

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОНСТРУКТОРА
HTML ВІДЖЕТІВ ДЛЯ ВЕБЗАСТОСУНКІВ
DEVELOPMENT AND SOFTWARE IMPLEMENTATION OF THE HTML
WIDGETS CONSTRUCTOR FOR WEB APPLICATIONS

Виконав: студент 4 курсу, групи КНТ-120сп
Спеціальності 121 Інженерія
програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДУБИНА О.М.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДУБІНИ Олександра Максимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка та програмна реалізація конструктора HTML віджетів для вебзастосунків. Development and Software Implementation of the HTML Widgets Constructor for Web Applications

керівник проєкту (роботи) к.т.н., доцент, ГЛАДКОВА Ольга Миколаївна,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 04 ” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання “ 4 ” квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Олександр ДУБІНА
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Ольга ГЛАДКОВА
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 97 с., 53 рис., 3 табл., 2 дод., 18 джерел.

JAVASCRIPT, TYPESCRIPT, HTML, MYSQL, БАЗА ДАНИХ, ВІДЖЕТ, ВЕБЗАСТОСУНОК

Об'єкт дослідження – процес розробки програмного забезпечення для створення HTML-віджетів.

Предмет дослідження – програмні засоби для створення віджетів.

Метою роботи є розробка програмного забезпечення створення HTML-віджетів.

Матеріали, методи та технічні засоби: мова програмування JavaScript з використанням TypeScript та Node.js, система керування базами даних MySQL.

Результати. Розроблено програмне забезпечення для створення віджетів за допомогою мови програмування JavaScript, що дозволяє забезпечити користувачам можливість самостійно створювати віджети.

Практична цінність роботи полягає у розробці програмного забезпечення для створення HTML-віджетів.

Висновки. Виконано проєктування програмного забезпечення для створення HTML-віджетів. Розроблено програмне забезпечення для створення HTML-віджетів. Здійснено тестування розробленого програмного забезпечення для створення HTML-віджетів.

Галузь використання – сфера ІТ.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 97 pages, 53 figures, 3 tables, 2 appendixes, 18 sources.

JAVASCRIPT, TYPESCRIPT, HTML, MYSQL, DATABASE, WIDGET, WEB APPLICATION

The object of research is the process of developing software for creating HTML-widgets.

The subject of the research is software for creating HTML-widgets.

The purpose of the work is to develop software for creating HTML widgets.

Materials, methods and technical means: JavaScript programming language using TypeScript and Node.js, MySQL database management system.

The results. Developed software for creating widgets using the JavaScript programming language, which allows users to create widgets themselves.

The practical value of the work lies in the development of software for creating HTML widgets.

Conclusions. Software design for creating HTML widgets has been completed. Developed software for creating HTML widgets. The developed software for creating HTML widgets was tested.

Scope of use is the field of IT.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

TS – TypeScript

DB – Database

HTML – Hypertext Markup Language

JS – JavaScript

IT – Інформаційні технології

БД – База даних

ЗМІСТ

Перелік скорочень та умовних позначок.....	6
Вступ.....	9
1 Аналіз предметної області та постановка завдання.....	10
1.1 Огляд популярних інформаційних ресурсів для створення html-віджетів.....	10
1.2 Огляд стеку технологій та засобів для створення віджетів.....	11
1.3 Висновки за розділом та постановка задачі.....	15
2 Вибір інструментарію розробки системи.....	16
2.1 Вибір середовища розробки.....	16
2.2 Вибір основного браузеру в процесі розробки.....	18
2.3 Тестування додатку на інших браузерах.....	20
2.4 Порівняння роботи різних арі.....	24
2.5 Висновки за розділом 2.....	26
3 Проєктування системи.....	27
3.1 Вибір стеку технологій для розробки.....	27
3.2 Проєктування архітектури системи.....	31
3.3 Проєктування графічного інтерфейсу.....	34
3.4 Проєктування архітектури бази даних.....	37
3.5 Висновки до розділу 3.....	39
4 Керівництво з використання програмного продукту.....	40
4.1 Призначення та умови використання програмного продукту	40
4.2 Функції системи.....	40
4.3 Вхідні та вихідні дані.....	40
4.4 Взаємодія із програмним продуктом	41
4.5 Висновки до розділу 4.....	49
Висновки.....	50
Перелік джерел посилання.....	51

Додаток А Текст програми.....	53
Додаток Б Слайди презентації.....	91

ВСТУП

У сучасному світі час – найцінніший та найдорожчий ресурс, недбале ставлення до якого може привести до катастрофічних наслідків, таких як, наприклад, великих грошових втрат. Це, авжеж, також стосується сфери Інформаційних Технологій (далі – ІТ).

