ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»
(освітньо-кваліфікаційний рівень)

на тему: Інформаційна система ідентифікації науково-технічної термінології

Information system for identification of scientific and technical terminology

Виконав: студент денної форми навчання
спеціальності 123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Освітня програма «Комп'ютерна інженерія»
(назва освітньої програми)
Абасов Альберт Реймірович
(прізвище, ім'я, по-батькові)

Керівник ст. викл. Лісіцина І.М.
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент к.т.н., доц. Волощук Л.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2023 р.

Завідувач кафедри
_____ _Євгеній МАЛАХОВ_
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «___» _____ 2023 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК
_____ _Алла КОБОЗЄВА_
(підпис) (ім'я, прізвище)

Одеса - 2023

АНОТАЦІЯ

Метою даної дипломної роботи є розробка електронного багатомовного термінологічного словника-довідника.

В роботі розглянуті основні поняття та проблеми термінології, приведений аналіз багатьох існуючих електронних словників. На цій основі сформульовані вимоги до системи, обрана архітектура системи, проведене проектування бази даних, розроблений інтерфейс користувача.

Результатом дипломної роботи є система, яка дозволяє спеціалістам публікувати багатомовні термінологічні словники в мережі Інтернет за рахунок наповнення бази даних.

Важливою особливістю створеної системи є її відкритість, що припускає можливість розширення як самої бази термінів, так і даних з області опису термінів (додавання інших мов, інших відомостей про терміни).

У роботі приведений огляд використовуваних засобів реалізації: мови програмування Kotlin, фреймворка Spring Framework, систему управління базами даних PostgreSQL та шаблону проектування MVC, а також даний повний опис функціонування системи і деталей реалізації.

ABSTRACT

The purpose of this graduate work is to develop an electronic multilingual terminological dictionary.

The work examines the main concepts and problems of terminology, provides an analysis of many existing electronic dictionaries. On this basis, the requirements for the system were formulated, the system architecture was selected, the database was designed, and the user interface was developed.

The result of the graduate work is a system that allows specialists to publish multilingual terminological dictionaries on the Internet by filling the database.

An important feature of the created system is its openness, which implies the possibility of expanding both the term base itself and data from the field of term description (adding other languages, other information about terms).

The work provides an overview of the implementation tools used: the Kotlin programming language, the Spring Framework, the PostgreSQL database management system and the «Model-View-Controller» design template, as well as a full description of the system's functioning and implementation details.

ЗМІСТ

ВСТУП

У сучасному світі, де міжнародні комунікації стають все більш важливими, вміння перекладати термінологію є необхідною складовою частиною навчання та професійної діяльності. Потреба у засобах для ефективного спілкування між різномовними користувачами стає невід'ємною, не вистачає упорядкованості та суворої стандартизації термінології.

Технічна термінологія переважно використовує англійську мову, але сучасні науково-технічні статті все більше друкуються різними мовами, що ускладнює коректне розуміння їх змісту. Теоретично, терміни під час перекладу мають бути повністю еквівалентними, тобто між термінами з різних мов повинна бути постійна й рівнозначна відповідність, незалежна від контексту. Проте, часто одного перекладу або одного джерела недостатньо для забезпечення повної відповідності.

Інформаційні системи, здатні перекладати та пояснювати значення слів з однієї мови на іншу, стають необхідним інструментом для подолання мовних бар'єрів і забезпечення ефективного міжкультурного спілкування. Вони надають можливість користувачам з різних мовних середовищ швидко знайти переклади та визначити значення слів.

Особистий професійний словник для фахівця – це якість перекладу, економія зусиль та часу, які за несприятливого розкладу йдуть на пошуки потрібного слова в інших джерелах.

Метою проекту є створення функціонального, зручного у використанні багатомовного тлумачного словника, який допоможе користувачам з різних мовних груп отримувати швидкий доступ до перекладів та вичерпної лексикографічної інформації.

Для досягнення такої мети потрібно розв’язати наступні задачі:

- сформулювати задачі інформаційної системи і провести дослідження предметної області;
- провести огляд існуючих аналогів;
- сформулювати вимоги до створюваної системи;
- побудувати загальну архітектуру системи;
- зробити вибір засобів та технологій розробки;
- виконати проектування бази даних згідно із задачами користувачів;
- розробити клієнтську частину застосунку;
- розробити серверну частину застосунку;
- перевірити працездатність програмної реалізації.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Задачі інформаційної системи

До основних задач інформаційної системи відносяться наступні:

- надання можливості відвідувачам створювати та редагувати свої облікові записи;
- додавання адміністратором рубрик на різних мовах, їх редагування та видалення;
- надання можливості відвідувачам переглядати терміни з їх інтерпретаціями на різних мовах з використанням різноманітних фільтрів.

1.2 Аналіз предметної області

Термін - це спеціальне слово або висловлення, яке використовується в певній професійній галузі та особливих умовах. Він служить для назви конкретного поняття в цій галузі професійних знань. Термінологія - це сукупність таких спеціальних слів і висловлень в певній галузі знання або виробництва, а також наука, яка вивчає структуру та використання цих термінів.

Термінологія є окремим сектором будь-якої мови, який тісно пов'язаний з професійною діяльністю. У кожній галузі науки, техніки або виробництва терміни формують свої системи, які базуються на понятєвих зв'язках професійних знань і виражаються за допомогою мовних засобів.

Термін відноситься до спеціальної термінологічної системи, яка використовується в різних галузях. Концептуальний зміст терміна визначається його місцем у цій системі. Кожен термін має свою власну дефініцію, яка точно визначає його значення у межах даної області, зазвичай однозначне, хоча одне й те саме слово може бути терміном у

різних сферах знань (наприклад, термін "хвиля" використовується в гідравліці, радіотехніці та оптиці). Терміни існують не просто у мові, а в рамках конкретної термінології. Це означає, що коли слово потрапляє до певної термінології, воно отримує чітке значення.

Термінологічна дефініція надає загальне уявлення про об'єкт, одночасно усуваючи можливу неоднозначність, яка притаманна загальним словам у мові. Дефініція повинна бути порівнянною з об'єктом, який вона визначає, не містити протиріччя. Термін завжди виражає конкретне спеціальне поняття і, у цьому контексті, має однозначне значення. Проте, можна знайти кілька визначень одного терміна у словнику. Це зазвичай відбувається через використання терміна для позначення спеціальних понять, що входять до різних понятійних систем. Багатозначність термінології зазвичай легко усувається шляхом визначення конкретної системи до якої належить термін, і в рамках цієї системи термін матиме лише одне значення. З усього вище сказаного випливає, що для створення інформаційної системи, яка буде корисною для складання та використання термінологічних словників, необхідно забезпечити однозначність і системність термінів. Одним з таких засобів може бути рубрикація.

1.3 Функціональне призначення словників

Словник - це довідкове видання, яке містить зібрані слова, і надає інформацію про їх значення, приналежність до певної групи, переклад на інші мови.

Основною функцією всіх довідкових видань, зокрема словників, є надання інформації. Словники, як довідкові посібники з мови, виконують комунікативну функцію, допомагаючи учасникам мовного співтовариства в вирішенні труднощів, пов'язаних з використанням мови. Крім того, словники виконують важливу навчальну функцію.

2 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

2.1 «Google Translate»

Є провідним гравцем у галузі обробки природної мови. Перекладач підтримує 103 мови, що є абсолютним рекордом серед подібних програм, що не лише перекладає введений текст, але також розпізнає та перекладає написи на фотографіях, або з екрану мобільного телефона чи з сторінки зошита у реальному часі. Більше того, ви можете просто проговорити декілька фраз, і програма розпізнає слова та перекладе їх.

Також зберігає історію минулих перекладів і дозволяє створювати списки для зручності.

Однак не має засобів уточнення понятійної системи, і більше спрямований на перекладі слова, а не його дефініції, тому він не може вважатися термінологічним.

Приклад використання «Google Translate» зображено на рисунку 2.1.

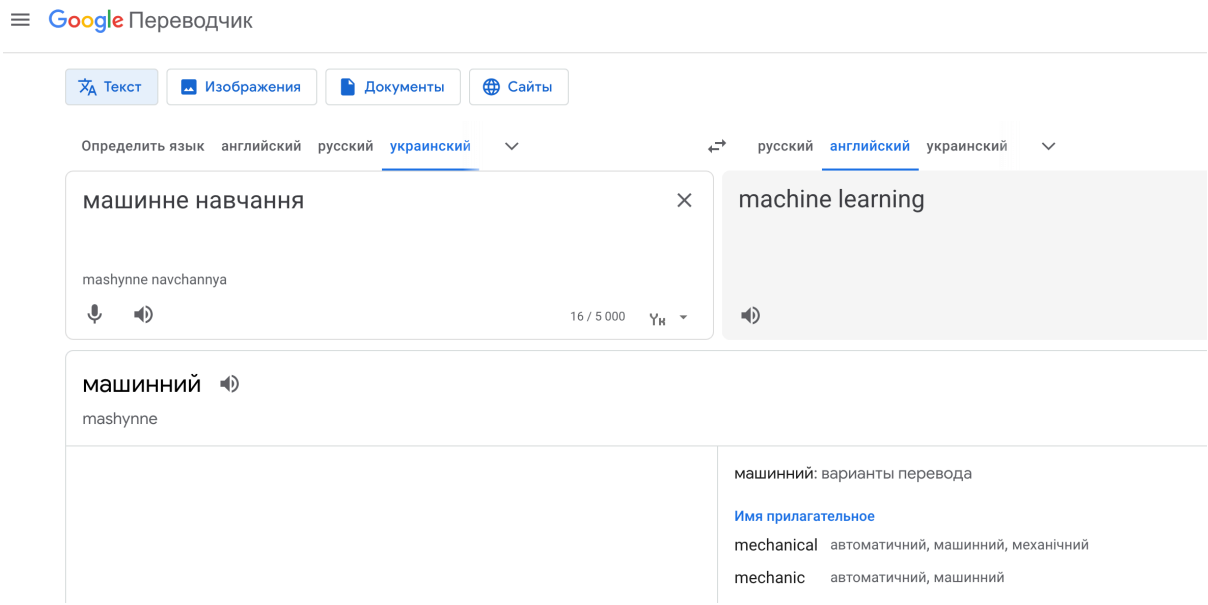

Рисунок 2.1 – приклад використання «Google Translate»

2.2 «Multitran»

Інтернет-система двомовних словників, що продовжує поповнюватись товариством користувачів, до якої спочатку входили онлайн-словники 12 мов, а з 2016 року після оновлення – ще більше.

