ПРИКЛАДНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК,
БЕЗПЕКИ ТА МАТЕМАТИКИ, 2023, № 1

УДК 004.9

ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ДОКУМЕНТООБІГУ ФАКУЛЬТЕТУ ЗАКЛАДУ ВИЩОЇ ОСВІТИ

Гордійчук Г.П. (ORCID: 0009-0005-8518-3399),
Булатецька Л.В. (ORCID: 0000-0002-7202-826X),
Булатецький В.В. (ORCID: 0000-0002-9883-4550).
Волинський національний університет імені Лесі Українки, Луцьк, Україна

DESIGN OF A WEB-ORIENTED DOCUMENT MANAGEMENT SYSTEM OF THE FACULTY OF THE INSTITUTION OF HIGHER EDUCATION

Hordiichuk H.P., Bulatetska L.V., Bulatetskyi V.V.
Lesya Ukrainka Volyn National University, Lutsk, Ukraine

Abstract. The paper examines the main requirements and necessary functionality for a web-based electronic document flow management system for the faculty of a higher education institution. A software development concept and methods have been chosen. The principles of building Server Side Applications (SSA) and Application Programming Interfaces (API) have been studied. The development principles of the database and data structuring for ensuring fast access to information, as well as the integration possibilities of Telegram messenger bots to provide quick access to documents, have been considered. The paper proposes a set of roles and permissions for creating, viewing, editing, deleting, and uploading electronic documents, and a basic user business schema has been selected.

Keywords: web based document management system, user role, database, software architecture, Telegram bot.

Вступ. Перевагами використання системи електронного документообігу у закладах вищої освіти (ЗВО) є збільшення швидкості доступу до необхідних документів, можливість швидко передавати або отримувати документи та можливість створення резервних копій документів. Пришвидшення оформлення документів може забезпечити більш ефективну взаємодію учасників освітнього процесу, а отже збільшить ефективність навчання в цілому. Тому впровадження системи електронного документообігу у ЗВО, зокрема на факультетах, є одним із актуальних завдань у сфері освіти. При вирішенні завдання впровадження системи електронного документообігу у ЗВО, або на окремому факультеті, постає питання використання вже готових платформ, або розробки власного програмного продукту. Основною перевагою розробки власної платформи документообігу є можливість навчальному закладу підлаштувати функціонал системи під свої потреби, обрати оптимальну кількість операцій, котрі будуть зрозумілими працівникам і не потребуватимуть спеціального навчання [1-3].

Метою дослідження є розробка веборієнтованої системи електронного документообігу факультету закладу вищої освіти.

Основною вимогою до системи електронного документообігу факультету ЗВО є полегшення та автоматизація подачі та опрацювання документів на факультеті в зручному, швидкому та інтуїтивно зрозумілому користувачеві інтерфейсі з використанням вебтехнологій. У процесі дослідження процесів руху документів на факультеті, починаючи з моменту їх створення або одержання, до завершення виконання або відправлення, було виділено необхідний функціонал системи:

- побудова та розробка бази даних для збереження даних сервісу;
- розробка серверної частини;
- інтегрування Telegram bot;
- розробка інтерфейсу користувача;

- налаштування вебсервера для роботи сервісу;
- розгортання сервісу на вебсервері;
- налаштування домену;
- налаштування автоматичного оновлення сервісу на сервері після зміни вихідного коду у репозиторії (CI/CD).

Дане програмне забезпечення повинно автоматизувати та спростити процес документообігу всередині факультету ЗВО, надавши користувачам можливість швидкого створення, редагування та завантаження своїх документів або ж документів інших користувачів, при цьому забезпечивши конфіденційність та цілісність цих даних.

Даний програмний продукт повинен відповідати наступним функціональним вимогам: система повинна розподіляти можливості користувачів на основі їх ролей, надавати користувачу підказки під час помилок у форматуванні вказаних даних та доступ тільки до дозволених методів API, забезпечувати можливість навігації тільки по дозволених користувачу вебсторінках, зберігати електронні документи на сервері у шифрованому вигляді, реалізовувати API для роботи із Telegram ботом; таблиці в системі повинні мати пагінацію, пошук та сортування; користувач повинен мати можливість оперувати деталями свого облікового запису (зокрема, мати можливість виконати реєстрацію нового облікового запису, аутентифікуватись у системі та здійснити вихід із неї, відновити та змінити пароль від свого облікового запису), мати можливість керувати мовою інтерфейсу, мати можливість оперувати власними електронними документами, включно з роботою через Telegram бот.

