УДК 681.3

АРХІТЕКТУРА ДИНАМІЧНОЇ ВЕБ-СИСТЕМИ ДЛЯ ВИВЧЕННЯ СЛІВ УКРАЇНСЬКОЇ МОВИ

Шевчук Р.П.[1)], Микитюк В.А.[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] магістрант

I. Постановка проблеми

Сучасний етап розвитку методики навчання української мови як іноземної, характеризується використанням різних навчальних і контрольних мультимедійних комп'ютерних програм, дистанційних курсів, веб-систем та електронних словників для інтенсифікації навчального процесу. Суттєвими недоліків відомих веб-ресурсів для вивчення української мови є їх статичність та відсутність чіткого морфологічного підходу до вивчення слів української мови, зокрема відомих словоформ, відмінків, граматичних значень і засобів їх вираження.

Розробка та впровадження динамічної веб-системи для вивчення слів української мови, у якій буде інтегровано словоформи українських слів їх граматичні значення, відмінки та засоби їх вираженнядозволить здійснювати ефективне керування масивами лексикографічних даних.

II. Мета роботи

Метою дослідження є розробка архітектури динамічної веб-системи для вивчення слів української мови як іноземної, яка дозволить інтегрувати на одному ресурсі словоформи українських слів їх граматичні значення, відмінки та засоби їх вираження.

III. Архітектура динамічної веб-системи для вивчення слів української мови

Архітектурою веб-системи можна вважати набір певних структурних компонентів зв'язаних між собою, які задають поведінку всієї системи [1-3].Сьогодні при проектуванні архітектури найчастіше використовується концепція типових рішень, яка базується на виборі з репозиторію архітектур веб-систем, отриманих з практики, найбільш підходящої для реалізації поставлених функціональних вимог до веб-системи.

Практична сторона розробки динамічних веб-додатків, передбачає використання веб-сторінок, з яких користувач переглядає та взаємодіє із сайтом через веб-браузер, що складаються з HTML, Javascript і CSS-файлів, які частково або повністю створені веб-сервером та відправлені в браузер у режимі реального часу [4]. Прикладами технологій, що використовують при розробці динамічних веб-систем є: ASP.NET Core MVC, JavaServerPages, Django.

У роботі було сформовано функціональні та нефункціональні вимоги до веб-системи для вивчення слів української мови. На основі функціональних вимог запропоновано архітектуру веб-системи (рисунок 1).

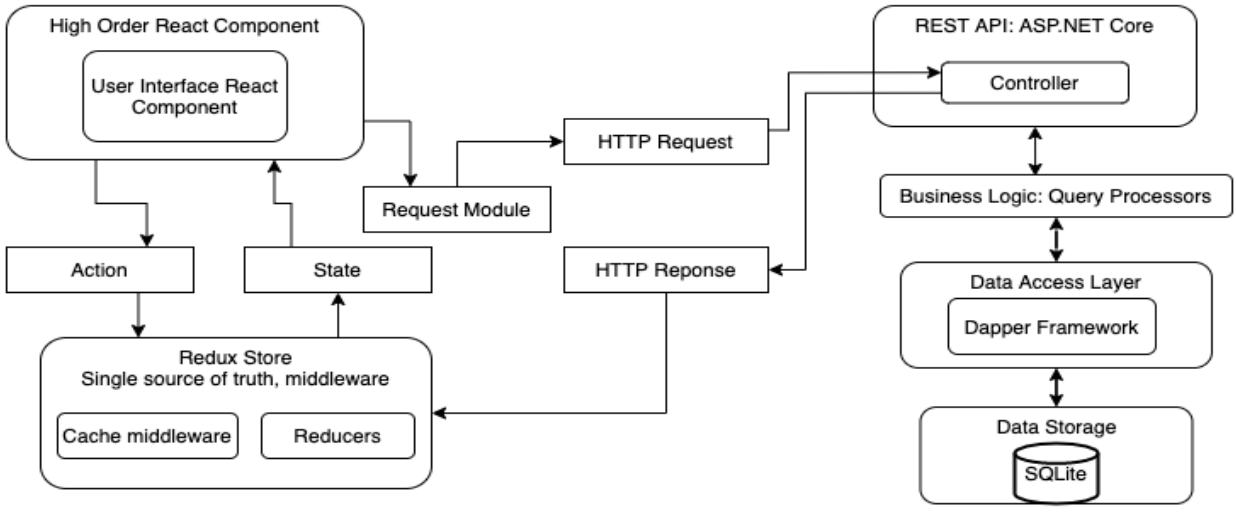

Рисунок 1 –Архітектура веб-системи для вивчення слів української мови

У запропонованій архітектурі веб-системи для вивчення слів української мови (рис.1) використовуєтьсябібліотекаReact на клієнтській стороні та ASP.NET Core WEB API на стороні сервера. Основними складовими запропонованої архітектури є:

1. HigherOrderReactComponent - контейнери ReactJS, які невидимі для користувачів, але містять основні компоненти до яких у користувача буде доступ. Контейнери забезпечують UI компоненти тим станом (State) і діями (Action), які вони можуть виконувати.
2. UserInterfaceReactComponents - елементи веб-інтерфейсу такі як: підписи, текстові вставки, випадаючі списки. Вони отримують параметри, здійснюють доступ до даних, викликаючи RequestModule (тобто ajaxgetfunction), відправляють дію (Action) в ReduxStore для оновлення стану. ReduxStore отримує дію (Action) від UserInterfaceReactComponent, ця дія містять інформацію, яка передається в Reducers, які змінюють дані і повертають новий стан (State). RequestModule здійснює взаємодію з REST API за допомогою HTTP запитів. ASP.NET Core додаток, що виступає в ролі REST API, містить в собі Controller, що приймає HTTP запит та передає обробку до потрібного QueryProcessor (сервісу).
3. QueryProcessors – сервіс у якому інкапсульована вся бізнес-логіка. Щоб отримати дані сервіс звертається в DAL (Data Access Layer), який в свою чергу здійснює доступ до даних за допомогою DapperFramework.
4. Dapper- мікро ORM, якому на вхід подається чистий SQL, а на виході Dapper автоматично перетворює результат вибірки в об’єкт. Відповідь на HTTP запит передається в ReduxStore, який в свою чергу змінює стан (State)[6-8].
5. ASP.NET Core підтримує створення RESTful сервісів, також відомих як веб-API, використовуючи C# [4,8].
6. SQLite - це реляційна база даних, що використовується в різних варіантах використання, таких як: невелике сховище даних, повноцінна база даних з величезною кількістю даних і складними запитами або ж для зберігання файлів для обміну даними [6].

Наведено приклад роботи динамічної веб-системи як електронного словника. На прикладі поданому у таблиці 1 показано особливості формування всіх словоформ обраного слова:

Таблиця 1

Приклад вибірки з таблиці “nom”

reestr	field2	part	type
агра"рний	0	11	2302

Для пошуку кореня слова здійснюються запит до таблиці «indents» словозмінний тип 2302 (таблиця 2).

Таблиця 2

Вибірка з таблиці “indents”. Визначення кореня слова

type	indent
2302	2

Далі необхідно видалили у слова наголос – “аграрний”. Для цього відрізається з кінця слова кількість символів, що дорівнює значенню колонки “indent” – 2, основа – “аграрн”. Робиться вибірка записів з таблиці “flexes”, що мають словозмінний клас (колонка “type”) 2302 (таблиця 3).

Таблиця 3

Результат вибірки закінчень для словоформ з таблиці “flexes”

flex	type
ий	2302
ого	2302
ому	2302
...	2302

У роботі створено даталогічну модель веб-системи для вивчення слів української мови (рисунок 2), яка складається з наступних таблиць:

- tablenom - таблиця реєстрових одиниць словника;
- tableindents - таблиця словозмінних класів;
- tableflexes - таблиця квазіфлексій словозмінних класів;

- tableparts - таблиця частин мови;
- tablegr - таблиця граматичних категорій;
- tableaccents_class - таблиця класів наголосів (акцентуаційних класів);
- tableaccent - таблиця значень наголосів;
- tablealphadigit - таблиця кодування літер;
- tableexamples - приклади вживання слова в контексті.