Особливістю «Мультитрану» є можливість користувачів робити свій внесок, доповнюючи його.

Однак «Multitran» не є багатомовним словником, адже надає тільки двомовні словники, з однієї мови на іншу.

Приклад використання «Multitran» зображено на рисунку 2.2.

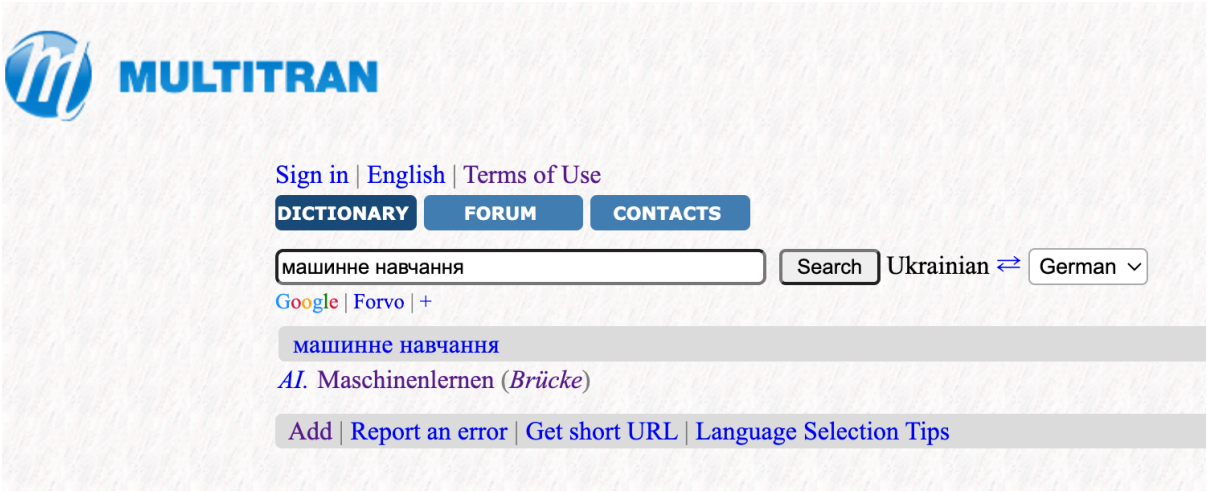

Рисунок 2.2 – приклад використання «Multitran»

2.3 «Langenscheidt»

Німецьке видавництво, яке спеціалізується на мовних довідниках. Крім одномовних словників, Langenscheidt також видає двомовні словники та дорожні розмовники. З переваг містить фонетичну транскрипцію іншомовних слів і приклади використання слова у реченні.

Однак, як і попередній словник, не є багатомовним, надаючи переклади тільки з однією мови на іншу.

Приклад використання «Langenscheidt» зображено на рисунку 2.3.

Рисунок 2.3 – приклад використання «Langenscheidt»

Основуючись на проведеному аналізі предметної області та існуючих аналогів, можна сформулювати наступні вимоги до проектованої системи:

– центальним поняттям системи повинно бути тлумачення термінів;
– система повинна підтримувати багатомовність словника, без обмежень на кількість мов;
– забезпечувати зручність для пошуку;
– підтримувати рубрикацію для збереження однозначності термінів та зручності користувачів;
– можливість збереження перекладу для полегшення користування системою;
– використання бази даних для зберігання інформації.

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Вибір архітектури та шаблону проектування

Для розв’язання поставлених задач була використана триланкова архітектура клієнт-сервер (рисунок 3.1), що включає в себе три компоненти: рівень представлення, рівень бізнес-логіки і рівень даних.

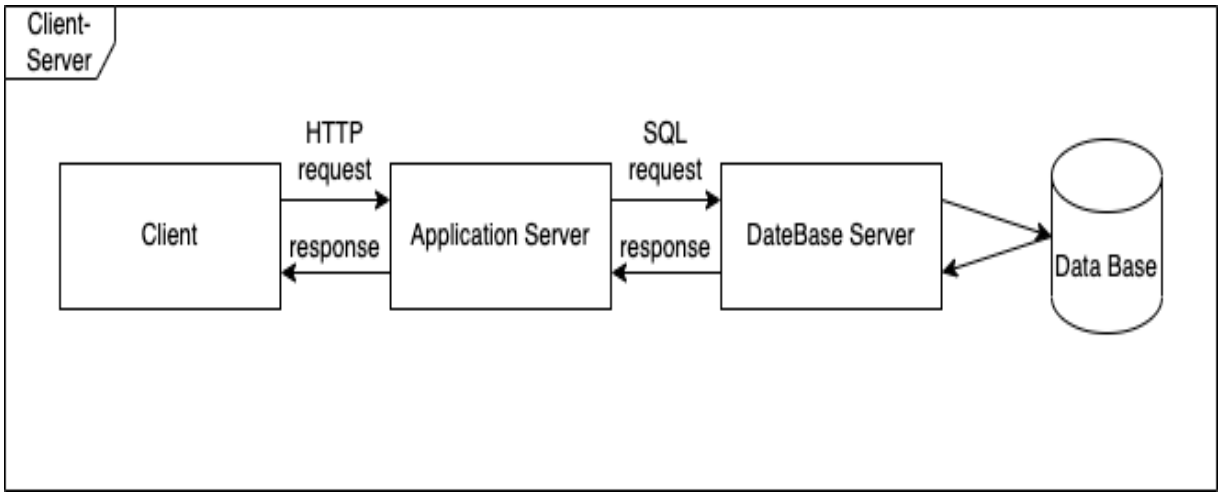

Рисунок 3.1 – триланковий клієнт-сервер

У якості шаблону проектування було обрано MVC (Model, View, Controller). Цей шаблон дозволяє відокремити різні компоненти системи і забезпечує більшу гнучкість, розширюваність та перевикористання коду. Основні ролі в MVC такі:

– модель (model): представляє дані та логіку їх обробки. Вона відповідає за доступ до даних, їх збереження. Модель не залежить від інших компонентів системи і може бути використана у різних контекстах;

– представлення (view): відповідає за відображення даних користувачу та взаємодію з ним. Воно отримує дані від контролера і відображає їх в інтерфейсі користувача;
– контролер (controller): відповідає за обробку вхідних даних від користувача та взаємодію з моделлю та представленням. Він обробляє запити користувача, оновлює модель та вибирає відповідне представлення для відображення результатів.

Блок-схема шаблону проектування MVC представлена на рисунку 3.2

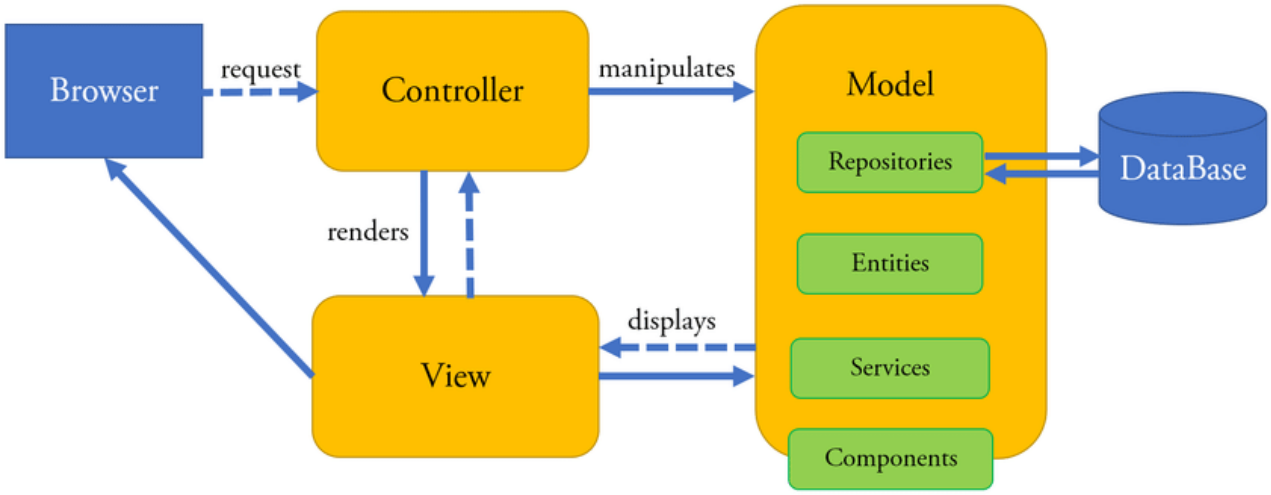

Рисунок 3.2 – шаблон проектування MVC

3.2 Програмна модель застосунку

Словник являє собою один застосунок, яким користуються адміністратор та звичайні користувачі. При авторизації в системі відповідно до ролі користувача, застосунок завантажує відповідну

сторінку, яка містить необхідну інформацію та інструменти для вирішення користувачем своїх задач.

Застосунок дозволяє користувачам увійти в систему під будь-якою роллю шляхом авторизації у відповідній формі. Клієнт надсилає запит, який визначає, який ресурс має обробити цей запит. Представлення (View) отримує запит і повертає користувачеві відповідь, якщо для цього не потрібна взаємодія з базою даних. У випадку, якщо запит вимагає доступу до бази даних, представлення передає інформацію до наступного рівня системи, де виконуються операції з базою даних (Model), після чого відповідь повертається клієнту.

3.3 Вибір програмного забезпечення

Інформаційна система реалізована у формі веб-застосунка, який дозволяє користувачам взаємодіяти з системою за допомогою веб-браузера. Щоб скористатися системою, достатньо лише ввести потрібну веб-адресу в рядок пошуку браузера.

Для забезпечення роботи з базою даних було обрано систему управління базами даних (СУБД) PostgreSQL. Для розробки програмного застосунку використовується мова програмування Kotlin та фреймворк Spring (для реалізації серверної частини), а також шаблонізатор Thymeleaf, який є частиною цього фреймворку.