Важливою частиною проектування та розробки системи документообігу є організація ролей та дозволів, для створення, перегляду, редагування, видалення або ж завантаження електронного документа [4]. Під час дослідження була обрана така базова бізнес схема користувачів:

- ***Супер адміністратор*** – користувач найвищого рівня (цього користувача неможливо видалити із системи; профіль використовується для адміністрування мережі документообігу, надання дозволів адміністратора; цей користувач має можливість перегляду, завантаження та видалення документа будь-якого користувача);
- ***Адміністратор*** – користувач із підвищеними привілеями (ця роль може бути надана користувачеві із іншою роллю, для того щоб мати можливість отримати доступ до документів інших учасників процесу із аналогічною або роллю нижчого рівня);
- ***Викладач*** – користувач високого рівня (ця роль дає можливість отримати доступ до документів користувачів із роллю нижчого рівня);
- ***Працівник*** – користувач із аналогічними можливостями що і викладач;
- ***Студент*** – користувач який має можливість завантаження, перегляду, редагування або ж видалення документів власного профілю.

Також при необхідності будь-який учасник має можливість надавати доступ до свого документа іншим користувачам.

При такій схемі організації дозволів, система електронного документообігу зможе забезпечити ефективне функціонування закладу освіти, користувачі матимуть можливість у повному обсязі виконувати необхідну роботу більш автоматизовано, що призведе до розвантаження їх робочого часу та збільшить продуктивність усієї організації.

Можливості та дозволи користувачів подано в табл.1.

Таблиця 1. Можливості та дозволи ролей користувачів

Ролі	Користувач	Адміністратор	Супер адміністратор
Створення, редагування, завантаження, видалення власного документа	+	+	+
Генерування власного документа за шаблоном	+	+	+
Перегляд, завантаження, редагування документів інших користувачів	-	+	+
Видалення документів інших користувачів	-	-	-
Перегляд, завантаження, редагування, видалення доступних шаблонів	-	+	+
Перегляд списку користувачів	-	+	+
Редагування профілю іншого користувача	-	+	+
Редагування власного профілю	+	+	+
Деактивація облікового запису іншого користувача	-	+	+
Надання ролі адміністратора	-	-	+
Перегляд доступного списку ролей	-	+	+
Редагування ролі (окрім ролі адміністратора та супер адміністратора)	-	+	+
Створення нової ролі	-	+	+
Видалення ролей (окрім ролі адміністратора та супер адміністратора)	-	+	+
Перегляд, створення, редагування, видалення доступних категорій документів	-	+	+
Перегляд списку адміністраторів	-	-	+
Деактивація ролі адміністратора для користувача	-	-	+

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

- використання браузера для доступу до вебсервісу;
- використання домену для доступу до вебсервісу;
- кросбраузерність;
- адаптивність;
- здійснення захищеного SSL з'єднання із браузером користувача за допомогою SSL сертифіката;
- використання інструментів перевірки та валідування коду під час розробки додатку;
- реляційна база даних;
- використання програмного забезпечення Git для контролю версій та платформи Gitlab для зберігання коду;
- використання програмного забезпечення для автоматичного оновлення поточної версії додатку на сервері під час зміни додатку у системі контролю версій (CI/CD);
- наявність документації до кінцевих точок API серверної частини та приклади даних для роботи із кожною із них;
- можливість робити резервне копіювання даних, які зберігаються на сервері;
- можливість перегляду статистики навантаження сервера.

Методологія дослідження. Розробка програмного засобу велась на основі ітеративної моделі із використанням методології Scrum.

Ітеративна модель розробки програмного забезпечення основана на його розробці через окремі випуски. Кожен випуск являє собою розроблену частину функціоналу. Після кожного з випусків виділяється наступна частина функціоналу, яку необхідно

впровадити. Випуск завершується через кінцевий варіант розробленого функціоналу та має фіксований час на розробку. Фази ітеративної моделі розробки:

- збір та аналіз вимог;
- проектування інтерфейсів та архітектури функціоналу;
- реалізація функціоналу;
- тестування;
- розгортання;
- огляд функціоналу на розгорнутому проекті.

Результати дослідження та їхнє обговорення.

Під час розробки архітектури проєкту її було розділено на три частини: база даних, бек-енд та фронт-енд.

Структура бази даних складається із 6 таблиць (рис. 1) [5]:

- ***Users*** – записи про зареєстрованих користувачі системи та інформація їх облікового запису;
- ***Fales*** – записи про наявні електронні документи користувачів;
- ***Templates*** – записи про наявні шаблони для електронних документів;
- ***Categories*** – записи наявних категорій електронних документів;
- ***Roles*** – записи наявних ролей користувачів;
- ***RegisterRoles*** – записи зв'язків користувача та його ролей.

