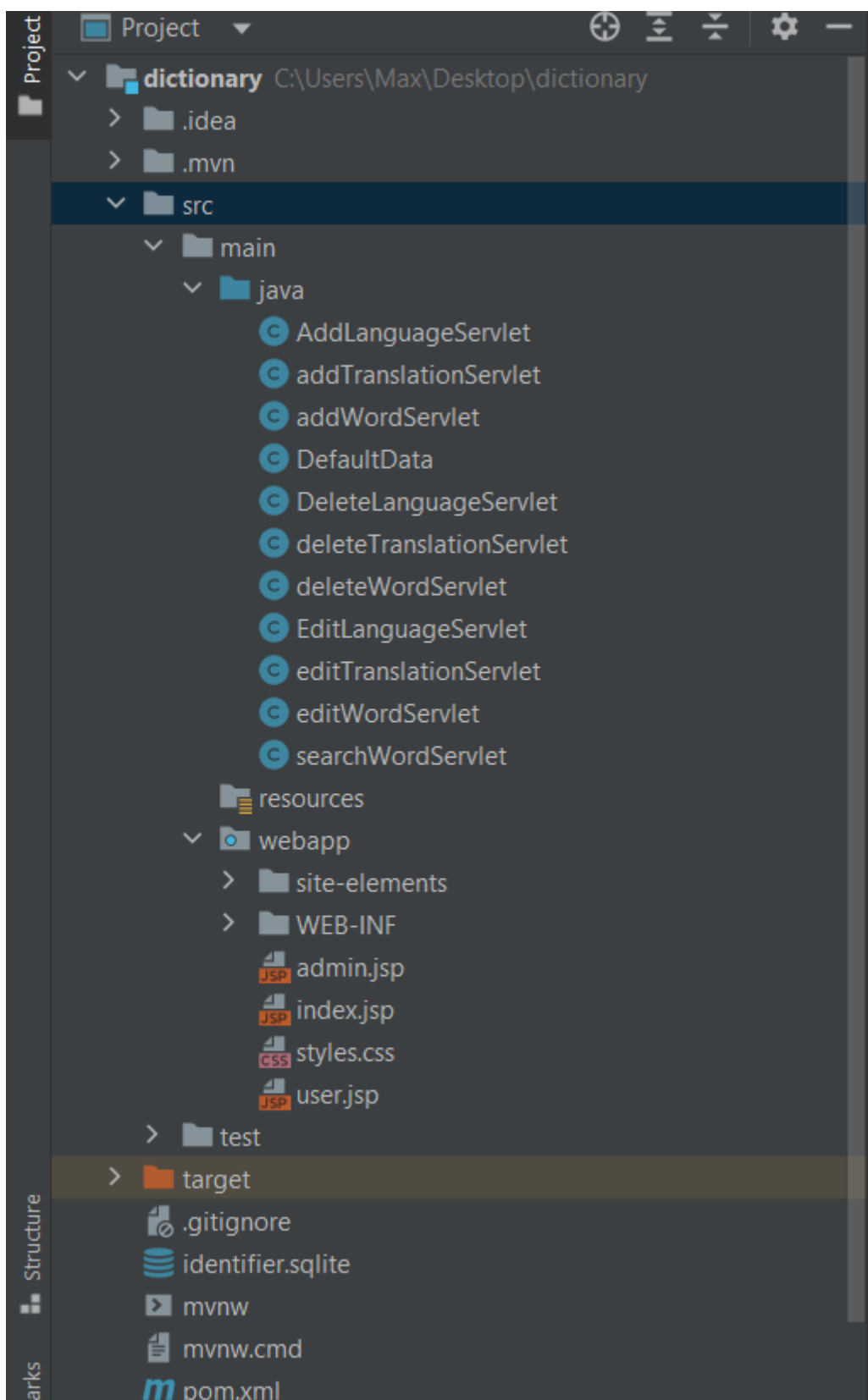


Рисунок 4.1 – Загальна структура проекту

Проект складається з веб-ресурсів, бібліотек, та вихідного коду, який в свою чергу можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти.