У сучасний час існує розмаїття систем керування базами даних, таких як MySQL, Oracle, MongoDB, PostgreSQL та інші. Серед цих варіантів, для роботи було обрано PostgreSQL з наступних причин:

– Реалізація реляційної моделі даних, що є зрозумілою для кінцевого користувача. Ця модель дозволяє логічно організовувати дані та встановлювати зв'язки між ними, спрощуючи роботу з базою даних;

- Гнучкий механізм управління правами користувачів бази даних за допомогою ролей. PostgreSQL надає зручні інструменти для встановлення рівнів доступу та обмежень, що дозволяє точно керувати доступом до даних в системі;
- Підтримка мови plpgSQL як розширення стандартного SQL. Ця мова надає розширені можливості для написання ефективних збережених процедур, що дозволяють зберігати та виконувати складні операції над даними без необхідності використовувати зовнішні інструменти.

Обираючи PostgreSQL, ми отримуємо потужний інструмент для зберігання та управління нашими даними, який поєднує зручність використання, безпеку та можливості розширення для задоволення наших потреб у проекті.

Spring Framework - це потужний фреймворк з відкритим вихідним кодом для розробки веб-застосунків на Java-платформі. Він складається з набору взаємопов'язаних міні-фреймворків, які можна підключати окремо в залежності від потреб проекту. Одним з ключових модулів є Spring Data, який надає інструменти для роботи з реляційними та нереляційними базами даних. Він містить набір інтерфейсів, які спрощують роботу з даними і дозволяють зручно взаємодіяти з базою даних. Крім того, Spring Framework має модуль Spring MVC, що надає структуру для створення веб-застосунків зі слабким зв'язуванням. Цей модуль розділяє основні аспекти розробки: об'єкти, бізнес-логіку та зовнішній вигляд програми. Він спрощує процес створення веб-інтерфейсу та дозволяє зосередитися на реалізації функціональності. Spring Framework є високопродуктивним інструментом, який сприяє розробці надійних та масштабованих веб-застосунків. Він дозволяє розподілити відповідальність між різними модулями, що полегшує розробку, тестування і підтримку програмного забезпечення.

Thymeleaf - це сучасний серверний механізм Java-шаблонів, призначений для веб-та автономних середовищ. Він може обробляти різноманітні типи файлів, такі як HTML, XML, JavaScript, CSS і навіть простий текст. Основна мета Thymeleaf полягає у створенні зручного способу шаблонізації. Thymeleaf ґрунтується на концепції Natural Templates, що дозволяє впроваджувати логіку безпосередньо у файли шаблонів. Це означає, що шаблони не мають негативного впливу на відображення дизайну прототипу. Thymeleaf надає зручні інструменти для взаємодії з даними, здатність до умовної обробки, ітерацій та інших операцій, що спрощують роботу з шаблонами. Завдяки своїм можливостям, Thymeleaf є популярним вибором для розробки веб-застосунків, де важливо ефективно використовувати шаблони та забезпечувати їх гнучкість та легкість зрозуміння. Він інтегрується з багатьма веб-фреймворками та надає розробникам потужні інструменти для створення елегантних і сучасних веб-інтерфейсів.

Для забезпечення взаємодії між сервером застосунка та сервером бази даних, було використано JDBC (Java DataBase Connectivity) - платформово-незалежний промисловий стандарт для взаємодії Java-застосунків з різними системами управління базами даних. JDBC реалізований у вигляді пакету java.sql, який входить до складу Java SE. JDBC базується на концепції драйверів, що дозволяють отримувати з'єднання з базою даних за допомогою спеціально визначеного URL. Драйвери можуть бути динамічно завантажені під час роботи програми. Після завантаження, драйвер автоматично реєструє себе та відповідає на виклики, коли програма вимагає URL-адресу, що включає протокол, який відповідає даному драйверу.

4 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

4.1 Специфікація вимог до бази даних

Першим кроком у проектуванні баз даних є аналіз реального предметного середовища. Мета цього етапу полягає в тому, щоб отримати загальне уявлення про предметну область, зібрати елементи даних, які потрібно буде відображати у базі даних, сформулювати обмеження цілісності, яким повинні задовольняти дані майбутньої бази, а також визначити основні транзакції та документи, які база даних має забезпечити.

У цьому етапі враховуються як функціональні, так і нефункціональні вимоги. Функціональні вимоги описують послуги та функції, які має надавати розроблювана система. Нефункціональні вимоги визначають обмеження, які впливають на роботу системи, такі як кількість одночасних користувачів або стандарти, якими повинна відповідати система.

Результатом цього етапу проектування є підготовка специфікацій вимог, які є характерними для даної системи. У специфікації визначаються вимоги до інформації, яка буде зберігатися у створюваній базі даних.

Виходячи з вимог предметної галузі в базі даних повинна зберігатись наступна інформація:

- поняття;
- терміни;
- тлумачення;
- рубрики;
- мови.

Поняття є центральною сутністю термінологічного словника, а його іменем є термін (один або кілька), з яким пов'язане його тлумачення.

Термін означає назву поняття на певній мові. У базі даних необхідно зберігати термін і посилання на мову, до якої він належить.

Для того, щоб надати користувачу найбільш підходящу назву певного поняття для досягнення його цілей, потрібно встановити класифікацію сутності "термін". Типи термінів можуть бути наступними:

- стандартизований;
- рекомендований;
- додатковий;
- застарілий;
- жаргонний.

Під тлумаченням поняття розуміється його вербальне визначення. Тлумачення поняття може бути наведено на різних мовах.

Для зручності використання словника необхідно мати можливість рубрикації. Кожне поняття може бути віднесено до однієї або декількох рубрик. Рубрикація здійснюється відповідно до тлумачення поняття. Це дозволяє забезпечити вимогу щодо однозначності терміна

З метою забезпечення зручності та корисності для фахівців у різних областях, додаток має надавати можливість вибору мови, яку користувач бажає використовувати.

4.2 Побудова моделі бази даних і визначення типів зв’язків

В інформаційній моделі слід виділити наступні сутності:

- користувач – містить інформацію про користувача системи (логін, пароль, пошта, роль, статус);
- рубрика – містить найменування рубрики, мову;
- поняття – містить інформацію щодо терміну, його інтерпретації і рубрики, до якої він відноситься;
- термін – містить найменування, тип та мову;

– інтерпретація – містить текст що відображає дефініцію терміну, мову.

Між сутностями у базі даних наявні 2 типи зв’язків:

– «один-до-багатьох»;

– «багато-до-багатьох».

Зв’язок «один-до-багатьох» реалізований між поняттям та терміном, а також між поняттям та інтерпретацією, адже кожному поняттю відповідають багато термінів і його інтерпретацій на різних мовах. Для формалізації цього типу зв’язку до таблиць terms та interpretations було додано зовнішній ключ, який посилається на унікальний атрибут таблиці concepts.

Зв'язок «багато-до-багатьох» наявний між поняттям і рубрикою. Кожна рубрика може містити багато понять, в свою чергу, одне і те саме поняття може належати до багатьох рубрик, відповідним різним понятійним підсистемам. Для формалізації цього типу зв’язку була додана допоміжна таблиця rubric_reference, до якої було додано два зовнішніх ключі, які посилаються на унікальні атрибути таблиць concepts та rubrics відповідно.

4.3 Визначення атрибутів і їх доменів

У специфікації необхідно виділити атрибути сутностей, які можуть бути представлені іменниками або відповідними поєднаннями слів. Кожен атрибут описує певний аспект конкретної сутності або зв'язку між ними. Інформація про виділені атрибути та їх приналежність до відповідних сутностей наведена у таблиці 4.1

Таблиця 4.1 – Опис сутностей та їх властивостей

Властивість	Опис	Обмеження
Об’єкт “users”		
id	Ідентифікатор користувача системи	Первинний ключ, непорожнє
username	Логін користувача системи	<= 50 символів, унікальне, непорожнє
password	Пароль користувача системи	<= 255 символів, непорожнє
email	Пошта користувача системи	<= 50 символів, унікальне, непорожнє, оброблюється регулярним виразом для типових email адрес
role	Роль користувача системи	Непорожнє, один із варіантів значень домену user_role; значення (‘ADMIN’, ‘VISITOR), значення за замовчуванням - ‘VISITOR’
language	Мова	один із варіантів значень домену language_type; Значення (‘ENGLISH’, ‘FRECH’, ‘GERMAN’)

Продовження таблиці 4.1

Об’єкт “rubrics”		
id	Ідентифікатор рубрики	Первинний ключ, непорожнє
name	Найменування рубрики	<= 50 символів, непорожнє
language	Мова	Непорожнє, один із варіантів значень домену language_type; Значення (‘ENGLISH’, ‘FRECH’, ‘GERMAN’)
Об’єкт “concepts”		
id	Ідентифікатор терміну	Первинний ключ, непорожнє
Об’єкт “rubric_reference”		
concept_id	Ідентифікатор поняття	Непорожнє, частина складного первинного ключа (concept_id, rubric_id), зовнішній ключ для зв’язку з сутністю cocnepts (id)
rubric_id	Ідентифікатор рубрики	Непорожнє, частина складного первинного ключа (concept_id, rubric_id), зовнішній ключ для зв’язку з сутністю rubrics (id)

Продовження таблиці 4.1

Об’єкт “terms”		
id	Ідентифікатор терміну	Непорожє, частина складного первинного ключа (id, language), частина складного зовнішнього ключа (id, language) для зв’язку з сутністю concepts (term_id, languge)
name	Найменування терміну	<= 50 символів, непорожнє
type	Тип терміну	Непорожнє, один із варіантів значень домену term_type; Значення (‘STANDARDIZED’, ‘RECOMMENDED’, ‘ADDITIONAL’, ‘DEPRECATED’, ‘SLANGY’)
language	Мова	Непорожнє, один із варіантів значень домену language_type; Значення (‘ENGLISH’, ‘FRECH’, ‘GERMAN’)

Продовження таблиці 4.1

concept_id	Ідентифікатор поняття	Непорожнє, зовнішній ключ для зв’язку з сутністю concepts (id)
Об’єкт “interpretations”		
id	Ідентифікатор інтерпретації	Первинний ключ (id, language), непорожнє, частина складного зовнішнього ключа (id, language) для зв’язку з сутністю concepts
text	Текст інтерпретації	<= 200 символів, непорожнє
language	Мова	Непорожнє, частина складного зовнішнього ключа (id, language) для зв’язку з сутністю concepts, один із варіантів значень домену language_type; Значення (‘ENGLISH’, ‘FRECH’, ‘GERMAN’)
concept_id	Ідентифікатор поняття	Непорожнє, зовнішній ключ для зв’язку з сутністю concepts (id)

Схему бази даних, що ілюструє сутності, їх атрибути з відповідними доменами, а також зв’язки між ними, детально представлено на рисунку 4.2. Це графічне зображення надає візуальний огляд структури бази даних, вказуючи на взаємозв’язки між сутностями та їх атрибутами. Кожна сутніть представлена своєю назвою, а атрибути відображаються разом з відповідними доменами, що визначають множину значень, які можуть бути збережені в цих атрибутах.