Бек-енд та фронт-енд розроблені на основі клієнт-серверної архітектури, що дозволить працювати багатьом користувачам у сервісі одночасно і мати завжди актуальні дані про усіх користувачів. Комунікація між цими частинами виконується через інтернет з'єднання.

Архітектура серверної частини поділена на три рівні:

- рівень доступу до даних;
- рівень бізнес логіки;
- презентаційний рівень.

Рівень доступу до даних виконує роль посередника між бізнес логікою та базою даних. Він займається конвертуванням моделей у необхідний для бази даних вигляд і навпаки. Сутності, які там описані реалізовують низько рівневі функції для оперування інформацією. Архітектура з'єднання рівня доступу до даних та бази даних наведена на рис. 2.

Рівень бізнес логіки описує основний функціонал серверної частини. Він приймає дані, обробляє їх та передає на рівень нижче для збереження стану даних у базу даних. Він складається із окремих сервісів, деякі з них мають посилання на інші потрібні їм сервіси та контролери доступу до даних. Кожен сервіс має свою зону відповідальності за певну обробку інформації, виконавши свою функціональність він повинен віддати об'єкти іншим сутностям для подальшого опрацювання.

Розглянемо архітектуру основних сервісів бізнес логіки. Основним функціоналом даного програмного забезпечення є зберігання та надання доступу до електронних документів користувача, тому головним сервісом серверної частини являється сервіс файлів. Його архітектура розроблена на основі інтерфейсу IFileService (рис. 3).

Сервіс файлів реалізовує базові функції для збереження, редагування, витягнення та видалення електронних документів. Його реалізація залежить від сервісів які будуть реалізовувати інтерфейси ITemplateService та IConfiguration.