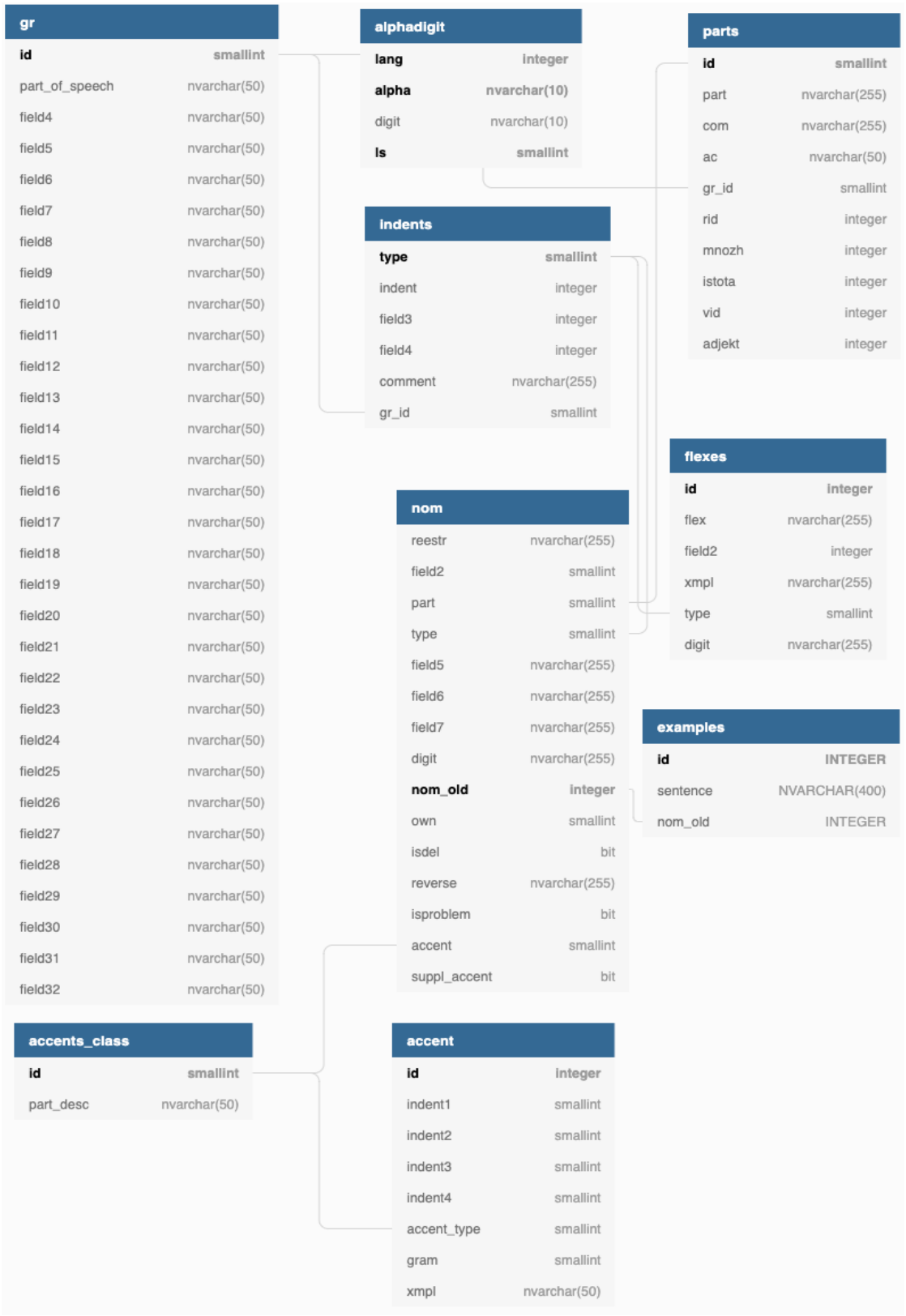

Рисунок 2 –Даталогічна модель бази даних

Для доступу до даних буде використовуватись Dapper ORM –це продукт об’єктно (ORM) для платформи Microsoft .NET, який забезпечує структуру для відображення об'єктно-орієнтованої моделі домену до традиційної реляційної бази даних. Його мета - звільнити розробника від значної частини завдань програмування, пов'язаних із стійкістю реляційних даних [5,6]. Data Access Layer веб-системи буде використовувати Dapper для витягування даних з SQLite бази даних. Для цього на вхід буде подаватись простий SQL запит, а Dapper буде автоматично перетворювати результат вибірки в об’єкт. Опис процесу отримання даних з бази, використовуючи дану технологію, подано на рисунку 3.