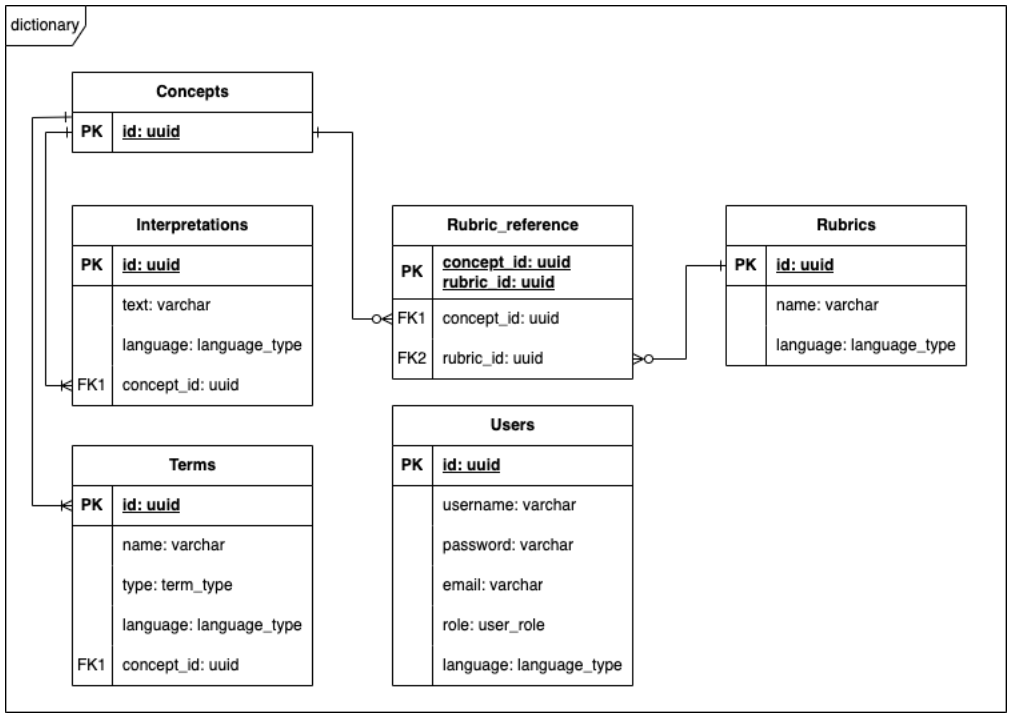

Рисунок 4.2 - Схема бази даних

5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

5.1 Реалізація бази даних

В цьому розділі описано створення основних об’єктів бази даних. Як приклад наведено програмний код створення таблиці users (включно зі створенням типу user_role) (рисунок 5.1). Повний код створення всіх таблиць БД наведено в додатку Б.

```
CREATE TYPE user_role AS ENUM ('ADMIN', 'VISITOR');
CREATE TABLE users(
     id UUID PRIMARY KEY NOT NULL,
     username VARCHAR(50) UNIQUE NOT NULL,
     password VARCHAR(255) NOT NULL,
     email VARCHAR(50) UNIQUE NOT NULL CHECK (email ~ '^.+@[a-
z]+\.[a-z]+$'),
     role user_role NOT NULL DEFAULT ('VISITOR'),
     language language_type
);
```

Рисунок 5.1 – Створення таблиці users та типу user_role

Створення таблиці виконується шляхом використання команди CREATE TABLE у контексті обраної Системи Управління Базами Даних (СУБД), у даному випадку PostgreSQL. Після цього слід вказати ім'я таблиці. Кожному полю таблиці присвоюється тип даних, який відповідає вибраній СУБД, а також встановлюються обмеження цілісності. Поле "id" визначається як первинний ключ (PRIMARY KEY) та має тип UUID. Поля "email" та "username" мають бути унікальними (UNIQUE). Використання NOT NULL при описі полів таблиці означає, що вони не можуть мати пусті значення. Ключове слово DEFAULT вказує, що значення цього поля приймаються за замовчуванням. Ключове слово CHECK використовується для перевірки введених значень на відповідність визначеним умовам.

Наприклад, поле "email" може бути перевірено, чи введений email відповідає формату правильної електронної пошти.

5.2 Основні запити щодо створення і отримання даних

– створення нової рубрики (замість знаків «?» підставляють ідентифікатор рубрики, найменування рубрики та мова). Лістинг відповідного запиту приведений на рисунку 5.2.

```
INSERT INTO rubrics (id, name, language)
VALUES (?, ?, cast(? as language_type))
```

Рисунок 5.2 – Створення нової рубрики

– створення нового поняття (замість знака «?» підставляється ідентифікатор поняття). Лістинг відповідного запиту приведений на рисунку 5.3.

```
INSERT INTO concepts (id)
VALUES (?))
```

Рисунок 5.3 – Створення нового поняття

– створення посилання на рубрику (замість знаків «?» підставляють ідентифікатор поняття, ідентифікатор рубрики та мова). Лістинг відповідного запиту приведений на рисунку 5.4.

```
INSERT INTO rubric_reference (concept_id, rubric_id,
language) VALUES (?, ?, cast(? as language_type))
```

Рисунок 5.4 – Створення посилання на рубрику

– створення відповідного терміну (замість знаків «?» підставляють ідентифікатор терміну, найменування терміну, тип терміну, мова, ідентифікатор поняття). Лістинг відповідного запиту приведений на рисунку 5.5.

```
INSERT INTO terms (id, name, type, language, concept_id)
VALUES (?, ?, cast(? as term_type), cast(? as
language_type), ?)
```

Рисунок 5.5 – Створення терміну

– створення інтерпретації терміну (замість знаків «?» підставляють ідентифікатор інтерпретації, текст інтерпретації, мова, ідентифікатор поняття). Лістинг відповідного запиту приведений на рисунку 5.6.

```
INSERT INTO interpretations (id, text, language, concept_id)
VALUES (?, ?, cast(? as language_type), ?)
```

Рисунок 5.6 – Створення інтерпретації

– отримання повної інформації про поняття за обраною мовою (замість знаків «?» підставляють обрана мова, ідентифікатори рубрик, за якими буде фильтрувати поняття пошук). Лістинг відповідного запиту приведений на рисунку 5.7.

```
SELECT t.term_id         as term_id,
       t.name            as term_name,
       i.text            as interpretation_name,
       t.type,
       array_agg(r.name) as rubrics,
       t.language        as language
FROM concepts c
         JOIN terms t ON t.concept_id = c.id
         JOIN interpretations i on i.concept_id = c.id
         JOIN rubric_reference rr on c.id = rr.concept_id
         JOiN rubrics r on r.id = rr.rubric_id
WHERE t.language = ? and r.rubric_id in (?, ..., ?)
GROUP BY t.term_id, t.name, i.text, t.type, t.language;
```

Рисунок 5.7 – Пошук понять за обраними мовою та списком рубрик

6 ПРОГРАМНА РЕАЛІЗАЦІЯ КЛІЄНТСЬКОГО ЗАСТОСУНКУ

Spring MVC реалізує архітектурний шаблон MVC (Model-View-Controller). Як приклад, наведемо View для проходження процесу реєстрації користувача веб-застосунка (рисунок 6.1):

```
<body>
<div class="container">
<form method="post" th:action="@{/createUser}"
th:object="${user}">
<h2 style="color: #dddddd; width: 400px">Please, enter
following information</h2>
<p>
<label for="username">Username</label>
<input type="text" id="username" placeholder="Username"
th:field="*{username}">
</p>
<p>
<label for="email">Username</label>
<input type="text" id="email" placeholder="email@gmail.com"
th:field="*{email}">
</p>
<p>
<label for="password">Password</label>
<input type="password" id="password" class="form-control"
placeholder="Password" th:field="*{password}">
</p>
<p>
<label for="verifyPassword" class="sr-only">Repeat
password</label>
<input type="password" id="verifyPassword"
placeholder="Confirm password" th:field="*{verifyPassword}">
</p>
<button class="btn btn-success" type="submit">Sign up</button>
<a style="color: springgreen" href="/login">Login</a>
</form>
</div>
</body>
```

Рисунок 6.1 – View для проходження процесу реєстрації

Як можна помітити, шаблон складається з HTML-коду, в якому також присутні фігурні дужки, що нагадують синтаксис мови Kotlin. Ці дужки дозволяють динамічно генерувати вміст та структуру сторінки в залежності від переданих даних.

Для кожного запиту, який користувач відправляє з клієнтської частини, існує контролер на серверній частині застосунку, який обробляє отримані дані. Нижче на рисунку 6.2 наведений приклад контролера, який відображає домашню сторінку застосунку залежно від користувача:

```
@GetMapping("/home")
fun homePage(): String {
     val authentication = SecurityContextHolder
                                  .getContext()
                                  .authentication

     val user = userService.
          loadUserByUsername(authentication.name)!!

     return if (user.role == UserRole.VISITOR) {
          "/homeVisitor"
     } else {
          "/homeAdmin"
     }
}
```

Рисунок 6.2 – View для відображення домашньої сторінки

Користувач вводить свої логін та пароль. Якщо такий користувач існує, він буде перенаправлений на відповідну сторінку, що відповідає його групі користувачів. У випадку, якщо введені дані не відповідають жодному користувачеві, користувачу знову буде запропоновано ввести свої дані.

Головна сторінка користувача змінюється відповідно до визначених можливостей та функціоналу кожної групи.

7 БЕЗПЕКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

Безпека забезпечується шляхом створення ролей і наділення їх відповідними привілеями. В інформаційній системі реалізовано 2 ролі: Адміністратор (A) та Відвідувач (B).

У таблиці 7.1 наведені привілеї ролей на таблиці та елементи бази даних. При цьому використані наступні позначення:

– I (Insert) – додавання нової інформації до таблиці;

– S (Select) – перегляд вмісту таблиці;

– U (Update) – редагування існуючої інформації;

– D (Delete) – видалення інформації з таблиці.