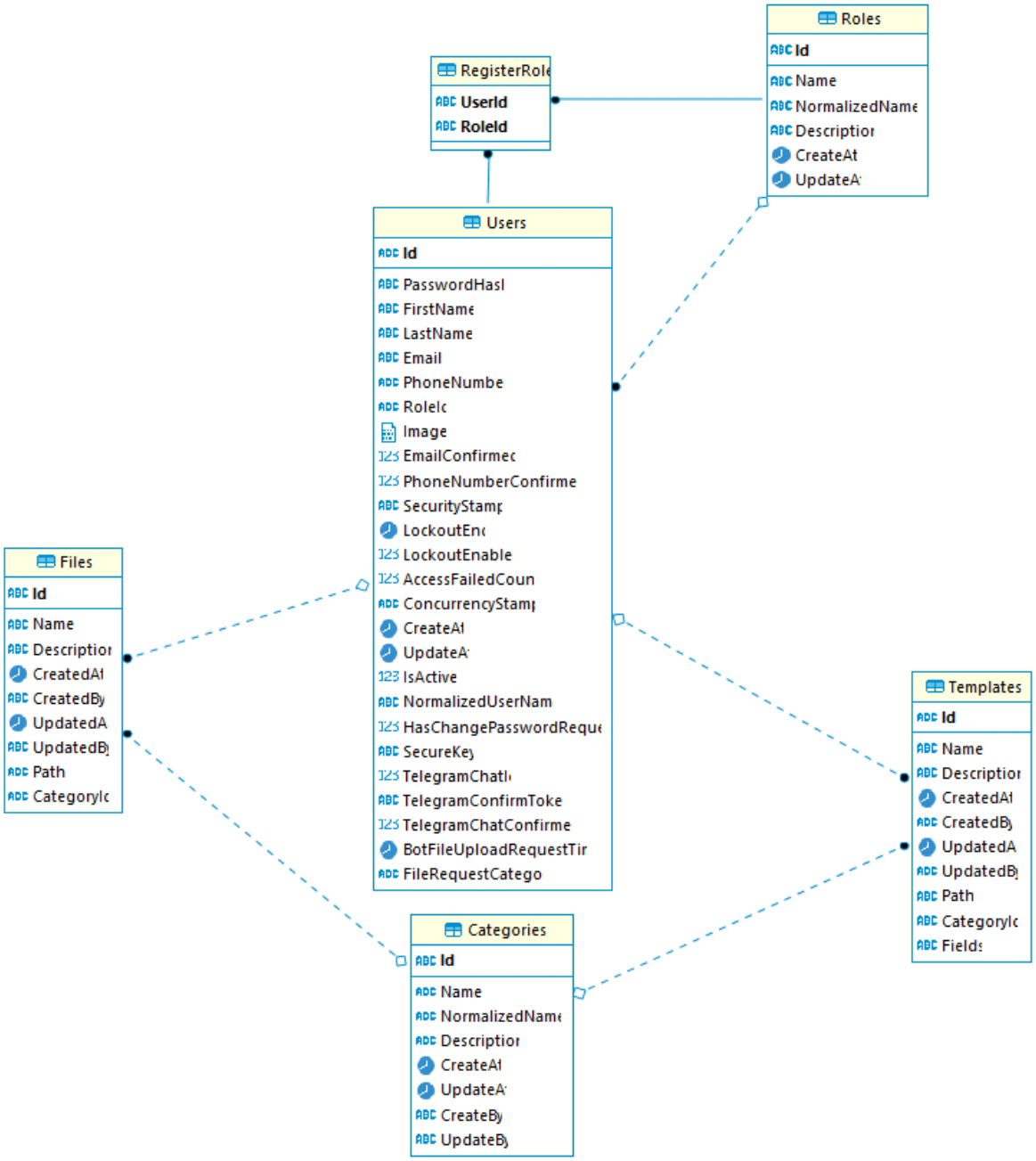

Рис. 1. Структура бази даних

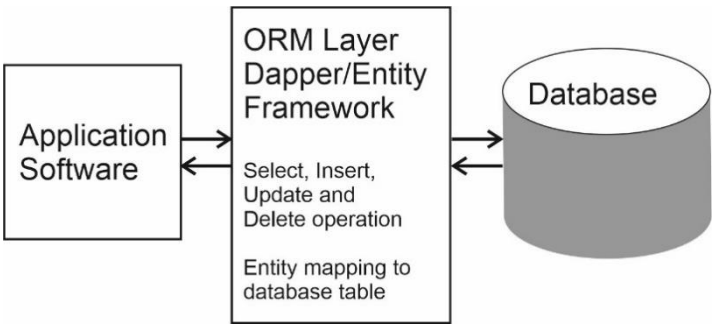

Рис. 2. Архітектура зв'язку рівня доступу до даних та бази даних

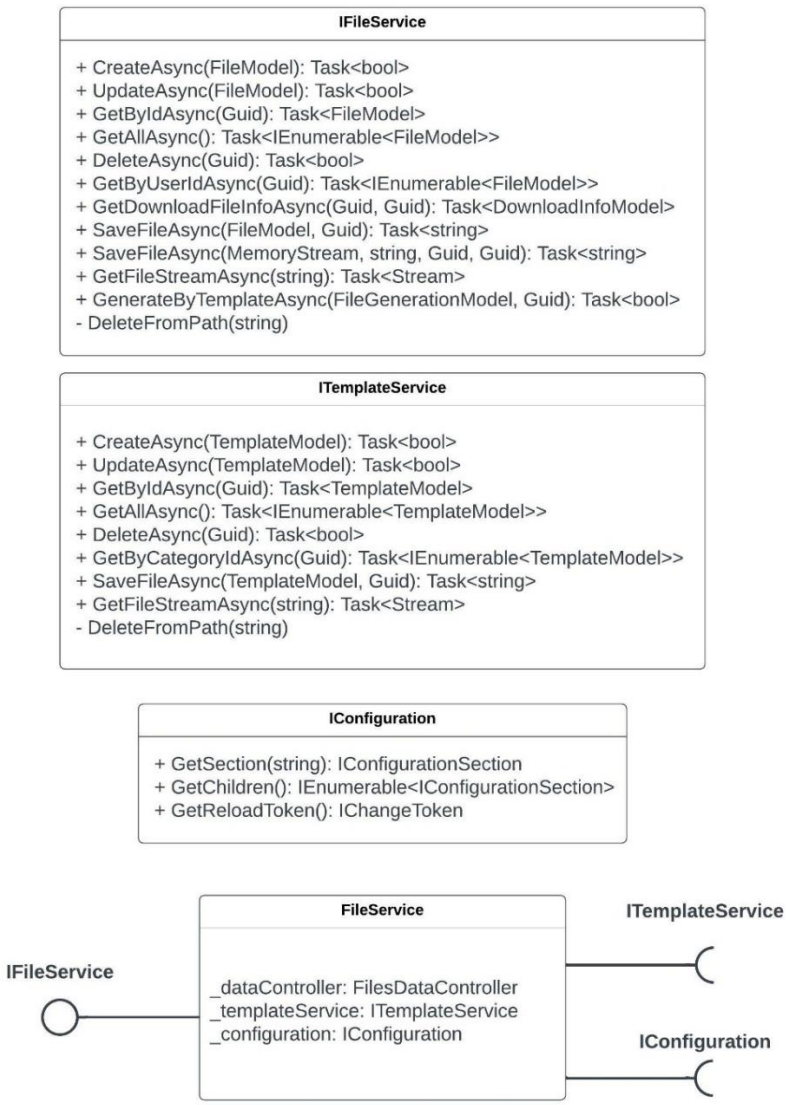

Рис. 3. Архітектура сервісу файлів

Архітектура презентаційного рівня побудована на основі архітектурного стилю REST. REST API (також відомий як RESTful API) – це інтерфейс прикладного програмування (API або веб-API), який відповідає обмеженням архітектурного стилю REST і дозволяє взаємодіяти з вебслужбами RESTful. REST розшифровується як репрезентативна передача стану і був створений комп'ютерним науковцем Роєм Філдінгом. API – це набір визначень і протоколів для створення та інтеграції прикладного програмного забезпечення.

Принципи REST архітектури:

- сервер повинен бути окремою частиною системи і не повинен бути пов'язаний із інтерфейсом користувача, який теж не повинен бути пов'язаним із сервером;
- сервер не повинен зберігати інформацію про своїх клієнтів;
- сервер повинен мати визначений інтерфейс для взаємодії клієнтів із ним;
- розподіл сервера на окремі шари, кожен шар повинен спілкуватись тільки із наступним шаром.

Коли клієнтський запит виконується через RESTful API, він передає представлення стану ресурсу запитувачу або кінцевій точці. Ця інформація або представлення надається в одному з кількох форматів через HTTP: JSON (нотація

об'єктів Javascript), HTML, XLT, Python, PHP або звичайний текст. JSON є найпопулярнішим форматом файлів для використання, оскільки він не залежить від мови, а також читається як людьми, так і машинами.

Заголовки та параметри також важливі в методах HTTP HTTP-запиту RESTful API, оскільки вони містять важливу ідентифікаційну інформацію щодо метаданих запиту, авторизації, уніфікованого ідентифікатора ресурсу (URI), кешування, файлів cookie та більше. Є заголовки запиту та заголовки відповіді, кожен із власною інформацією про з'єднання HTTP та кодами стану.

