**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-2»

на тему: Онлайн словник

Виконав : студент \_\_1\_ курсу, групи \_\_\_\_ІА-14\_\_\_\_\_

(шифр групи)

Сергійчук Олександр Андрійович \_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові) (підпис)

Науковий керівник

асистент інформаційних систем та технологій В.О. Міщенко \_\_\_\_\_\_\_\_\_

(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

Сергійчук.О.А. \_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2020

ЗМІСТ

[ВСТУП 3](#_Toc105442119)

[1 ВИМОГИ ДО СИСТЕМИ 4](#_Toc105442120)

[1.1 Функціональні вимоги: 4](#_Toc105442121)

[1.2 Нефункціональні вимоги до системи 4](#_Toc105442122)

[2.1 Діаграма прецедентів 5](#_Toc105442123)

[2.2 Опис сценаріїв використання системи 6](#_Toc105442124)

[3 АРХІТЕКТУРА СИСТЕМИ 10](#_Toc105442125)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 12](#_Toc105442126)

[4.1 Загальна структура проекту 12](#_Toc105442127)

[4.2 Компоненти рівня доступу до даних 13](#_Toc105442128)

[4.3 Компоненти рівня бізнес-логіки 14](#_Toc105442129)

[4.4 Компоненти рівня інтерфейсу користувача 15](#_Toc105442130)

[ВИСНОВКИ 16](#_Toc105442131)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 18](#_Toc105442132)

[ДОДАТКИ 19](#_Toc105442133)

# ВСТУП

У наші часи, коли люди спілкуються багатьма мовами світу, вивчення інших мов дозволяє підтримувати це спілкування. Постає проблема вивчення мов, що займає доволі багато часу, проте для зручності ми можемо використовувати сайти словники, адже ми не завжди маємо можливість носити «великі 10 кілограмові словники» у кармані, просто у своєму телефоні.

Також багато сайтів які мають функцію словника, також мають у собі зручний функціонал перекладача, що дозволяє швидко і якісно переводити фрази, речення, слова з однієї мови на іншу.

Метою роботи є створення сайту, що дозволяє людям виконувати такий перелік дій та вирішувати такі задачі:

* Вибір мови
* Вибір словника.
* Відображення певних слів у словнику
* Відображення перекладу слів з однієї мови на іншу

# 1 ВИМОГИ ДО СИСТЕМИ

## 1.1 Функціональні вимоги:

* Створення, видалення та редагування інформації про мови, словники та переклад
* Вибір мови
* Вибір словника.
* Відображення певних слів у словнику
* Відображення перекладу слів з однієї мови на іншу

## 1.2 Нефункціональні вимоги до системи

* Система має відповідати наступним функціональним вимогам:
* система повинна мати відкриту архітектуру;
* система повинна мати веб-інтерфейс;
* інтерфейс користувача має бути зручним та інтуїтивно-зрозумілим;
* система повинна бути крос-платформною.

2 СЦЕНАРІЙ ВИКОРИСТАННЯ

## 2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами виконання є користувач та адміністратор, взаємодії наведені нижче.

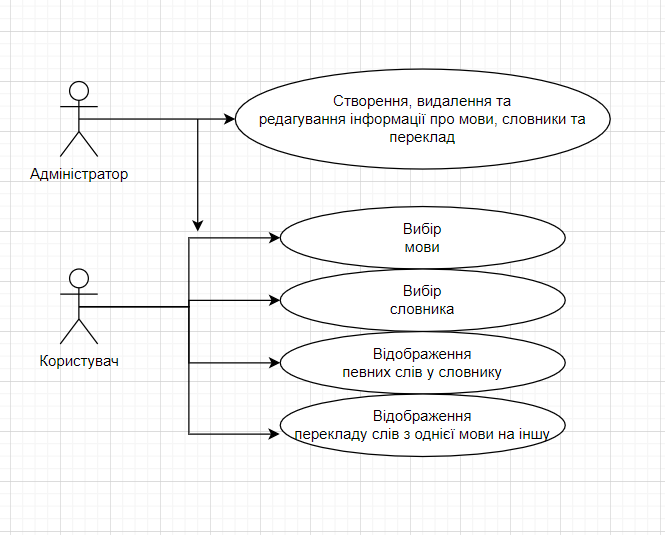


Рис 2.1. Діаграма прецендентів

## 2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.4

Таблиця 2.1 – Сценарій використання «Вхід у аккаунт»

|  |  |
| --- | --- |
| Назва | Вхід в аккаунт |
| ID | 1 |
| Опис | Використовуючи форму для входу у аккаунт взаємодіє з входом для входу відповідно. |
| Актори | Гість |
| Вигоди компанії | Якщо користувач не зайшов у аккаунт він не зможе взаємодіяти з функціоналом сайту |
| Частота користування | Постійно |
| Тригери | Користувач вводить дані у форму входу |
| Передумови | Форма доступна на екрані, якщо в аккаунт не ввійдено |
| Пост-умови | Користувач потрапляє на сторінку де може взаємодіяти зі всім доступним функціоналом |
| Основний розвиток | Гість вводить свої дані для входу у аккаунт та натискає вхід |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