Таблиця 7.1. – Права доступу користувачів до бази даних

Сутності бази даних	Ролі	
	Адміністратор (ADMIN)	Відвідувач (VISITOR)
Таблиці		
users	ISUD	SU
rubrics	ISUD	S
concepts	ISUD	S
terms	ISUD	S
interpretations	ISUD	S

Під час процедури автентифікації користувач надсилає своє ім'я користувача (username) та введений пароль. У разі успішної автентифікації, користувач автоматично переадресовується на головну сторінку, враховуючи його роль в системі. На головній сторінці забезпечуються відповідні інструменти та можливості, що відповідають завданням цієї ролі.

8 ІНСТРУКЦІЯ КОРИСТУВАЧА

8.1 Адміністратор словника

Після підключення до веб-адреси додатку, користувач бачить сторінку з описом готелю (рисунок 8.1).

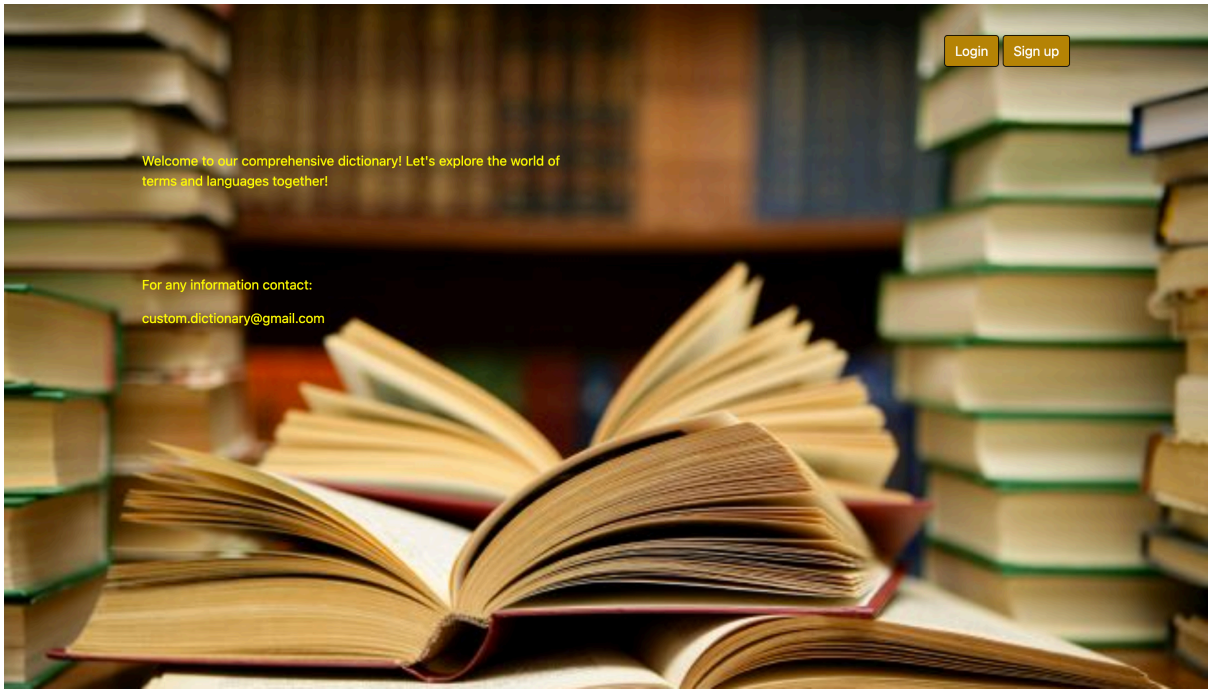

Рисунок 8.1 – Початкова сторінка

Користувач може ввести свої логін та пароль, якщо він має аккаунт, або зареєструватися, якщо користується додатком вперше. Вікна авторизації та реєстрації зображені на рисунках 8.2 та 8.3 відповідно.

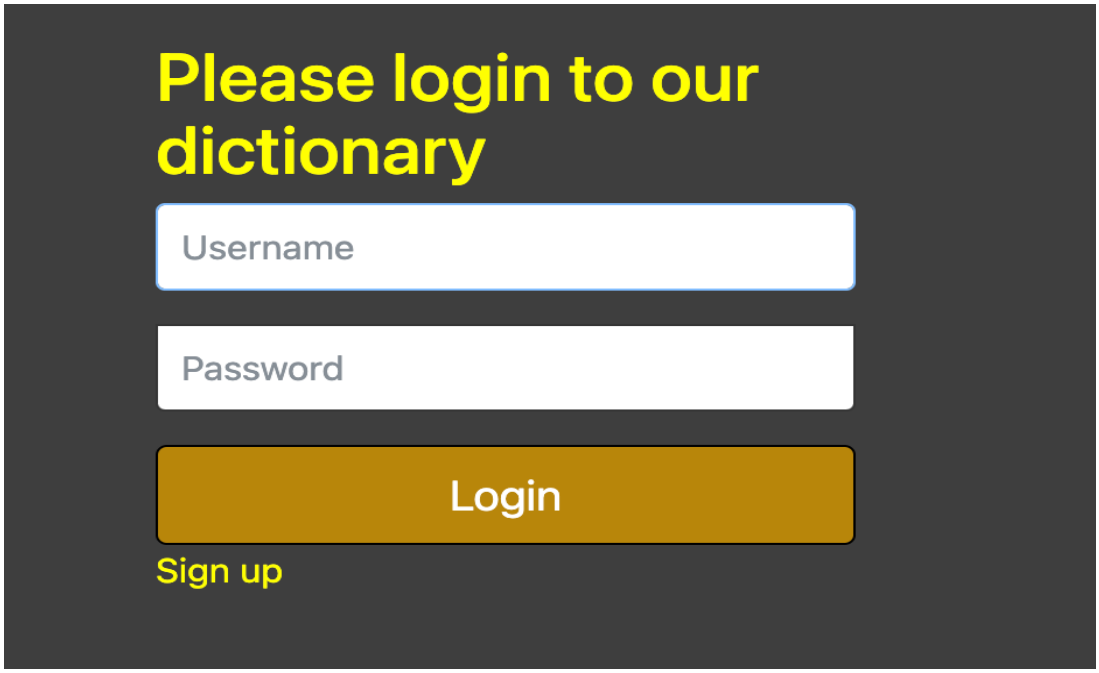

Рисунок 8.2 – Сторінка авторизації

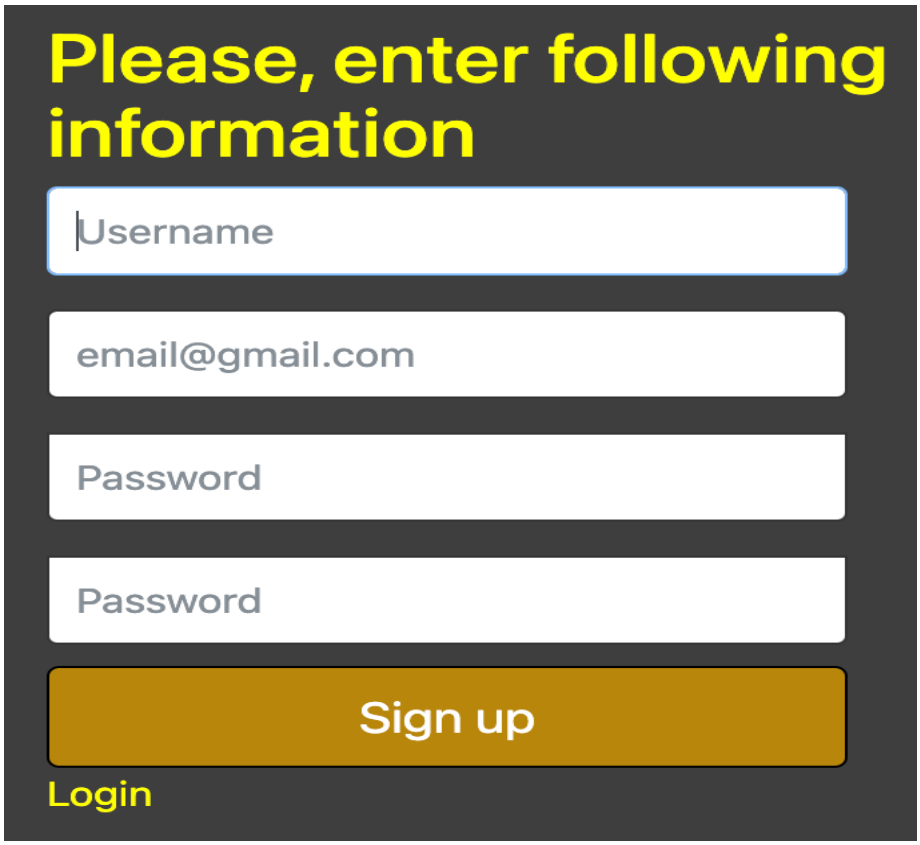

Рисунок 8.3 – Сторінка реєстрації

Якщо користувач авторизувався як адміністратор, його домашня сторінка буде виглядяти наступним чином (рисунок 8.4)