Протокол передачі гіпертексту (HTTP) є основою всесвітньої павутини та використовується для завантаження вебсторінок за допомогою гіпертекстових посилань. HTTP — це протокол прикладного рівня, призначений для передачі інформації між мережевими пристроями та працює поверх інших рівнів стеку мережевих протоколів. Типовий потік через HTTP передбачає, що клієнтська машина надсилає запит серверу, який потім надсилає повідомлення у відповідь. HTTP-запит — це спосіб, у який платформи Інтернет-комунікації, наприклад веббраузери, запитують інформацію, необхідну для завантаження вебсайту. Кожен HTTP-запит, виконаний через Інтернет, несе в собі ряд закодованих даних, які містять різні типи інформації. Типовий HTTP-запит містить:

- тип версії HTTP;
- URL;
- метод HTTP;
- заголовки запитів HTTP;
- додаткове тіло HTTP.

Метод HTTP, який іноді називають дієсловом HTTP, вказує на дію, яку HTTP-запит очікує від запитуваного сервера. Наприклад, двома найпоширенішими методами HTTP є «GET» і «POST»; запит «GET» очікує повернення інформації (зазвичай у формі вебсайту), тоді як запит «POST» зазвичай вказує на те, що клієнт надсилає інформацію на вебсервер (наприклад, інформацію форми, подане ім'я користувача та пароль). Основними методами протоколу HTTP є:

- OPTIONS – запит для визначення можливостей сервера;
- GET – запит вмісту даного ресурсу;
- POST – передача даних або завантаження файлів на сервер;
- PUT – оновлення ресурсів;
- DELETE – видалення ресурсу.

HTTP-заголовки містять текстову інформацію, що зберігається в парах ключ-значення, і вони включені в кожен HTTP-запит. Ці заголовки передають основну інформацію, наприклад, який браузер використовує клієнт, які дані запитуються. Приклад заголовку показано на рис. 4.