4.2 Компоненти рівня доступу до даних

Компоненти рівня доступу до даних в даному проєкті включають сервлети для додавання, редагування та видалення даних з бази даних. Нижче наведені короткі описи кожного сервлету:

1. `AddLanguageServlet` - сервлет для додавання нової мови в базу даних. Отримує дані з форми на стороні клієнта та вставляє їх в таблицю мов.
2. `AddTranslationServlet` - сервлет для додавання нового перекладу в базу даних. Отримує дані з форми на стороні клієнта та вставляє їх в таблицю перекладів.
3. `AddWordServlet` - сервлет для додавання нового слова в базу даних. Отримує дані з форми на стороні клієнта та вставляє їх в таблицю слів.
4. `DeleteLanguageServlet` - сервлет для видалення мови з бази даних. Отримує ID мови з форми на стороні клієнта та видаляє відповідний рядок з таблиці мов.
5. `DeleteWordServlet` - сервлет для видалення слова з бази даних. Отримує ID слова з форми на стороні клієнта та видаляє відповідний рядок з таблиці слів.
6. `DeleteTranslationServlet` - сервлет для видалення перекладу з бази даних. Отримує ID перекладу з форми на стороні клієнта та видаляє відповідний рядок з таблиці перекладів.
7. `EditLanguageServlet` - сервлет для редагування мови в базі даних. Отримує ID та нову назву мови з форми на стороні клієнта та оновлює відповідний рядок в таблиці мов.
8. `EditWordServlet` - сервлет для редагування слова в базі даних. Отримує ID та нове значення слова з форми на стороні клієнта та оновлює відповідний рядок в таблиці слів.
9. `EditTranslationServlet` - сервлет для редагування перекладу в базі даних. Отримує ID та нове значення перекладу з форми на стороні клієнта.