Таблиця 2.2 – Сценарій використання «Додавання перекладу слова»

|  |  |
| --- | --- |
| Назва | Додавання перекладу слова |
| ID | 2 |
| Опис | Адміністратор створює переклад слова |
| Актори | Адміністратор |
| Вигоди компанії | Взаємодія адміністратора для наповнення сайту даними |
| Частота користування | Постійно |
| Тригери | Адміністратор заповняє відповідну форму |
| Передумови | Вхід в аккаунт адміністратора, та у адмін панель |
| Пост-умови | - |
| Основний розвиток | Адміністратор взаємодіє з формою для введення даних щоб додати переклад слова |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

Таблиця 2.3 – Сценарій використання «Відобразити словар за назвою»

|  |  |
| --- | --- |
| Назва | Відобразити словар за назвою |
| ID | 3 |
| Опис | Користувач отримує список друзів |
| Актори | Користувач |
| Вигоди компанії | Взаємодія користувача з словниками |
| Частота користування | Постійно |
| Тригери | Користувач вводить назву словника |
| Передумови | Сторінка доступна при вході в аккаунт |
| Пост-умови | - |
| Основний розвиток | Користувач вводить назву словника та натискає відобрази, отримуючи всі слова з словника |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

Таблиця 2.4 – Сценарій використання «Переклад слів»

|  |  |
| --- | --- |
| Назва | Переклад слів |
| ID | 4 |
| Опис | Користувач отримує переклад слова |
| Актори | Користувач |
| Вигоди компанії | Взаємодія користувача з перекладом |
| Частота користування | Постійно |
| Тригери | Користувач вводить слово, вибирає мову для перекладу |
| Передумови | Сторінка доступна при вході в аккаунт |
| Пост-умови | - |
| Основний розвиток | Користувач вводить слово та вибирає мову для перекладу, натиснувши «Перекласти» отримує переклад |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на Рис 3.1

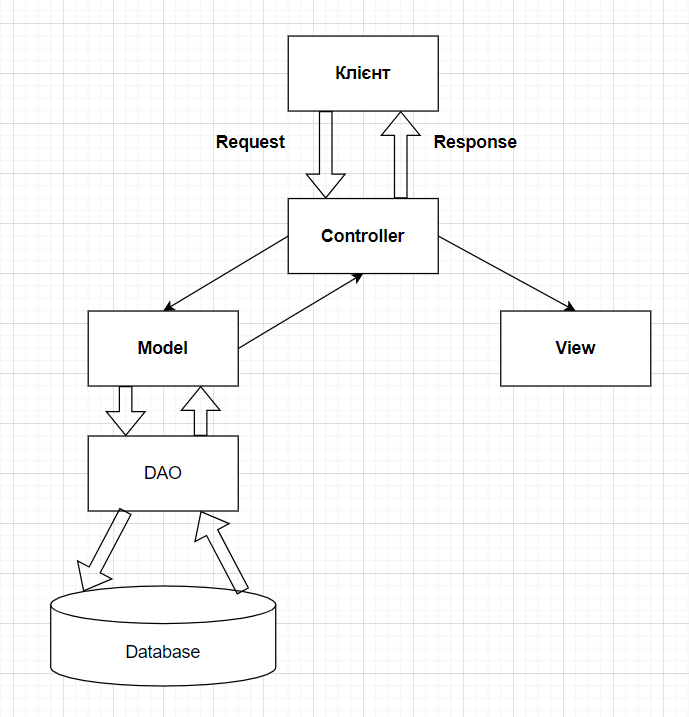


Рис 3.1 Загальна архітектура системи

Система складається з наступних елементів:

- графічний інтерфейс;

- серверна частина;

- база даних.

Графічний інтерфейс необхідний для взаємодії з користувачем. HTTP запит надходить до серверної частини, де оброблюється і повертається відповідь. На серверній частині виконується основна логіка системи. Дані, отриманні з графічного інтерфейсу конвертуються. Також, серверна частина формує запит до бази даних та оброблює відповідь і передає її до графічного інтерфейсу. База даних зберігає дані, які були сформовані на серверній частині та повертає їх у разі запиту. Схема зображена на Рис 3.1.