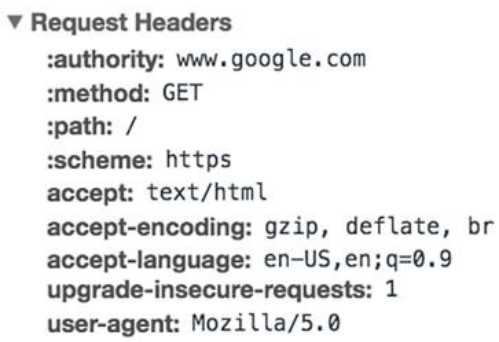

Рис. 4. Приклад HTTP заголовку запиту.

Тіло запиту – це частина, яка містить «тіло» інформації, яку запит передає. Тіло запиту HTTP містить будь-яку інформацію, яка надсилається на вебсервер, наприклад ім'я користувача та пароль, або будь-які інші дані, введені у форму.

Відповідь HTTP – це те, що вебклієнти (часто браузери) отримують від Інтернет-сервера у відповідь на запит HTTP. Ці відповіді передають цінну інформацію на основі запиту HTTP. Типова відповідь HTTP містить:

- код статусу HTTP;
- заголовки відповідей HTTP;
- додаткове тіло HTTP.

Коди статусу HTTP – це 3-значні коди, які найчастіше використовуються для вказівки, чи було успішно виконано запит HTTP. Коди стану розбиті на такі 5 блоків:

- 1XX інформаційний;
- 2XX успіх;
- 3XX перенаправлення;
- 4XX помилка клієнта;
- 5XX помилка сервера.

«XX» означає різні цифри від 00 до 99.

Подібно до HTTP-запиту, HTTP-відповідь містить заголовки, які передають важливу інформацію, таку як мова та формат даних, які надсилаються в тілі відповіді (рис. 5).

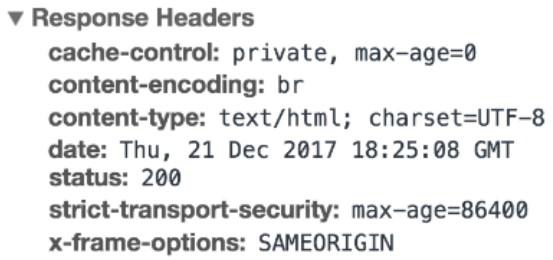

Рис. 5. Приклад HTTP заголовку відповіді

Успішні відповіді HTTP на запити «GET» зазвичай мають тіло, яке містить запитувану інформацію. У більшості вебзапитів це дані HTML, які вебпереглядач перетворює на вебсторінку.

Архітектуру презентаційного рівня наведено на рис. 6. Архітектурою клієнтської частини було обрано SSA (Server side application). Перевагою такої реалізації є можливість використання потужностей сервера під час роботи додатку. У такому випадку коли користувач завантажуватиме вебсторінку у браузері фронт-енд зробить усі обрахунки та завантаження необхідний даних на стороні сервера, де буде розгорнутий, після того у браузер клієнта надійде уже готовий HTML для перегляду. Такий підхід дозволяє зменшити навантаження на машину користувача. Але є і недолік: в такому випадку браузер користувача під час будь-яких дій запитуватиме оновлення сторінки у сервера через під'єднаний інтерфейс спілкування (рис. 7).

Програмний код є розділеним на три частини: клієнтська частина, серверна частина та база даних. Клієнтська частина розроблена як SSA (Server Side Application), із використанням фреймворку Blazor та його моделлю Blazor Server. Серверна частина використовує технологію ASP.NET Core та розділена на три проекти: WebApi, Business, DataAccess. Проект бази даних складається із описаних таблиць, процедур та сценаріїв,

що виконуються після розгортання [6,7]. В проєкт інтегровано бот месенджера Telegram для забезпечення швидкого доступу до документів [8].