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.2

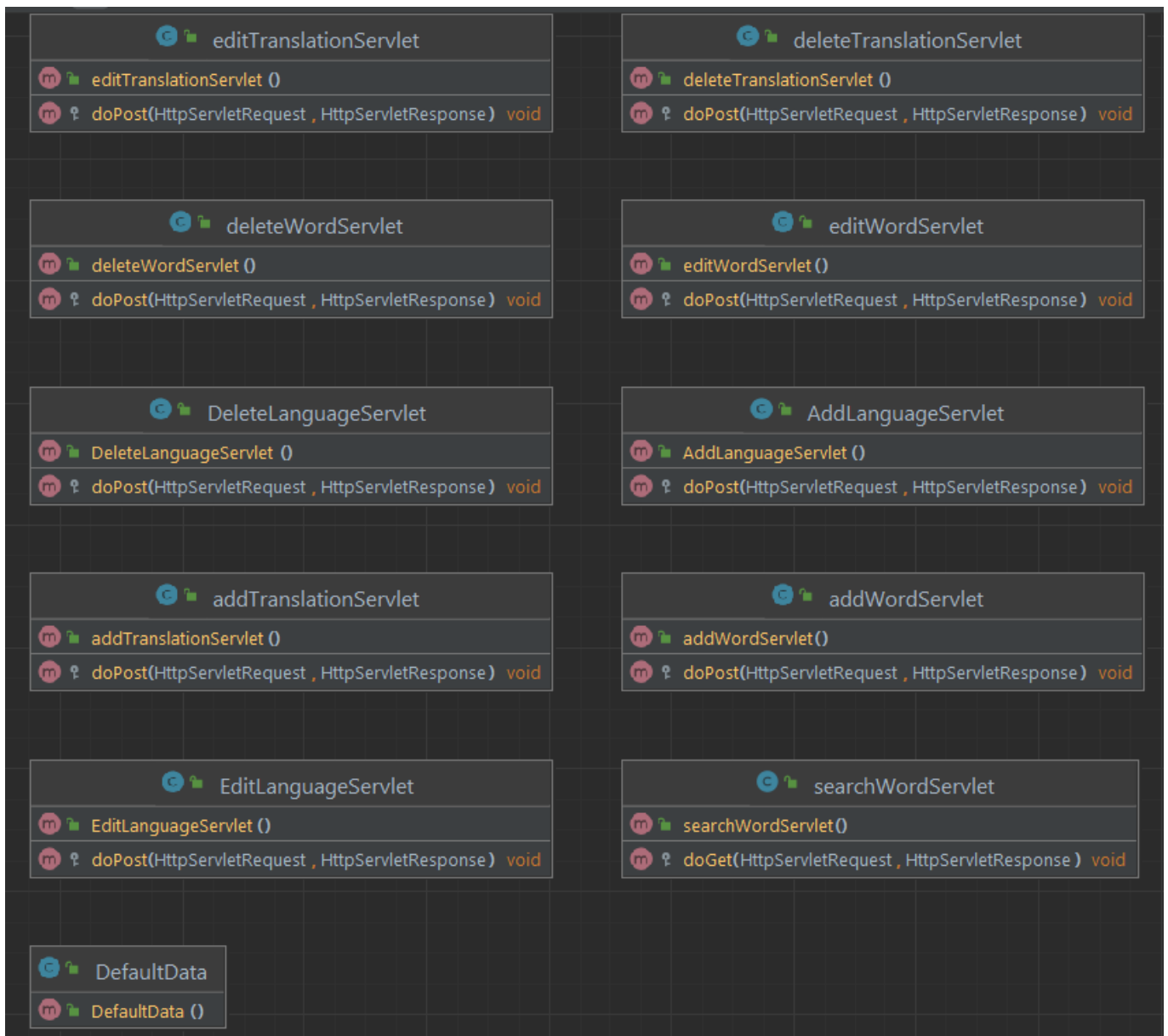


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

1. Сутність "Мова" (Language)

- Поля:
 - `id` - унікальний ідентифікатор мови
 - `name` - назва мови
- Зв'язки:
 - Один-до-багатьох з сутністю "Слово" (Word)

2. Сутність "Слово" (Word)

- Поля:

- id - унікальний ідентифікатор слова
- word - слово
- Зв'язки:
 - Багато-до-одного з сутністю "Мова" (Language)
 - Один-до-багатьох з сутністю "Переклад" (Translation)

3. Сутність "Переклад" (Translation)

- Поля:
 - id - унікальний ідентифікатор перекладу
 - translation - переклад слова на вибрану мову
- Зв'язки:
 - Багато-до-одного з сутністю "Слово" (Word)

4. Сутність "Користувач" (User)

- Поля:
 - id - унікальний ідентифікатор користувача
 - username - логін користувача
 - password - пароль користувача
 - role - роль користувача (admin або user)

Сутності взаємодіють між собою наступним чином:

- Кожен запис сутності "Мова" містить багато записів сутності "Слово".
- Кожен запис сутності "Слово" містить багато записів сутності "Переклад".
- Кожен запис сутності "Переклад" посилається на запис сутності "Слово".
- Користувачі мають можливість авторизуватись у системі з відповідною роллю.

4.3 Компоненти рівня бізнес-логіки

Компоненти рівня бізнес-логіки включають сервіси та моделі, які виконують операції з даними та забезпечують виконання бізнес-логіки застосунку.

Сервіси забезпечують взаємодію з компонентами рівня доступу до даних та виконання операцій з ними, таких як створення, оновлення та видалення даних. Також сервіси забезпечують валідацію даних та інші бізнес-правила, які повинні бути виконані перед збереженням даних.