До серверної частини належать наступні елементи:

- Контролер;

- Модель та вигляд;

-Дао;

- БД.

На контролер надходять дані з графічного інтерфейсу. З контролеру, дані формуються в запит до бази даних і зберігаються. Також в контролері формується вид, тобто об’єкт і його ім’я для відображення на графічному інтерфейсі.

# 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

## 4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

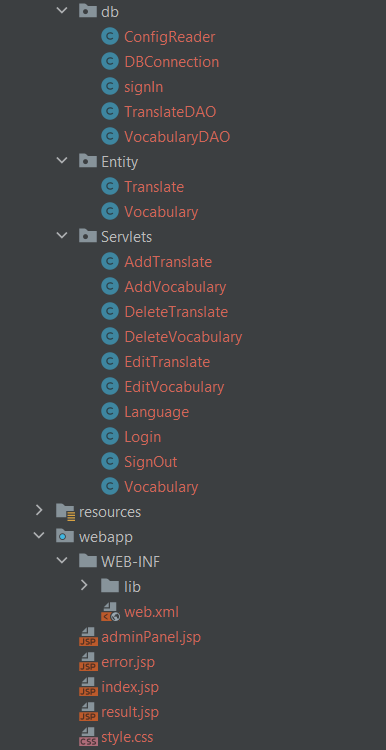


Рисунок 4.1 – Загальна структура проекту

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

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

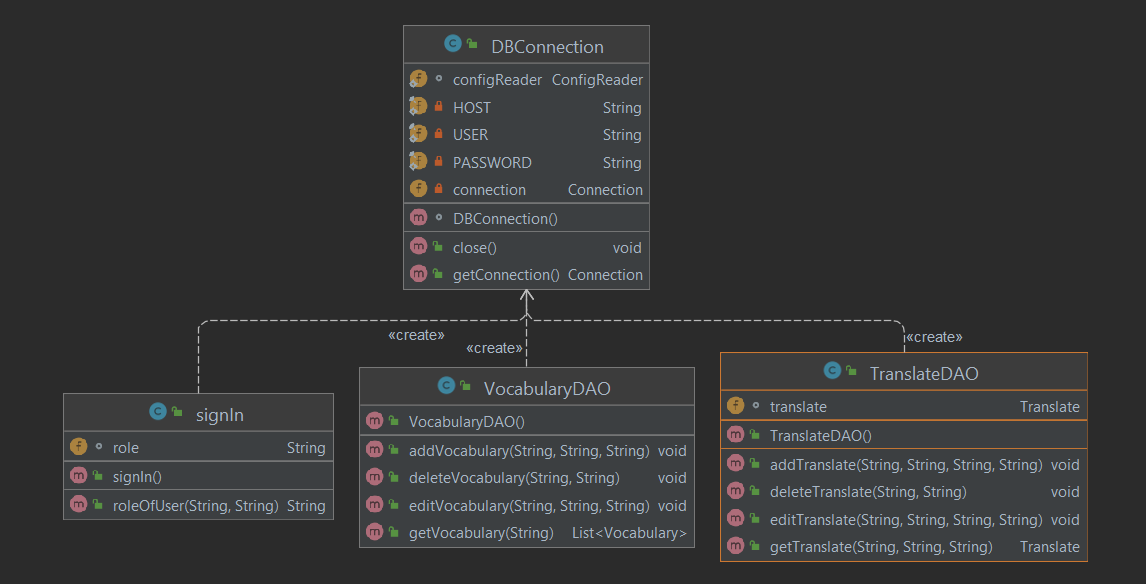
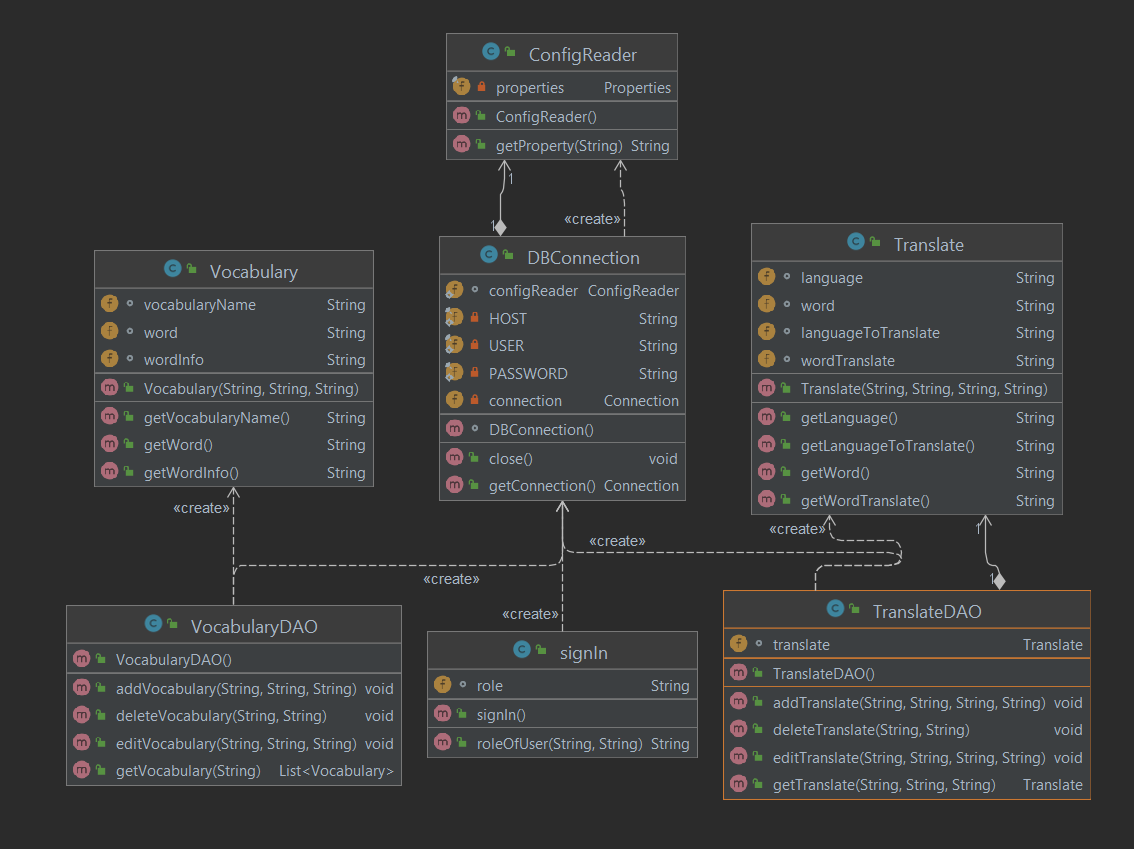
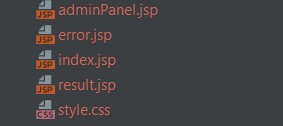