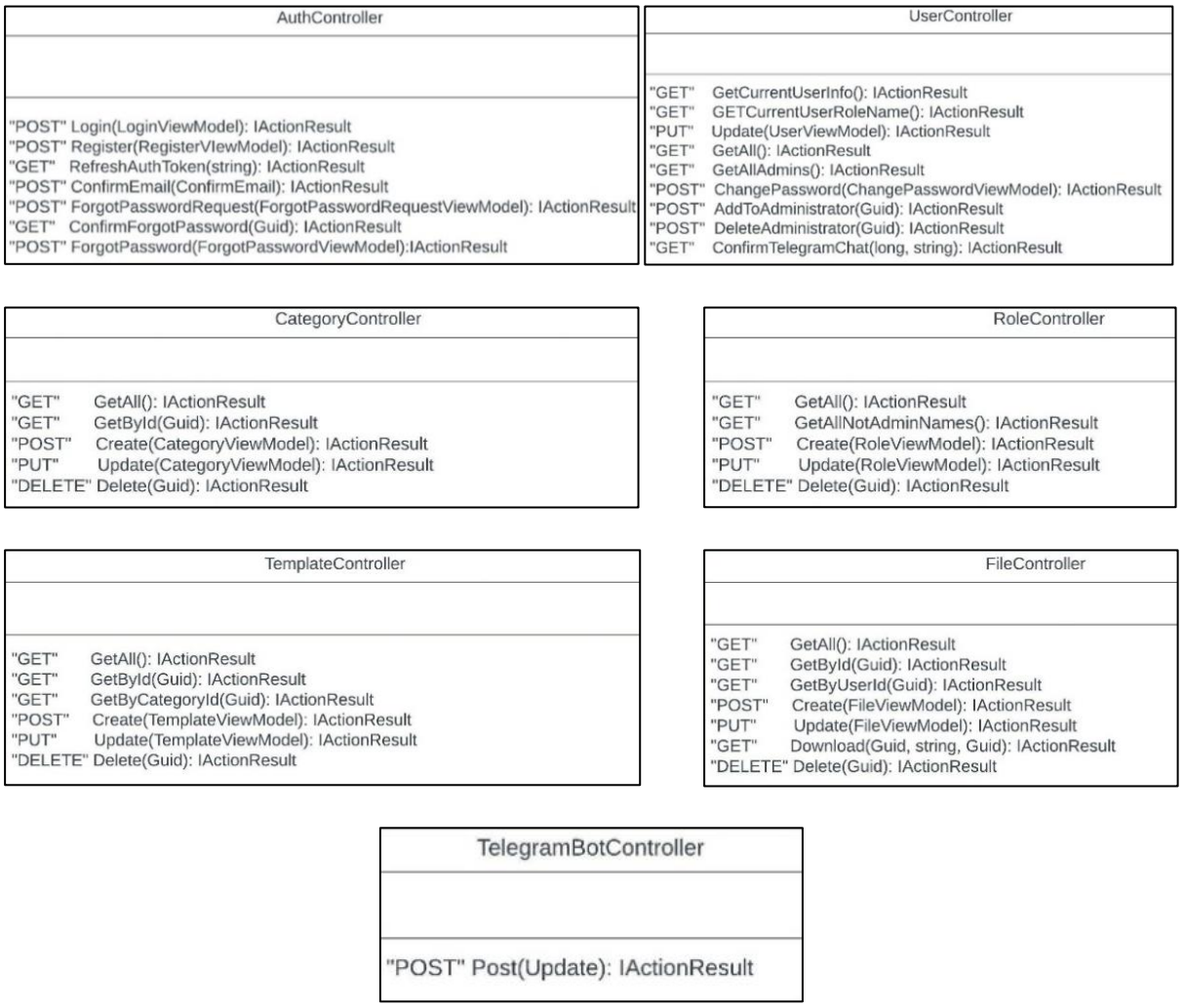

Рис. 6. Архітектура презентаційного рівня

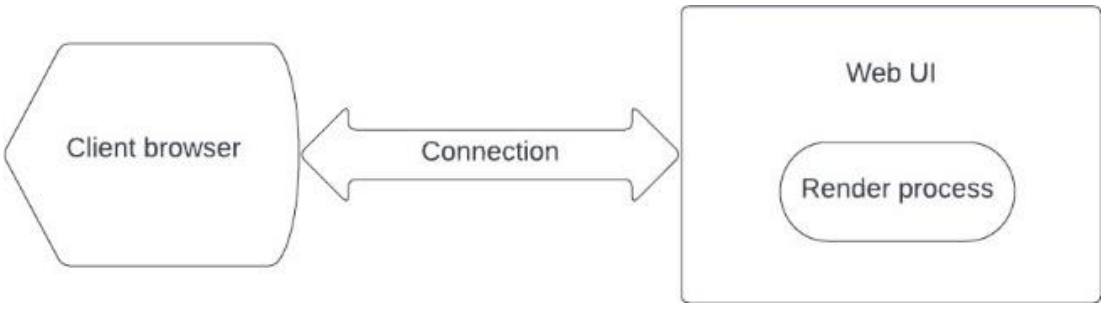

Рис. 7. Принцип архітектури SSA

Висновки. У роботі було досліджено основні аспекти роботи документообігу факультету закладу вищої освіти та підходи до проєктування, розробки і впровадження його електронного варіанту. Досліджено основні вимоги та необхідну функціональність для вебсервісу електронного документообігу. Обрано концепцію та методи розробки програмного забезпечення. Досліджено принципи побудови SSA (Server Side

Application) та API (Application Programming Interface), принципи розробки бази даних та структурування даних для забезпечення швидкого доступу до інформації, а також можливості інтеграції ботів месенджера Telegram для забезпечення швидкого доступу до документів.