Моделі представляють сутності даних та зв'язки між ними. Кожна модель містить логіку, яка забезпечує коректне взаємодію з компонентами рівня доступу до даних. Наприклад, модель мови містить методи для додавання, видалення та оновлення мови, а також методи для отримання списку всіх слів та перекладів цієї мови.

Компоненти рівня бізнес-логіки забезпечують гнучкість та розширюваність застосунку, оскільки вони можуть бути використані в різних інтерфейсах та компонентах. Наприклад, моделі та сервіси можуть бути використані як в web-інтерфейсі, так і в мобільному додатку або API.

У нашій системі цей рівень складається з наступних компонентів:

1. `LanguageService` - відповідає за операції з мовами.

- `addLanguage()` - додає нову мову до бази даних.
- `editLanguage()` - змінює існуючу мову в базі даних.
- `deleteLanguage()` - видаляє мову з бази даних.

2. `WordService` - відповідає за операції зі словами.

- `addWord()` - додає нове слово до бази даних.
- `editWord()` - змінює існуюче слово в базі даних.
- `deleteWord()` - видаляє слово з бази даних.

3. TranslationService - відповідає за операції з перекладами.

- addTranslation() - додає новий переклад до бази даних.
- editTranslation() - змінює існуючий переклад в базі даних.
- deleteTranslation() - видаляє переклад з бази даних.

Кожен з цих сервісів має відповідні методи для взаємодії з базою даних. Наприклад, метод addLanguage() використовує об'єкт LanguageDao для додавання нової мови до бази даних.

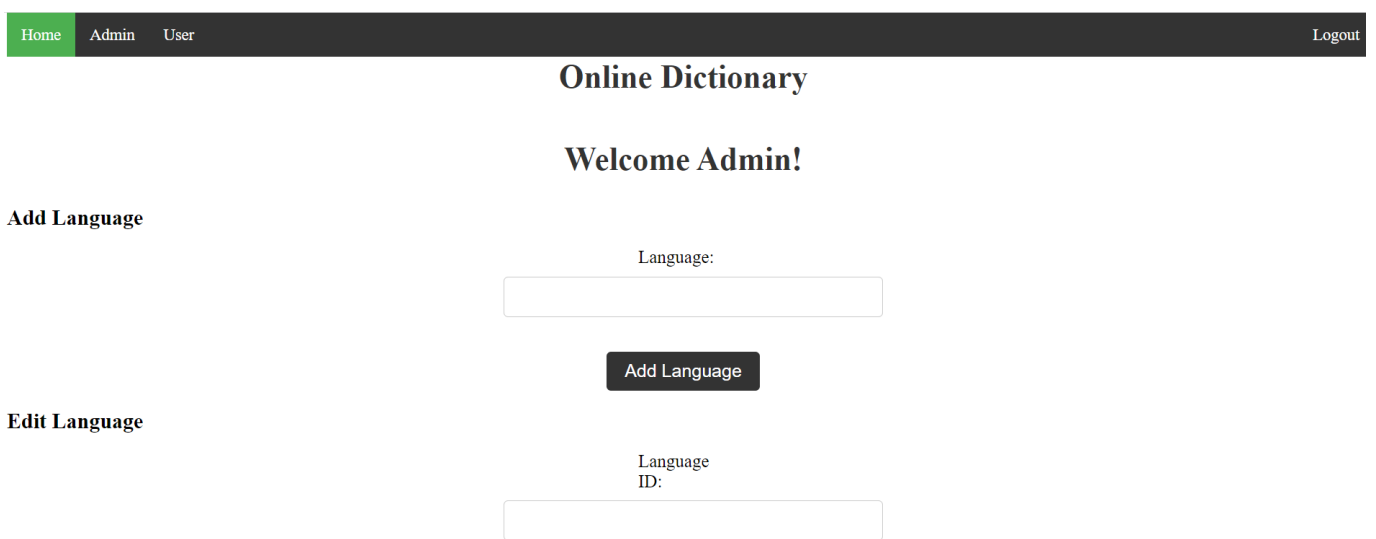
Для забезпечення правильної роботи системи, рівень бізнес-логіки виконує такі функції:

1. Перевірка коректності введених даних перед їх збереженням у базу даних. Наприклад, перевірка на унікальність ідентифікаторів або на наявність обов'язкових полів.
2. Виконання бізнес-правил, пов'язаних зі зміною даних. Наприклад, забезпечення того, що переклади, пов'язані з конкретним словом, будуть видалені при видаленні цього слова.
3. Обробка запитів на отримання даних з бази. Наприклад, отримання списку всіх мов або всіх перекладів.

4.4 Компоненти рівня інтерфейсу користувача

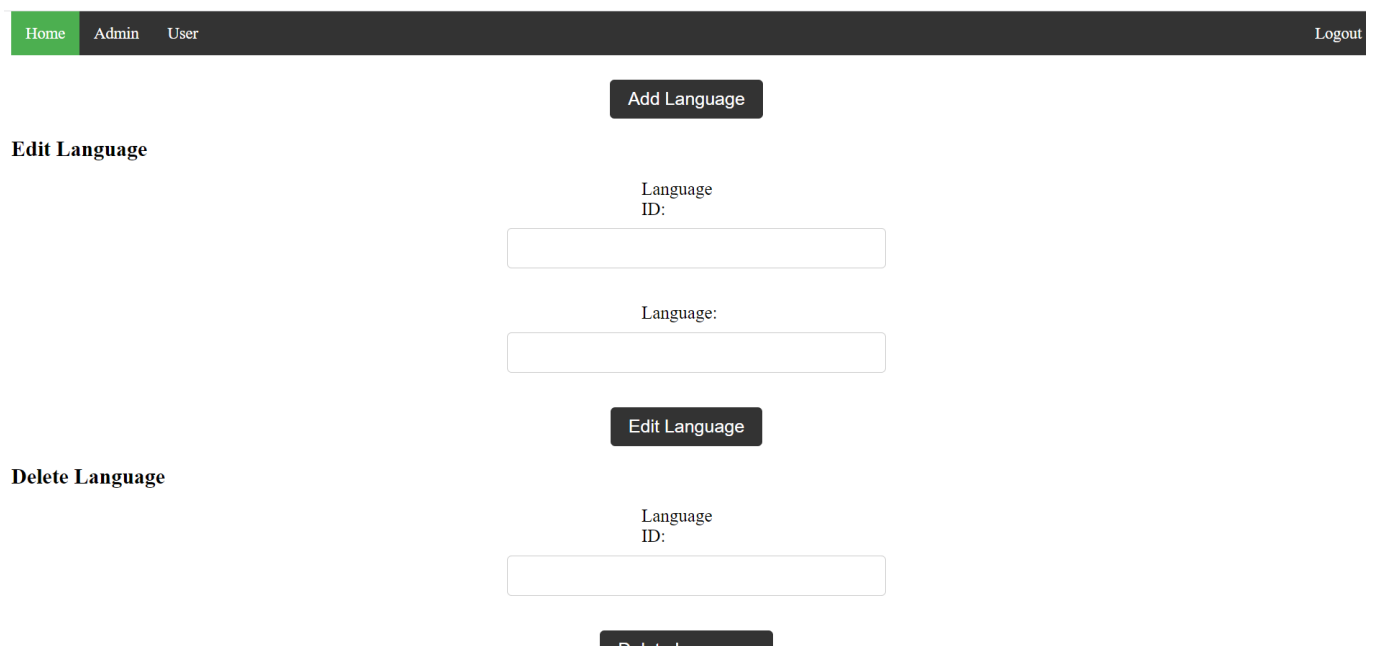
Компоненти рівня інтерфейсу користувача в нашому проєкті складаються з JSP сторінок та CSS файлів для стилізації веб-сторінок. Головна сторінка рівня інтерфейсу користувача - `index.jsp`, яка відображає форму авторизації. У разі успішної авторизації, користувач залежно від ролі буде перенаправлений на сторінку адміністратора або звичайного користувача.