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

## 4.3 Компоненти рівня бізнес-логіки



## 4.4 Компоненти рівня інтерфейсу користувача



JSP + JSTL використовується для виведення результатів для користувача

Також прикріплений файл стилів, що дозволяє користувачу бачити приємніший інтерфейс

# ВИСНОВКИ

Під час виконання роботи було використано ідею google translate, проаналізовано переваги та недоліки цієї ідеї та задумки, виконання основної роботи та проаналізований подальший розвиток проекту.

Є ще багато рішень для зручнішого користування сайтом, тому у подальшому розвитку проекту планується додати автоматичний вибір мови з якої перекладати, розширений словник та інші зручні функції, також проект має ще багато способів для доповнення та покращення.

Сам розвиток був за такими пунктами:

1. Формулювння функціональні та нефункціональні вимоги до системи, що визначило очікувану поведінку системи.
2. Обрано технології на яких буде написана система. Java була обрана в якості мови програмування, середа розробки – IntelliJ IDEA за її можливості та потужність. В якості системи управління базами даних було вирішено використовувати MySQL за легкість її налаштування та підтримки.
3. Обдумані сценарії використання, та можливі нові функції для подальшого збільшення проекту.
4. Розраховуючи загальну архітектуру було вибранно використання шаблону проектування такого як – MVC pattern, через його зручність та безпечність за рахунок розділення логічних рівнів.В моєму випадку таких як Controller(Сервлети для взаємодії користувача з сервером), Model(Для взаємодії з ДАО, що в свою чергу взаємодіє з базою даних), View(Для взаємодії з користувацьким інтерфейсом), та наявністю 2 інтерфейсів, для можливості зручної заміни модуля(DAO).
5. В кінці виконання всіх попередніх пунктів розвитку проекту стало підбирання стилів, для приємного відображення користувацького інтерфейсу. Використовуючи стилі CSS, та підбираючи ідеальні параметри у % для гарного відображення на різних девайсах, та використання max\_width для формування іншого стилю для девайсів що мають мале розширення, щоб користувач бачив нормальний розмір обєктів при малому екрані.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Вся інформація була використана з ресурсу https://metanit.com/java/

Для аналізу ідеї роботи був використаний сайт https://translate.google.com/?hl=uk