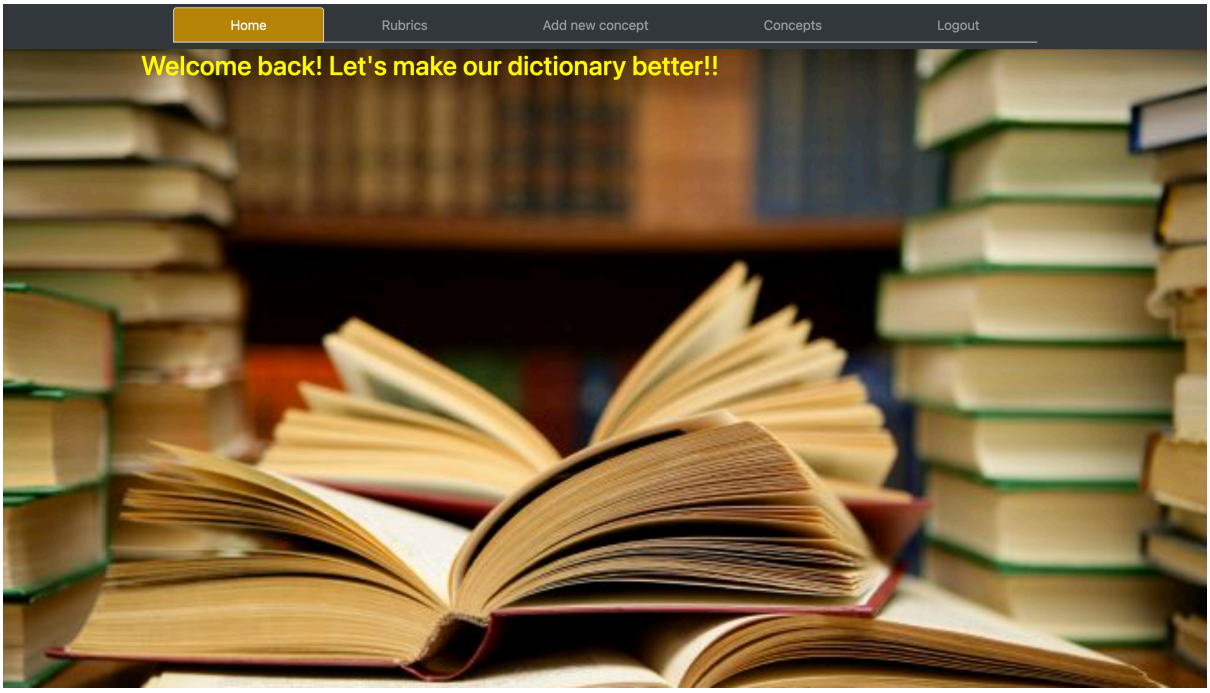

Рисунок 8.4 – домашня сторінка адміністратора

Адміністратор може переглядати, додавати, редагувати та видаляти рубрики (рисунок 8.5 та рисунок 8.6)

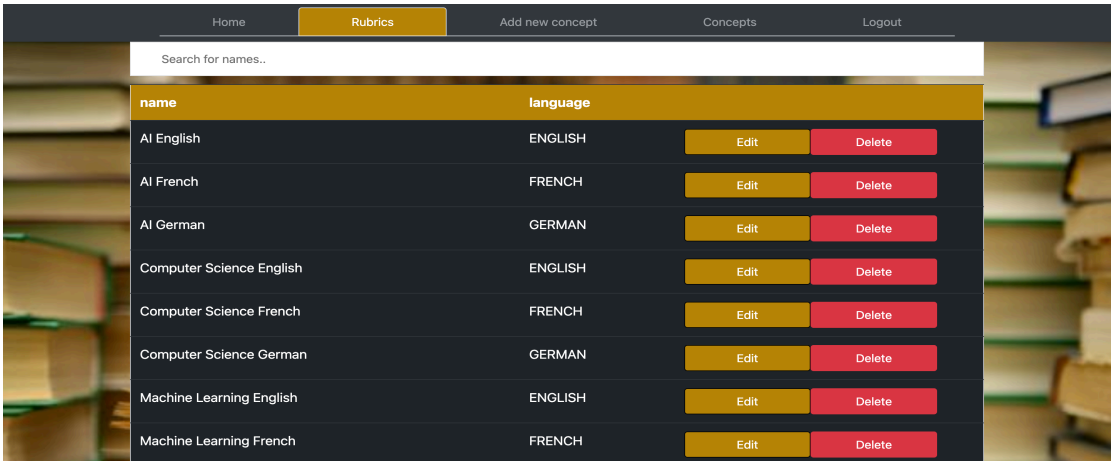

Рисунок 8.5 – сторінка рубрик

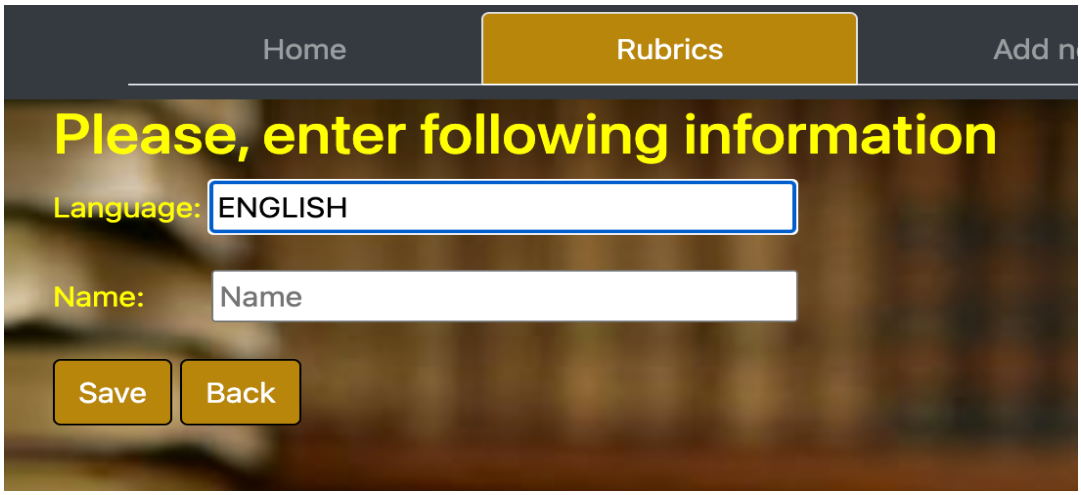

Рисунок 8.6 – додавання нової рубрики

Адміністратор може додавати нові поняття (рисунок 8.7)

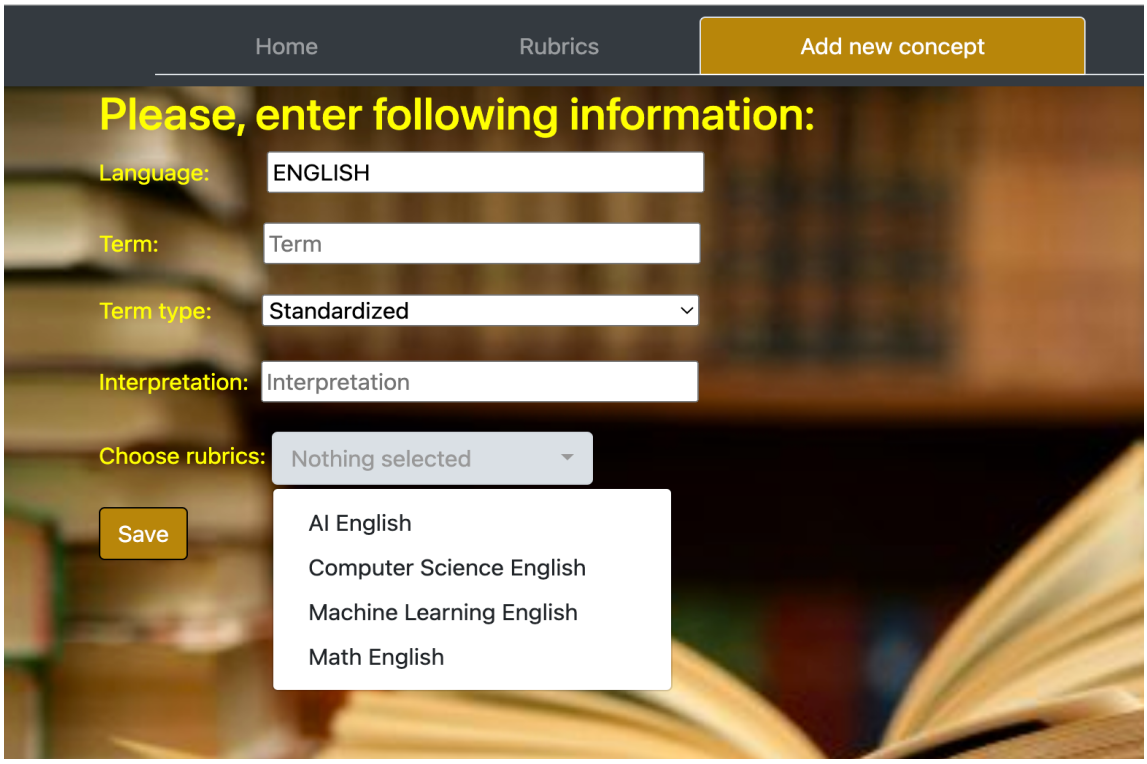

Рисунок 8.7 – додавання нового поняття

Адміністратор може переглядати, редагувати, видаляти поняття (рисунок 8.8)

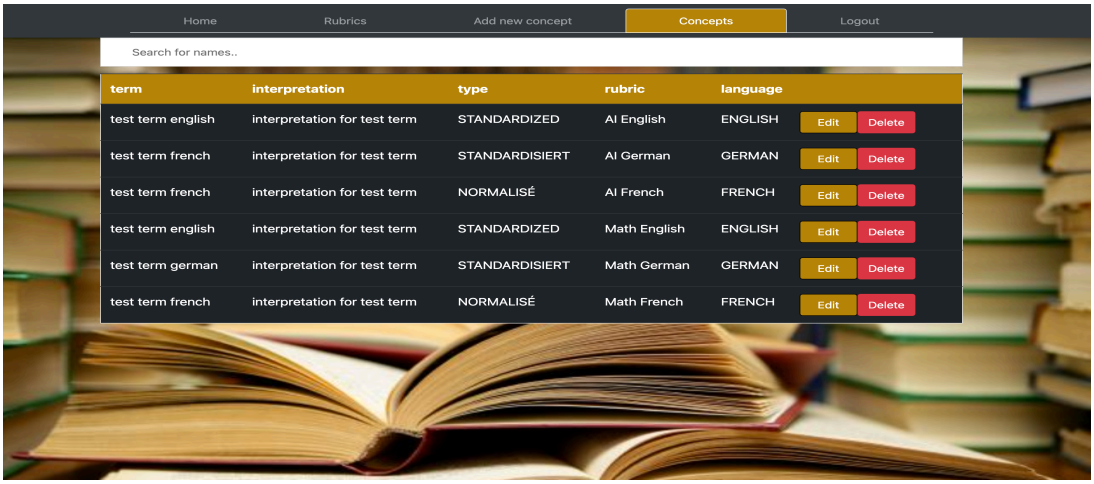

Рисунок 8.8 – сторінка понять для адміністратора

8.2 Відвідувач словника

Якщо користувач авторизувався як відвідувач, його домашня сторінка буде виглядати наступним чином (рисунок 8.9)