На сторінці адміністратора (`admin.jsp`) відображені форми для додавання, редагування та видалення мов, слів та перекладів.



The screenshot shows the 'Online Dictionary' admin interface. At the top is a navigation bar with 'Home', 'Admin', and 'User' links, and a 'Logout' link on the right. Below the navigation bar, the title 'Online Dictionary' is centered, followed by a welcome message 'Welcome Admin!'. On the left side, there is a section titled 'Add Language'. To the right of this section is a form with a label 'Language:' above a text input field. Below the input field is a dark button labeled 'Add Language'.

Рис.4.4.1 – Сторінка адміністратора



This screenshot shows the same admin interface as the previous one, but with the 'Edit Language' and 'Delete Language' sections visible. The 'Add Language' button is still at the top. Below it, the 'Edit Language' section has a label 'Language ID:' above a text input field. Below this input field is another label 'Language:' above a second text input field. Below the second input field is a dark button labeled 'Edit Language'. At the bottom, the 'Delete Language' section has a label 'Language ID:' above a text input field. Below this input field is a dark button labeled 'Delete Language'.

Рис.4.4.2 – Функції адміністратора

На сторінці користувача (user.jsp) реалізована форма для пошуку перекладів слова на вибраній мові. Результат пошуку відображається на тій же сторінці.

The screenshot shows the 'Online Dictionary' user interface. At the top is a dark navigation bar with 'Home' (highlighted in green), 'Admin', and 'User' links on the left, and a 'Logout' link on the right. Below the navigation bar, the title 'Online Dictionary' is centered. On the left, there is a 'Select Language' label. In the center, there is a 'Language:' label with a dropdown menu showing '--Select--'. Below this is a 'Word:' label and an empty text input field. A dark 'Search' button is positioned below the input field. At the bottom of the page, a light gray footer contains the text '© 2023 Online Dictionary'.

Рис.4.4.3 – Сторінка користувача

This screenshot shows the 'Online Dictionary' user interface after a search. The navigation bar and title are the same as in the previous screenshot. The 'Word:' input field now contains the text 'test'. The 'Search' button is still present. Below the search form, there is a 'Search Result' section. It displays 'Word: test' and 'Translations:' followed by a bulleted list containing 'test' and 'test1'. The footer at the bottom remains '© 2023 Online Dictionary'.

Рис.4.4.4 – Переклад слова

Для стилізації веб-сторінок використовуються CSS файли, які знаходяться в окремій папці. Такий підхід дозволяє легко змінювати вигляд веб-сторінок без необхідності вносити зміни в HTML код.

Також у проєкті використовується файл header.jsp, який містить заголовок сторінки з основними посиланнями на рівні інтерфейсу користувача, і файл footer.jsp, який містить підвал сторінки. Ці файли включаються на кожній сторінці за допомогою тегу `jsp:include`. Це дозволяє уникнути дублювання коду та зручно змінювати елементи, які повторюються на кожній сторінці.

ВИСНОВКИ

Проект включає в себе три рівні архітектури: рівень доступу до даних, рівень бізнес-логіки та рівень інтерфейсу користувача.

Рівень доступу до даних реалізований за допомогою Java Servlets, які взаємодіють з базою даних за допомогою JDBC API. В проекті використовуються такі компоненти рівня доступу до даних, як "AddLanguageServlet", "EditLanguageServlet", "DeleteLanguageServlet", "AddWordServlet", "EditWordServlet", "DeleteWordServlet", "AddTranslationServlet", "EditTranslationServlet" та "DeleteTranslationServlet".