# ДОДАТКИ

package db;  
  
import Entity.Translate;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
  
public class TranslateDAO {  
 Translate translate;  
  
 public void addTranslate(String language, String word, String languageToTranslate, String wordTranslate) {  
 String query = "INSERT IGNORE INTO slovnik.wordtranslate(Language, word, LanguageToTranslate, wordTranslate) " +  
 "values (?,?,?,?);";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, language);  
 preparedStatement1.setString(2, word);  
 preparedStatement1.setString(3, languageToTranslate);  
 preparedStatement1.setString(4, wordTranslate);  
 preparedStatement1.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void deleteTranslate(String word, String wordTranslate) {  
 String query = "DELETE FROM slovnik.wordtranslate" +  
 "where word=? and wordTranslate=?;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, word);  
 preparedStatement1.setString(2, wordTranslate);  
 preparedStatement1.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void editTranslate(String word, String wordTranslate, String word1, String wordTranslate1) {  
 String query = "UPDATE wordtranslate set word = ?, wordTranslate=? where word =? and wordTranslate=?;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, word);  
 preparedStatement1.setString(2, wordTranslate);  
 preparedStatement1.setString(3, word1);  
 preparedStatement1.setString(4, wordTranslate1);  
 preparedStatement1.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public Translate getTranslate(String language, String word, String languageToTranslate) {  
 String query = "SELECT wordTranslate FROM slovnik.wordtranslate " +  
 "where Language=? and word=? and LanguageToTranslate=?;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, language);  
 preparedStatement1.setString(2, word);  
 preparedStatement1.setString(3, languageToTranslate);  
 ResultSet set = preparedStatement1.executeQuery();  
 if (set.next()) {  
 String wordTranslate = set.getString(1);  
 translate = new Translate(language, word, languageToTranslate, wordTranslate);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return translate;  
 }  
}

package db;  
  
import Entity.Translate;  
import Entity.Vocabulary;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.util.ArrayList;  
import java.util.List;  
  
public class VocabularyDAO {  
 public void addVocabulary(String vocabularyName, String word, String wordInfo) {  
 String query = "INSERT IGNORE INTO slovnik.vocabulary(VocabularyName, word, WordInfo) " +  
 "values (?,?,?);";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, vocabularyName);  
 preparedStatement1.setString(2, word);  
 preparedStatement1.setString(3, wordInfo);  
 preparedStatement1.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void deleteVocabulary(String vocabularyName, String word) {  
 String query = "DELETE FROM slovnik.vocabulary" +  
 "where VocabularyName=? and word=?;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, vocabularyName);  
 preparedStatement1.setString(2, word);  
 preparedStatement1.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void editVocabulary(String vocabularyName, String word, String wordInfo) {  
 String query = "UPDATE vocabulary set WordInfo = ? where VocabularyName =? and word=?;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, wordInfo);  
 preparedStatement1.setString(2, vocabularyName);  
 preparedStatement1.setString(3, word);  
 preparedStatement1.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public List<Vocabulary> getVocabulary(String vocabularyName) {  
 String query = "SELECT word, WordInfo FROM slovnik.vocabulary where VocabularyName=?;";  
 List<Vocabulary> vocabularyList = new ArrayList<>();  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query);  
 preparedStatement1.setString(1, vocabularyName);  
 ResultSet set = preparedStatement1.executeQuery();  
 while (set.next()){  
 String word = set.getString(1);  
 String wordInfo = set.getString(2);  
 vocabularyList.add(new Vocabulary(vocabularyName, word, wordInfo));  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return vocabularyList;  
 }  
}

package db;  
  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
  
public class signIn {  
 String role;  
  
 public String roleOfUser(String login, String password) {  
 try (DBConnection db = new DBConnection()) {  
 String query = "SELECT role FROM sinoptik.users where login = ? and password = ?;";  
 PreparedStatement preparedStatement = db.getConnection().prepareStatement(query);  
 preparedStatement.setString(1, login);  
 preparedStatement.setString(2, password);  
 ResultSet result = preparedStatement.executeQuery();  
 if (result.next()) {  
 role=result.getString("role");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return role;  
 }  
  
}

package db;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DBConnection implements AutoCloseable{  
 static ConfigReader configReader = new ConfigReader();  
 private static final String HOST = configReader.getProperty("HOST");  
 private static final String USER = configReader.getProperty("USER");  
 private static final String PASSWORD = configReader.getProperty("PASSWORD");  
 private Connection connection = null;  
  
 DBConnection() {  
 try {  
 connection = DriverManager.getConnection(HOST, USER, PASSWORD);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public Connection getConnection() {  
 return connection;  
 }  
  