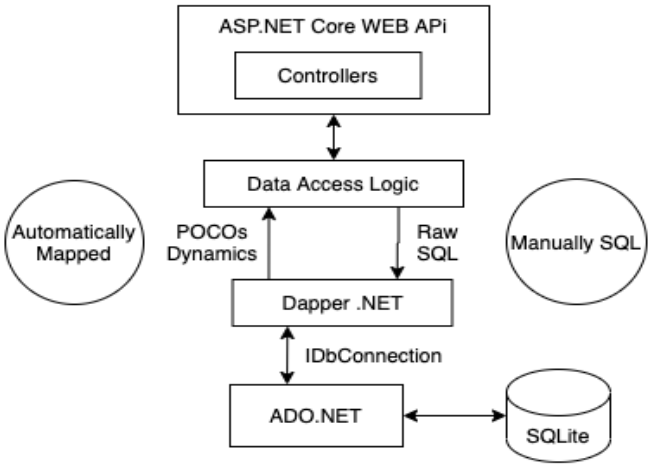

Рисунок 3 –Особливості отримання даних із бази даних

При здійсненні HTTP запиту, веб-API Controller обробляє його і здійснює запит в DAL (Data Access Layer), в якому здійснюються операції по отриманню даних зі сховища. Для взаємодії з базою даних DAL використовує Dapper, якому на вхід подається простий SQL. По-суті, Dapper розширює інтерфейс IDbConnection, який знаходиться в просторі імен System.Data. Всередині Dapper використовує технологію ADO.NET, яка надає доступ до даних, що зберігаються в базі даних або інших джерелах. За допомогою об’єкта Connection, використовуючи відповідний провайдер даних (в нашому випадку Microsoft.Data.Sqlite), відбувається встановлення з’єднання з джерелом даних. Об’єкт Command дозволяє виконувати операції з даними в БД. Об’єкт DataReader зчитує отримані в результаті запиту дані. Далі Dapper автоматично мапить отримані дані в plainold CLR object (POCO), тобто звичайний об’єкт, що використовується в .NET CommonLanguageRuntime (CLR).

Висновок

У роботі запропоновано архітектуру динамічної веб-системи для вивчення слів української мови як іноземної, яка базується на Single-PageApplications та REST API та використовує JavaScript-бібліотеку React та фреймворк ASP.NET Core.

Створено даталогічну модель бази даних веб-системи для вивчення слів української мови та запропоновано алгоритм вибірки даних із бази даних.

Список використаних джерел

1. Фаулер М. Архитектура корпоративных программных приложений / М. Фаулер. – М.: Издательский дом Вильямс, 2006 – 544 с.
2. Шевчук Р.П. Підвищення ефективності клієнт-серверних систем середньої складності / Р.П. Шевчук., А.І. Яцинич // Вісник Тернопільського державного технічного університету. —2010. —Том 15. —No 1. —С. 182—186
3. Мельник А.М. Інформаційна технологія автоматичної генерації тестових завдань з керованою складністю / А.М. Мельник, Р.М. Пасічник, Р.П. Шевчук // Системи обробки інформації. — 2011. — № 3(93). – С. 57-61.
4. Prayaag Kasundra (2020). React Architecture Best Practices and Tips from Community Experts [Online]. Available: https://www.simform.com/react-architecture-best-practices/
5. Chris Woodruff (2018). Advanced Architecture for ASP.NET Core Web API [Online]. Available: https://www.infoq.com/articles/advanced-architecture-aspnet-core/
6. Matthew Jones (2019). Dapper vs EF Core Query Performance Benchmarking [Online]. Available: https://exceptionnotfound.net/dapper-vs-entity-framework-core-query-performance-benchmarking-2019/
7. Kannan Eswar (2020). How to Build CRUD REST APIs with ASP.NET Core 3.1 and Entity Framework Core, Create JWT Tokens, and Secure APIs [Online]. Available: https://www.syncfusion.com/blogs/post/how-to-build-crud-rest-apis-with-asp-net-core-3-1-and-entity-framework-core-create-jwt-tokens-and-secure-apis.aspx
8. Nirjhar Choudhury (2020). How to use SQLite with Dapper (In ASP.NET Core 3.1) [Online]. Available: https://dotnetcorecentral.com/blog/how-to-use-sqlite-with-dapper/
9. A. Melnyk., R. Shevchuk. Transcoding of Formats of Compressed Speech Signals // Proceedings of the 8-th International Conference CADSM’2005. - Lviv-Polyana, Ukraine, 23 - 26 February 2005, P. 151-153.