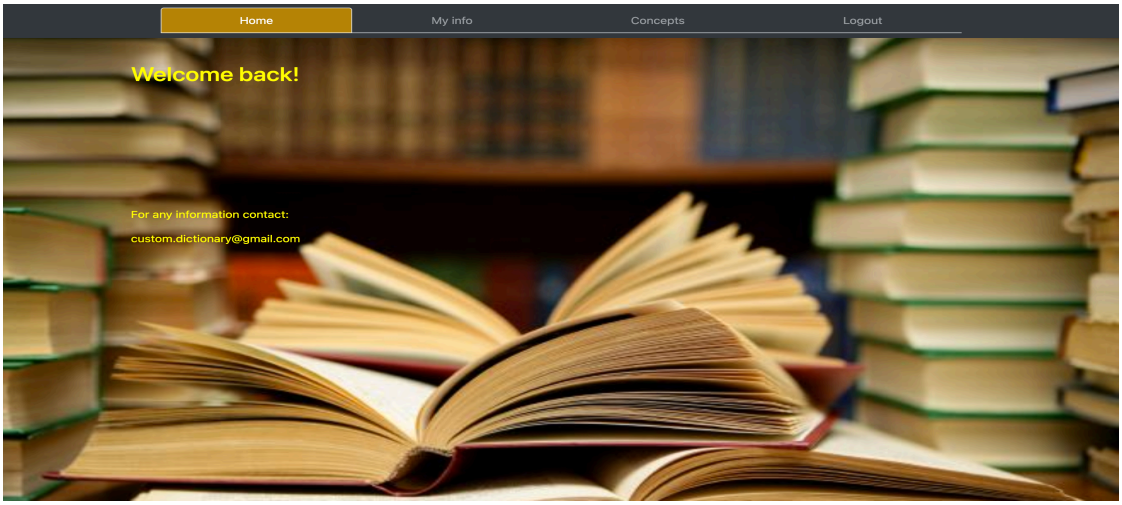

Рисунок 8.9 – сторінка понять для адміністратора

Відвідувач може обирати мову, на якій він бажає отримувати поняття (рисунок 8.10)

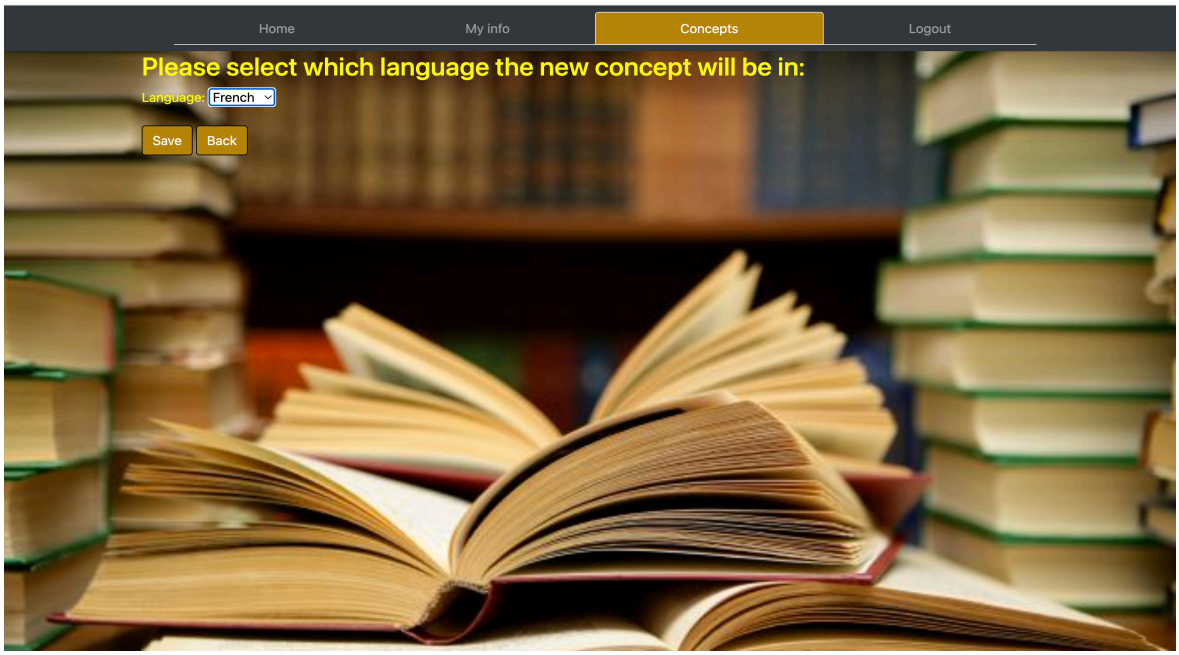

Рисунок 8.10 – вибір мови для понять

Відвідувач може переглядати поняття на обраній мові, фильтруючи їх за рубриками, проводячи пошук по терміну або інтерпретації (8.11)

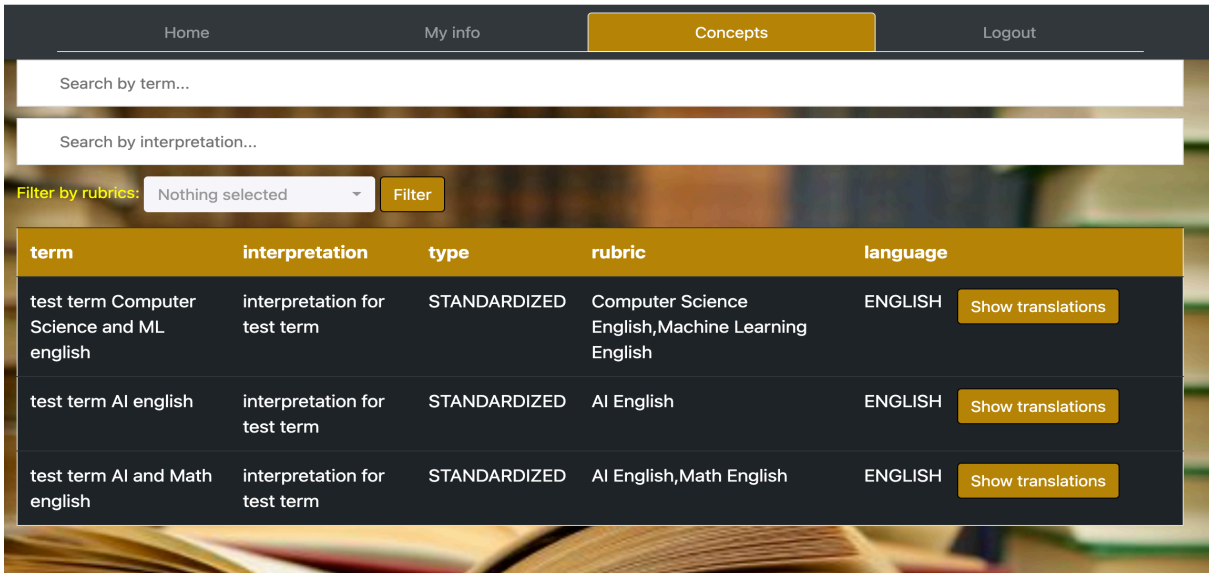

Рисунок 8.11 – поняття на обраній мові

Відвідувач може отримувати переклади обраного слова на всіх мовах, що підтримуються информаційною системою (рисунок 8.12)

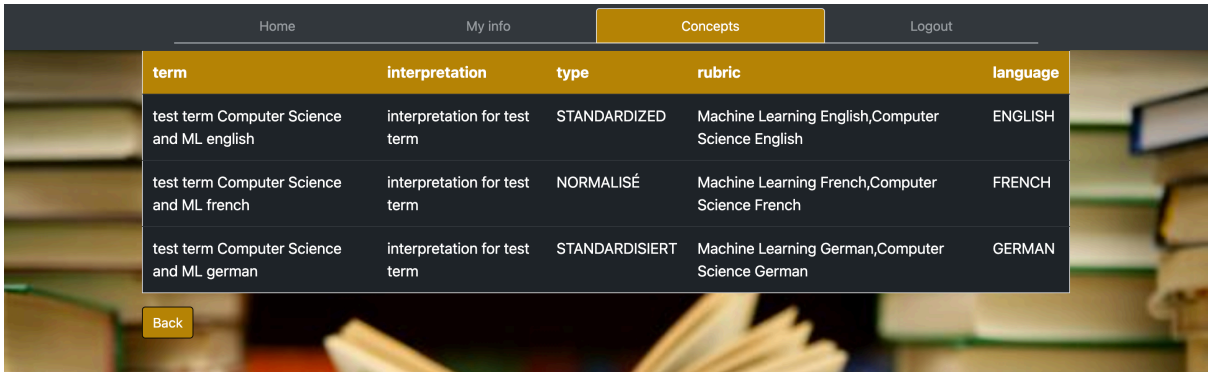

Рисунок 8.11 – переклад обраного слова на мови, що підтримуються інформаційною системою

ВИСНОВКИ

Під час виконання дипломної роботи було проведено аналіз існуючих аналогів багатомовних тлумачних словників, їх переваги та недоліки, обгрунтовано необхідність у розробці своєї системи. Було сформульовано задачі системи і проведено дослідження предметної області. Також було вирішено питання вибору архітектури системи, створено інформаційну модель та проведено аналіз і вибір необхідних технологій для розробки.

В результаті виконання дипломної роботи було спроектовано і розроблено базу даних на основі СУБД PostgreSQL, яка реалізує реляційну модель даних. Також було раздроблено веб-додаток з використанням таких програмних інструментів як мову програмування Kotlin, фреймворк Spring, його шаблонізатор thymeleaf.

У результаті розробки багатомовного тлумачного словника було створено потужний інструмент для користувачів, що дозволяє зручно та ефективно перекладати слова та вирази між різними мовами. Важливим аспектом проекту було створення зручного та інтуїтивно зрозумілого інтерфейсу, що дозволяє користувачам швидко знаходити потрібні слова та отримувати точні тлумачення. Використання сучасних технологій дозволило досягти високої якості тлумачень та швидкого доступу до інформації, що робить проект ефективним та корисним для користувачів. У майбутньому можливо розширення функціональності словника, додавання нових мов та покращення інтерфейсу на основі отриманого досвіду та повідомлень користувачів. Результати цього проекту можуть бути використані як основа для подальших досліджень у галузі машинного перекладу та лінгвістики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Малахов Є.В., Розновець О.І. Проектування інформаційних систем: методичні вказівки до курсового проектування з навчального курсу// Є.В. Малахов, О.І. Розновець / Одеса: Одес. нац. ун-т ім. І.І. Мечникова,2020.
2. PostgreSQL: The world's most advanced open source database [Електронний ресурс] – Режим доступу: http://www.postgresql.org/
3. Spring In Action 5Th Edition by Craig Walls [Електронний ресурс] – Режим доступу: https://www.manning.com/books/spring-in-action-fifth-edition
4. Kotlin In Action by Dmitry Jemerov and Svetlana Isakova [Електронний ресурс] – Режим доступу: https://www.amazon.com/Kotlin-Action-Dmitry-Jemerov/dp/1617293296
5. Thymeleaf with Spring Boot by Michael Good [Електронний ресурс] – Режим доступу: https://pdfcoffee.com/thymeleaf-spring-boot-pdf-free.html
6. Smith, J. (2017). "Multilingual Lexicography: A Perspective". Journal of Lexicography, 42(2), 345-362.