  
 @Override  
 public void close() throws Exception {  
 connection.close();  
 }  
}

package db;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.Properties;  
  
public class ConfigReader {  
  
 private final Properties properties = new Properties();  
  
 public ConfigReader() {  
 String filePath = "db.properties";  
 try (InputStream out = getClass().getClassLoader().getResourceAsStream(filePath)) {  
 properties.load(out);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 public String getProperty(String propertyName){  
 return properties.getProperty(propertyName);  
 }  
}

package Entity;  
  
public class Translate {  
 String language;  
 String word;  
 String languageToTranslate;  
 String wordTranslate;  
  
 public Translate(String language, String word, String languageToTranslate, String wordTranslate) {  
 this.language = language;  
 this.word = word;  
 this.languageToTranslate = languageToTranslate;  
 this.wordTranslate = wordTranslate;  
 }  
  
 public String getLanguage() {  
 return language;  
 }  
  
 public String getWord() {  
 return word;  
 }  
  
 public String getLanguageToTranslate() {  
 return languageToTranslate;  
 }  
  
 public String getWordTranslate() {  
 return wordTranslate;  
 }  
}

package Entity;  
  
public class Vocabulary {  
 String vocabularyName;  
 String word;  
 String wordInfo;  
  
 public Vocabulary(String vocabularyName, String word, String wordInfo) {  
 this.vocabularyName = vocabularyName;  
 this.word = word;  
 this.wordInfo = wordInfo;  
 }  
  
 public String getVocabularyName() {  
 return vocabularyName;  
 }  
  
 public String getWord() {  
 return word;  
 }  
  
 public String getWordInfo() {  
 return wordInfo;  
 }  
}

package Servlets;  
  
import db.TranslateDAO;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
  
@WebServlet(name = "AddTranslate", value = "/AddTranslate")  
public class AddTranslate extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 if (request.getSession().getAttribute("role").equals("admin")) {  
 String language = request.getParameter("language");  
 String word = request.getParameter("word");  
 String languageToTranslate = request.getParameter("languageToTranslate");  
 String wordTranslate = request.getParameter("wordTranslate");  
 TranslateDAO translateDAO = new TranslateDAO();  
 translateDAO.addTranslate(language, word, languageToTranslate, wordTranslate);  
 request.getRequestDispatcher("adminPanel.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "Not admin, plz login");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import db.TranslateDAO;  
import db.VocabularyDAO;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "AddVocabulary", value = "/AddVocabulary")  
public class AddVocabulary extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 if (request.getSession().getAttribute("role").equals("admin")) {  
 String vocabularyName = request.getParameter("vocabularyName");  
 String word = request.getParameter("word");  
 String wordInfo = request.getParameter("wordInfo");  
 VocabularyDAO vocabularyDAO = new VocabularyDAO();  
 vocabularyDAO.addVocabulary(vocabularyName,word,wordInfo);  
 request.getRequestDispatcher("adminPanel.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "Not admin, plz login");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import db.TranslateDAO;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "EditTranslate", value = "/EditTranslate")  
public class DeleteTranslate extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 if (request.getSession().getAttribute("role").equals("admin")) {  
 String word = request.getParameter("word");  
 String wordTranslate = request.getParameter("wordTranslate");  
 TranslateDAO translateDAO = new TranslateDAO();  
 translateDAO.deleteTranslate(word,wordTranslate);  
 request.getRequestDispatcher("adminPanel.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "Not admin, plz login");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import db.VocabularyDAO;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "DeleteVocabulary", value = "/DeleteVocabulary")  
public class DeleteVocabulary extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 if (request.getSession().getAttribute("role").equals("admin")) {  
 String vocabularyName = request.getParameter("vocabularyName");  
 String word = request.getParameter("word");  
 VocabularyDAO vocabularyDAO = new VocabularyDAO();  
 vocabularyDAO.deleteVocabulary(vocabularyName,word);  
 request.getRequestDispatcher("adminPanel.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "Not admin, plz login");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import db.TranslateDAO;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "EditTranslate", value = "/EditTranslate")  
public class EditTranslate extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 if (request.getSession().getAttribute("role").equals("admin")) {  
 String word = request.getParameter("word");  
 String wordTranslate = request.getParameter("wordTranslate");  
 String word1 = request.getParameter("word1");  
 String wordTranslate1 = request.getParameter("wordTranslate1");  
 TranslateDAO translateDAO = new TranslateDAO();  
 translateDAO.editTranslate(word,wordTranslate,word1,wordTranslate1);  
 request.getRequestDispatcher("adminPanel.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "Not admin, plz login");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import db.VocabularyDAO;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "EditVocabulary", value = "/EditVocabulary")  
public class EditVocabulary extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 if (request.getSession().getAttribute("role").equals("admin")) {  
 String vocabularyName = request.getParameter("vocabularyName");  
 String word = request.getParameter("word");  
 String wordInfo = request.getParameter("wordInfo");  
 VocabularyDAO vocabularyDAO = new VocabularyDAO();  
 vocabularyDAO.editVocabulary(vocabularyName,word,wordInfo);  
 request.getRequestDispatcher("adminPanel.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "Not admin, plz login");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import Entity.Translate;  
import db.TranslateDAO;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
  