Рівень бізнес-логіки включає в себе компоненти для опрацювання даних, які надходять від рівня доступу до даних, та обробки запитів користувачів. Компоненти рівня бізнес-логіки відповідають за зберігання даних відповідно до бізнес-логіки додатку, а також за їх валідацію. В проекті використовуються такі компоненти рівня бізнес-логіки, як "LanguageService", "WordService" та "TranslationService".

Рівень інтерфейсу користувача відповідає за відображення даних користувачеві та обробку його запитів. В проекті реалізовано інтерфейс користувача за допомогою HTML, CSS та JavaServer Pages (JSP). Компоненти рівня інтерфейсу користувача включають в себе "login.jsp" для входу користувача в систему, "admin.jsp" для адміністративного доступу до даних, "user.jsp" для звичайного користувача та "search.jsp" для пошуку перекладів слів.

Загальна структура проекту дозволяє легко додавати нові мови, слова та переклади за допомогою відповідних сервлетів та JSP сторінок. Крім того, різні рівні компонентів добре розділені, що сприяє кращій модульності та розширюваності проекту.

Компоненти рівня доступу до даних дозволяють взаємодіяти з базою даних, зберігаючи та отримуючи інформацію про мови, слова та їх переклади. Компоненти рівня бізнес-логіки виконують різні завдання, такі як перевірка прав доступу до адміністративної панелі та обробка запитів на додавання, редагування та видалення даних.

Компоненти рівня інтерфейсу користувача забезпечують користувачам зручний та простий інтерфейс для перегляду та пошуку мов, слів та їх перекладів.

Хоча проєкт є досить функціональним, однак можна вважати, що він може бути поліпшеним, особливо з точки зору функціональності пошуку. Додавання можливості пошуку перекладів за допомогою фрази замість окремих слів може бути корисною функцією для покращення досвіду користувача та розширення функціоналу додатку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Б. Бейтс, К. Сьєрра, Б. Бейтс. Вивчаємо Java. Пер. з англ. - СПб.: Пітер, 2013. - 704 с.
2. Д. Макфарланд, Д. Лебровські. Head First HTML and CSS. 2-е изд. - М.: ТОВ "И.Д. Вільямс", 2014. - 720 с.
3. Б. Фленаган. JavaScript. Докладний посібник. 6-е изд. - СПб.: Пітер, 2012. - 1104 с.
4. М. Мейєр. CSS. Каскадні таблиці стилів. Докладний посібник. 3-е изд. - СПб.: Питер, 2011. - 550 с.
5. Б. Бейтс, К. Сьєрра, Б. Бейтс. Вивчаємо SQL. Пер. з англ. - М.: ТОВ "И.Д. Вільямс", 2014. - 672 с.
6. Ш. Шилдт. Java 8. Повне керівництво. 9-е изд. - СПб.: Пітер, 2015. - 1376 с.
7. М. Фаулер. Архітектура корпоративних програмних додатків. Пер. з англ. - М.: ТОВ "И.Д. Вільямс", 2011. - 576 с.
8. Э. Фрімен, Е. Робсон. Патерни проектування. Пер. з англ. - М.: ТОВ "И.Д. Вільямс", 2012. - 416 с.
9. И. Соммервіл. Інженерне проектування програмного забезпечення. Пер. з англ. - М.: Видавничий дім "Вільямс", 2017. - 864 с.

ДОДАТКИ

```

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/addLanguageServlet")
public class AddLanguageServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String language = request.getParameter("language");

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        try {
            Class.forName("org.sqlite.JDBC");
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "INSERT INTO languages (name) VALUES (?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, language);
            pstmt.executeUpdate();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

```

```

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/addTranslationServlet")
public class addTranslationServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int wordId = Integer.parseInt(request.getParameter("wordId"));
        int languageId = Integer.parseInt(request.getParameter("languageId"));
        String translation = request.getParameter("translation");
    }
}