ДОДАТОК А

Ієрархія елементів системи

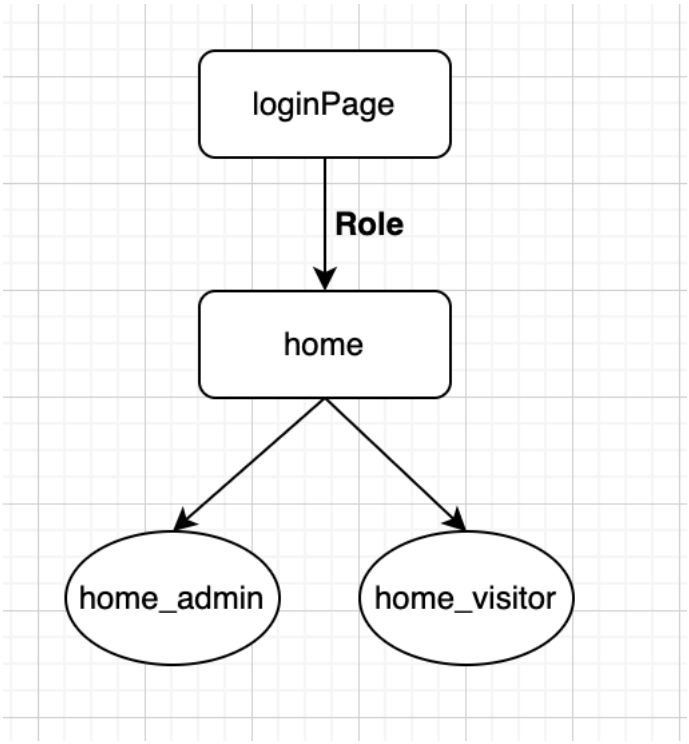

Рисунок А.1 - ієрархія сторінок додатку

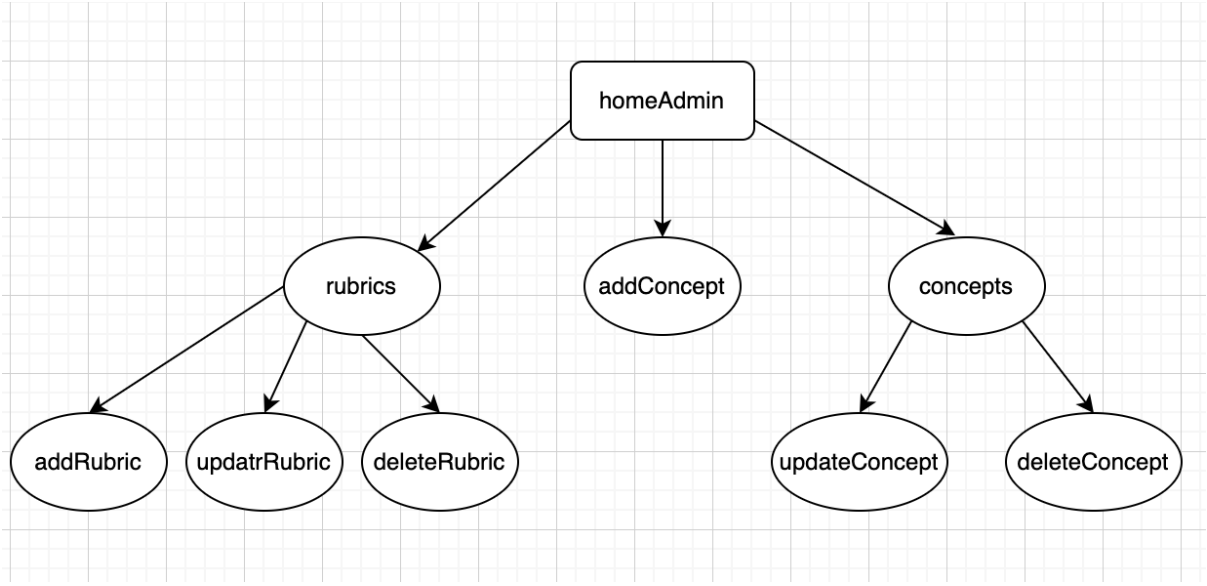

Рисунок А.2 - ієрархія сторінок додатку для адміністратора

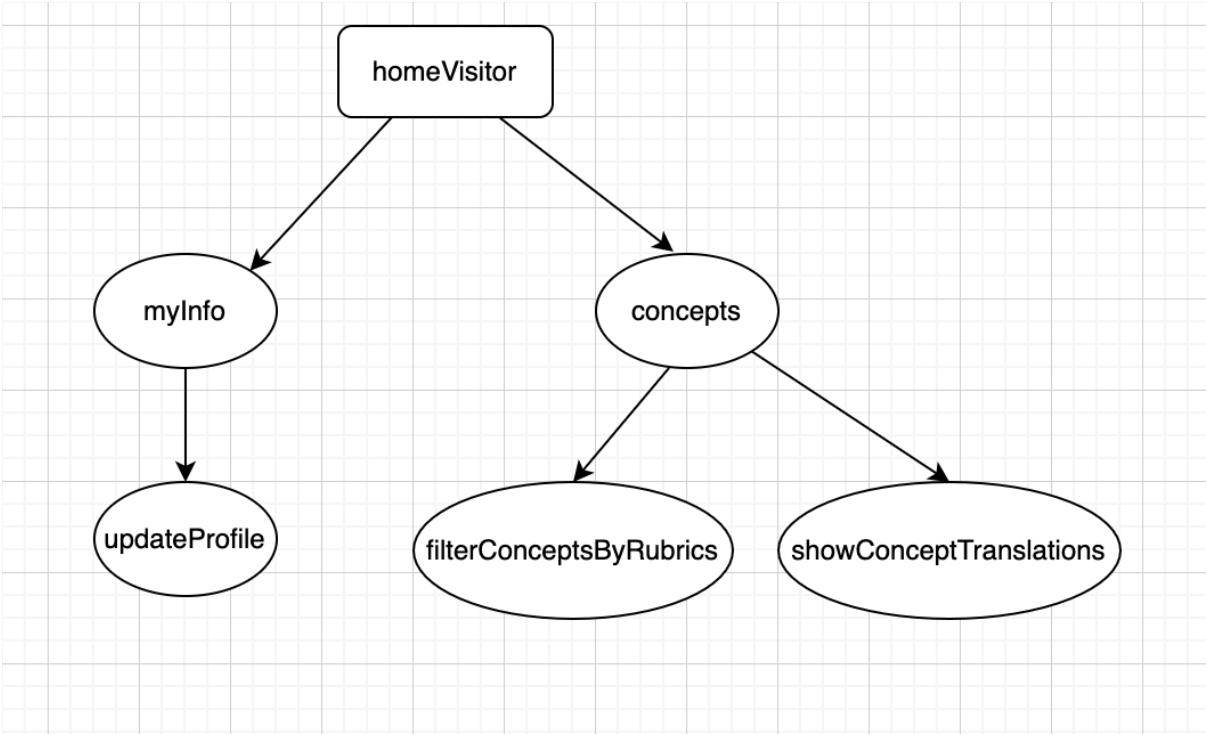

Рисунок А.3 - ієрархія сторінок додатку для відвідувача

ДОДАТОК Б

Запити для створення бази даних

Таблиці та типи:

1) Таблиця users, тип user_role

```
CREATE TYPE user_role AS ENUM ('ADMIN', VISITOR);

CREATE TYPE language_type AS ENUM ('ENGLISH', 'GERMAN',
'FRENCH');

CREATE TABLE users(
   id       uuid PRIMARY KEY   NOT NULL,
   username VARCHAR(50) UNIQUE NOT NULL,
   password VARCHAR(255)       NOT NULL,
   email    VARCHAR(50) UNIQUE NOT NULL
CHECK (email ~ '^.+@[a-z]+\.[a-z]+$'),
   role     user_role          NOT NULL DEFAULT (VISITOR'),
   language language_type
);
```

2) Таблиця rubrics

```
CREATE TABLE rubrics
(
    id       uuid PRIMARY KEY NOT NULL,
    name     VARCHAR(50)      NOT NULL,
    language language_type    NOT NULL
);
```

3) Таблиця concepts

```
CREATE TABLE concepts
(
    id       uuid PRIMARY KEY NOT NULL
);
```

4) Таблиця rubric_reference

```
CREATE TABLE rubric_reference
(
    concept_id   UUID          NOT NULL,
    rubric_id    UUID          NOT NULL,

    PRIMARY KEY (concept_id, rubric_id),

    FOREIGN KEY (concept_id) references concepts (id) ON
UPDATE CASCADE ON DELETE CASCADE,

    FOREIGN KEY (rubric_id) references rubrics (id) ON
UPDATE CASCADE ON DELETE CASCADE
);
```

5) Таблиця terms

```
CREATE TYPE term_type AS ENUM ('STANDARDIZED',
'RECOMMENDED', 'ADDITIONAL', 'DEPRECATED', 'SLANGY');

CREATE TABLE terms
(
    id          uuid PRIMARY KEY NOT NULL,
    name        VARCHAR(50)      NOT NULL,
    type        term_type        NOT NULL,
    language    language_type    NOT NULL,
    concept_id uuid


    FOREIGN KEY (concept_id) REFERENCES concepts (id) ON
UPDATE CASCADE ON DELETE CASCADE
);
```

6) Таблиця interpretations

```
CREATE TABLE interpretations
(
    id       UUID PRIMARY KEY NOT NULL,
    text     VARCHAR(200)     NOT NULL,
    language language_type    NOT NULL,
    concept_id uuid


    FOREIGN KEY (concept_id) REFERENCES concepts (id) ON
UPDATE CASCADE ON DELETE CASCADE
);
```