@WebServlet(name = "Language", value = "/Language")  
public class Language extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String role = (String) request.getSession().getAttribute("role");  
 if (role != null) {  
 String language = request.getParameter("language");  
 String languageToTranslate = request.getParameter("languageToTranslate");  
 String word = request.getParameter("word");  
 TranslateDAO translateDAO = new TranslateDAO();  
 Translate translate = translateDAO.getTranslate(language,word,languageToTranslate);  
 request.getSession().setAttribute("translate", translate);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "BAD LOGIN");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Servlets;  
  
import db.signIn;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "Login", value = "/Login")  
public class Login extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String login = request.getParameter("login");  
 String password = request.getParameter("password");  
 signIn signIn = new signIn();  
 String role = signIn.roleOfUser(login, password);  
 if (role != null) {  
 request.getSession().setAttribute("role", role);  
 request.getSession().setAttribute("logged", true);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "BAD LOGIN");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
  
 }  
}

package Servlets;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "SignOut", value = "/SignOut")  
public class SignOut extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 request.getSession().setAttribute("logged", false);  
 request.getSession().setAttribute("role", null);  
 request.getRequestDispatcher("/index.jsp").forward(request,response);  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
}

package Servlets;  
  
import Entity.Translate;  
import db.TranslateDAO;  
import db.VocabularyDAO;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
import java.util.List;  
  
@WebServlet(name = "Vocabulary", value = "/Vocabulary")  
public class Vocabulary extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String role = (String) request.getSession().getAttribute("role");  
 if (role != null) {  
 String vocabularyName = request.getParameter("vocabularyName");  
 VocabularyDAO vocabularyDAO = new VocabularyDAO();  
 List<Entity.Vocabulary> vocabularyList = vocabularyDAO.getVocabulary(vocabularyName);  
 request.getSession().setAttribute("vocabularyList", vocabularyList);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 } else {  
 request.getSession().setAttribute("status", "BAD LOGIN");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Admin Panel</title>  
 <link rel="stylesheet" href="stylesheet.css">  
</head>  
<body>  
<**c:choose**>  
 <**c:when** test="${role=='admin'}">  
 <a class="button" href="index.jsp">Home page</a>  
 <div class="form-container">  
 <div>  
 <p>Translate</p>  
 <form action="/AddTranslate" method="post">  
 <label>language</label><br>  
 <input type="text" required name="language"><br>  
 <label>word</label><br>  
 <input type="text" required name="word"><br>  
 <label>language To Translate</label><br>  
 <input type="text" required name="languageToTranslate"><br>  
 <label>word Translate</label><br>  
 <input type="text" required name="wordTranslate"><br>  
 <button type="submit">Add</button>  
 </form>  
 <form action="/EditTranslate" method="post">  
 <label>Word</label><br>  
 <input type="text" required name="word"><br>  
 <label>Word translate</label><br>  
 <input type="text" required name="wordTranslate"><br>  
 <label>New Word</label><br>  
 <input type="text" required name="word1"><br>  
 <label>New Word Translate</label><br>  
 <input type="text" required name="wordTranslate1"><br>  
 <button type="submit">Edit</button>  
 </form>  
 <form action="/DeleteTranslate" method="post">  
 <label>Word</label><br>  
 <input type="text" required name="word"><br>  
 <label>Word translate</label><br>  
 <input type="text" required name="wordTranslate"><br>  
 <button type="submit">Delete</button>  
 </form></div>  
 <div>  
 <p>Vocabulary</p>  
 <form action="/AddVocabulary" method="post">  
 <label>vocabularyName</label><br>  
 <input type="text" required name="vocabularyName"><br>  
 <label>word</label><br>  
 <input type="text" required name="word"><br>  
 <label>wordInfo</label><br>  
 <input type="text" required name="wordInfo"><br>  
 <button type="submit">Add</button>  
 </form>  
 <form action="/EditVocabulary" method="post">  
 <label>vocabularyName</label><br>  
 <input type="text" required name="vocabularyName"><br>  
 <label>word</label><br>  
 <input type="text" required name="word"><br>  
 <label>wordInfo</label><br>  
 <input type="text" required name="wordInfo"><br>  
 <button type="submit">Edit</button>  
 </form>  
 <form action="/DeleteVocabulary" method="post">  
 <label>vocabularyName</label><br>  
 <input type="text" required name="vocabularyName"><br>  
 <label>word</label><br>  
 <input type="text" required name="word"><br>  
 <button type="submit">Delete</button>  
 </form>  
 </div>  
 </div>  
 </**c:when**>  
 <**c:otherwise**>  
 <**jsp:include** page="error.jsp"></**jsp:include**>  
 </**c:otherwise**>  
</**c:choose**>  
</body>  
</html>