```

```

// Підключення до бази даних
String jdbcUrl = "jdbc:sqlite:dictionary.db";
try {
    Connection conn = DriverManager.getConnection(jdbcUrl);
    String sql = "INSERT INTO translations (word_id, language_id,
translation) VALUES (?, ?, ?)";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, wordId);
    pstmt.setInt(2, languageId);
    pstmt.setString(3, translation);
    pstmt.executeUpdate();
    conn.close();
} catch (SQLException e) {
    e.printStackTrace();
}

response.sendRedirect("admin.jsp");
}
}

```

```

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

```

```

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

```

```

@WebServlet("/addWordServlet")
public class addWordServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String word = request.getParameter("word");
        int languageId = Integer.parseInt(request.getParameter("languageId"));

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "INSERT INTO words (word, language_id) VALUES (?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, word);
            pstmt.setInt(2, languageId);
            pstmt.executeUpdate();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

```



```

public class DefaultData {
    public static final String DB_PATH =
        "jdbc:sqlite:C:/Users/Max/Desktop/dictionary/identifier.sqlite";
}

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/deleteLanguageServlet")
public class DeleteLanguageServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int languageId = Integer.parseInt(request.getParameter("languageId"));

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "DELETE FROM languages WHERE id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, languageId);
            pstmt.executeUpdate();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/deleteTranslationServlet")
public class deleteTranslationServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    int translationId = Integer.parseInt(request.getParameter("translationId"));

    // Підключення до бази даних
    String jdbcUrl = DefaultData.DB_PATH;
    try {
        Connection conn = DriverManager.getConnection(jdbcUrl);
        String sql = "DELETE FROM translations WHERE id = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, translationId);
        pstmt.executeUpdate();
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    response.sendRedirect("admin.jsp");
}
}

```

```

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/deleteWordServlet")
public class deleteWordServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int wordId = Integer.parseInt(request.getParameter("wordId"));

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "DELETE FROM words WHERE id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, wordId);
            pstmt.executeUpdate();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

```

```

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/editLanguageServlet")
public class EditLanguageServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int languageId = Integer.parseInt(request.getParameter("languageId"));
        String language = request.getParameter("language");

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "UPDATE languages SET name = ? WHERE id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, language);
            pstmt.setInt(2, languageId);
            pstmt.executeUpdate();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

```

```

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

@WebServlet("/editTranslationServlet")
public class editTranslationServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int translationId = Integer.parseInt(request.getParameter("translationId"));
        String translation = request.getParameter("translation");

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;

```

```

        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "UPDATE translations SET translation = ? WHERE id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, translation);
            pstmt.setInt(2, translationId);
            pstmt.executeUpdate();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
@WebServlet("/editWordServlet")
public class editWordServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int wordId = Integer.parseInt(request.getParameter("wordId"));
        String word = request.getParameter("word");
        int languageId = Integer.parseInt(request.getParameter("languageId"));

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "UPDATE words SET word = ?, language_id = ? WHERE id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, word);
            pstmt.setInt(2, languageId);
            pstmt.setInt(3, wordId);
            pstmt.executeUpdate();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        response.sendRedirect("admin.jsp");
    }
}

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.HttpServlet;

```

```

import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

@WebServlet("/searchWordServlet")
public class searchWordServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int languageId = Integer.parseInt(request.getParameter("language"));
        String word = request.getParameter("word");

        // Підключення до бази даних
        String jdbcUrl = DefaultData.DB_PATH;
        List<String> translations = new ArrayList<>();
        try {
            Connection conn = DriverManager.getConnection(jdbcUrl);
            String sql = "SELECT t.translation FROM words w " +
                "INNER JOIN translations t ON w.id = t.word_id " +
                "WHERE w.word = ? AND t.language_id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, word);
            pstmt.setInt(2, languageId);
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                String translation = rs.getString("translation");
                translations.add(translation);
            }
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        request.setAttribute("result", translations);
        request.getRequestDispatcher("user.jsp").forward(request, response);
    }
}

```