Бібліографія

1. Ткачук Г. І. Використання електронної системи документообігу у ВНЗ. *Магістратура в умовах євроінтеграційних процесів вищої школи*. 2014. С. 254.
2. Корнійчук К. С. Електронний документообіг в інфраструктурі управління промисловим підприємством. *Вісник Харківської державної академії культури. Серія: Соціальні комунікації.* 2017. № 50. С. 188–197.
3. Копняк К. В., Костунець Т. А. Автоматизація документообігу як складова підвищення ефективності діяльності підприємства. *Економіка. Фінанси. Менеджмент: актуальні питання науки і практики*. 2017. Т. 11, № 27. С. 57–68.
4. Гордійчук Г. П., Булатецька Л. В. Організація схеми ролей та дозволів для користувачів при розробці системи документообігу факультету. *Науково-практична конференція,присвячена 130-річчю від дня народження М. П. Кравчука* : матеріали конф., м. Луцьк, 11 жовт. 2022 р. Луцьк, 2022. С. 133–134.
5. Гордійчук Г. П., Булатецька Л. В. Підхід до організації зберігання даних в системі документообігу факультету. *Математика. Інформаційні технології. Освіта.* : матеріали XI міжнар. науково-практ.конф., м. Луцьк, 3–5 черв. 2022 р. Луцьк, 2022. С. 59–60.
6. Гордійчук Г. П., Булатецька Л. В. Розробка системи електронного документообігу факультету на платформі .NET. *Молода наука Волині: пріоритети та перспективи досліджень* : МАТЕРІАЛИ XVI Міжнар. науково-практ. конф. студентів, аспірантів та молодих вчен., м. Луцьк, 17 трав. 2022 р. Луцьк, 2022. С. 415–417.
7. ASP.NET documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1 (date of access: 21.03.2023).
8. Telegram Bot API. *Telegram APIs*. URL: https://core.telegram.org/bots/api (date of access: 21.03.2023).

References

1. Tkachuk H. I. Vykorystannia elektronnoi systemy dokumentoobihu u VNZ. *Mahistratura v umovakh yevrointehratsiinykh protsesiv vyshchoi shkoly*. 2014. S. 254.
2. Korniichuk K. S. Elektronnyi dokumentoobih v infrastrukturi upravlinnia promyslovym pidpryiemstvom. *Visnyk Kharkivskoi derzhavnoi akademii kultury. Seriia: Sotsialni komunikatsii.* 2017. № 50. S. 188–197.
3. Kopniak K. V., Kostunets T. A. Avtomatyzatsiia dokumentoobihu yak skladova pidvyshchennia efektyvnosti diialnosti pidpryiemstva. *Ekonomika. Finansy. Menedzhment: aktualni pytannia nauky i praktyky*. 2017. T. 11, № 27. S. 57–68.
4. Hordiichuk H. P., Bulatetska L. V. Orhanizatsiia skhemy rolei ta dozvoliv dlia korystuvachiv pry rozrobtsi systemy dokumentoobihu fakultetu. *Naukovo-praktychna konferentsiia,prysviachena 130-richchiu vid dnia narodzhennia M. P. Kravchuka* : materialy konf., m. Lutsk, 11 zhovt. 2022 r. Lutsk, 2022. S. 133–134.
5. Hordiichuk H. P., Bulatetska L. V. Pidkhid do orhanizatsii zberihannia danykh v systemi dokumentoobihu fakultetu. *Matematyka. Informatsiini tekhnolohii. Osvita.* : materialy KhI mizhnar. naukovo-prakt.konf., m. Lutsk, 3–5 cherv. 2022 r. Lutsk, 2022. S. 59–60.
6. Hordiichuk H. P., Bulatetska L. V. Rozrobka systemy elektronnoho dokumentoobihu fakultetu na platformi .NET. *Moloda nauka Volyni: priorytety ta perspektyvy doslidzhen* : MATERIALY KhVI Mizhnar. naukovo-prakt. konf. studentiv, aspirantiv ta molodykh vchen., m. Lutsk, 17 trav. 2022 r. Lutsk, 2022. S. 415–417.
7. ASP.NET documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1 (date of access: 21.03.2023).
8. Telegram Bot API. *Telegram APIs*. URL: https://core.telegram.org/bots/api (date of access: 21.03.2023).