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Error</title>  
</head>  
<body>  
<**c:choose**>  
 <**c:when** test="${status!=null}">  
 <p>${status}</p>  
 </**c:when**>  
 <**c:otherwise**>  
 <p>Smth wrong</p>  
 </**c:otherwise**>  
 </**c:choose**>  
${status=null}  
<a href="index.jsp">Click here to go Home Page</a>  
</body>  
</html>

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **page** contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  
<!DOCTYPE html>  
<html>  
<head>  
 <title>Vocabulary</title>  
 <link rel="stylesheet" href="style.css">  
</head>  
<body>  
<**c:choose**>  
<**c:when** test="${logged==true}">  
<script>  
 function showMore() {  
 document.getElementById('vocabularyList').parentElement.classList.add('close');  
 document.getElementById('language').parentElement.classList.remove('close');  
 }  
  
 function showLess() {  
 document.getElementById('language').parentElement.classList.add('close');  
 document.getElementById('vocabularyList').parentElement.classList.remove('close');  
 }  
</script>  
<div class="allForms">  
 <**c:if** test="${role == 'admin'}">  
 <a href="adminPanel.jsp" class="button">Admin panel</a>  
 </**c:if**>  
 <a href="/SignOut" class="button">SignOut</a>  
 <div class="form-container">  
 <form id="vocabularyList" method="post" action="/Vocabulary">  
 <label>Vocabulary Name</label><br>  
 <input type="text" required name="vocabularyName"><br>  
 <button type="submit">Show</button>  
 </form>  
 <button onclick="showMore()">Translate</button>  
 </div>  
 <div class="form-container close">  
 <form id="language" method="post" action="/Language">  
 <label>Enter language</label><br>  
 <input type="text" required name="language"><br>  
 <label>Enter language To translate</label><br>  
 <input type="text" required name="languageToTranslate"><br>  
 <label>Enter word</label><br>  
 <input type="text" required name="word"><br>  
 <button type="submit">Translate</button>  
 <br>  
 </form>  
 <button onclick="showLess()">Vocabulary</button>  
 </div>  
 </**c:when**>  
 <**c:otherwise**>  
 <div class="form-container">  
 <form action="/Login" method="post">  
 <label>Enter Login</label><br>  
 <input name="login" required type="text"><br>  
 <label>Enter password</label><br>  
 <input name="password" required type="password"><br>  
 <button type="submit">Login</button>  
 <br>  
 </form>  
 </div>  
 </**c:otherwise**>  
 </**c:choose**>  
</div>  
<**jsp:include** page="result.jsp"></**jsp:include**>  
</body>  
</html>

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<**c:if** test="${translate!=null}">  
 <div>  
 <p>${translate.language} </p>  
 <p>${translate.languageToTranslate} </p><br>  
 <p>${translate.word} </p>  
 <p>${translate.wordTranslate} </p>  
 </div>  
 ${translate=null}  
</**c:if**>  
<**c:if** test="${vocabularyList!=null}">  
 <div>  
 <**c:forEach** items="${vocabularyList}" var="list">  
 <div>  
 <p>Word :${list.word} </p>  
 <p>Info :${list.wordInfo} </p>  
 </div>  
 </**c:forEach**>  
 </div>  
 ${vocabularyList=null}  
</**c:if**>

div.form-container.close{  
 display: none;  
}  
div.form-container {  
 display: block;  
 width: 100%;  
 text-align: -webkit-center;  
 float: left;  
}  
a.button{  
 border-radius: 25px;  
 border: 4px solid #dcdcdc;  
 display: block;  
 cursor: pointer;  
 color: #666666;  
 padding: 1% 2%;  
 text-decoration: none;  
 background-color: wheat;  
 width: 10%;  
 float: left;  
}  
body{  
 background-color: seashell;  
}  
form{  
 padding: 3%;  
 border-radius: 25px;  
 border: 4px solid #dcdcdc;  
 width: 20%;  
}  
div.allForms{  
 display: block;  
 width: 100%;  
 float: left;  
}