Інформаційні технології та зв'язок – це сфера, що відповідає за збір, зберігання і передачу інформації за допомогою технічних пристроїв.

Сьогодні жодна галузь не може нормально існувати та взагалі функціонувати без фахівців з інформаційних технологій: збір та аналіз статистики, швидка передача інформації привели до якісних змін майже в кожному напрямку. Автоматизація на виробництвах, електронний облік документів в державних управліннях та медицині, дистанційна освіта – це гарні приклади успішного повсюдного впровадження інформаційно-комунікаційних технологій.

Постійний та надшвидкий розвиток цієї сфери призводить до великої конкуренції як у потенціальних програмістів, так і у ІТ-компаній, бо ІТ-сфера являється однією з найперспективніших галузей в світовій економіці.

Тож швидка та якісна розробка сайтів, додатків та іншого програмного забезпечення забезпечить гарну репутацію та високу конкурентоспроможність як окремої компанії, так і самого програміста.

Саме тут на допомогу приходять різні конструктори сайтів, фреймворки, бібліотеки та інші набори інструментів. Ці засоби створені спеціально для того, аби полегшити та пришвидшити роботу програміста.

Тож можна сказати, що на даний момент часу додаток для створення HTML-віджетів може стати найактуальнішою задачею, бо це допоможе підвищити ефективність, спростити та пришвидшити процес розробки веб-сайтів та додатків як для компаній, так і для програмістів в цілому, або навіть для звичайних користувачів, які бажають мати на робочому столі свого персонального пристрою всі необхідні засоби та програми.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд популярних інформаційних ресурсів для створення html-віджетів

Авжеж, існує багато інструментів для створення віджетів, таких як, наприклад, бібліотека Bootstrap[1] – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-застосунків, який містить шаблони CSS та HTML для форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення мови JavaScript[2]. Ця бібліотека значно спрощує розробку динамічних сайтів і веб-додатків.

Але якщо брати до уваги саме окремі додатки для створення html-віджетів, а не тільки інструменти для їх створення, то наразі їх існує не так багато, та функціонал цих додатків не завжди може відповідати усім вимогам користувачів. Тож найбільш популярні та корисні додатки будуть перелічені нижче.

Widgetsmith – цей додаток дозволяє створювати більш «прокачані» віджети та виводити на екран пристрою таку інформацію: час та місцезнаходження, температуру повітря та погоду, кількість кроків, нагадування, й це тільки частина можливостей цього додатку. Є безкоштовним додатком.

Якщо ви дуже любляєте рослини та вирощуєте їх вдома, то, мабуть, знаєте про те, що можна дуже легко забути про полив. Щоби рослина не загинула, можна скористатись додатком-нагадуванням Plant Daddy. Цей застосунок дозволяє внести дані про всі ваші рослини і налаштувати зручний інтервал для нагадування про обов'язковий полив. Віджет з таким нагадуванням без проблем розміщується на екрані блокування, і можливість пропустити момент поливу зводиться практично до нуля. Недоліками такого додатку може бути те, що регулярні сповіщення інколи можуть відволікати людину. Цей додаток також безкоштовний.

Якщо Вам потрібен дійсно корисний додаток, зверніть свою увагу на Code Widgets. Програма дозволяє додавати на екран штрих-код карт, якими ви часто користуєтеся, щоб не залазити до програми «Гаманець» щоразу, коли стоїте на касі. Ви можете додавати стільки карт, скільки вам потрібно мати у швидкому доступі. У додатку можна налаштовувати кольори оформлення віджету за власним смаком та бажанням та задати йому будь-яку назву. На мій погляд, це один із найкращих додатків для створення віджетів. Додаток повністю безкоштовний.

Battery Widget – ця програма візуалізує вибрані показники (рівень заряду, витрата ОЗУ, кількість вільної пам'яті та навіть версію iOS) у компактному форматі. Якщо кілька шаблонів, то всі вони безкоштовні. Для зміни колірної схеми доведеться купити Pro-версію (є і передплата, і можливість одноразової покупки), тож цей додаток не є безкоштовним.

Chart.js – ця дуже проста та зрозуміла програма дозволяє створити 6 видів діаграм. Бібліотека використовує HTML5 Canvas[3] для побудови діаграм та створює простий, швидко реагуючий на зміни дизайн. Цей додаток є безкоштовним.

1.2 Огляд стеку технологій та засобів для створення віджетів

Віджет – це елемент графічного інтерфейсу користувача, що має стандартний зовнішній вигляд та виконує стандартні дії. Також віджетом називають елемент керування [4]. Цей невеликий застосунок відображає необхідну користувачеві інформацію або просто прикрашає екран пристрою. До найбільш розповсюджених віджетів відносяться стрічка новин, годинник, прогноз погоди та інші. Тож віджети корисні тим, що розширюють функціонал гаджета або онлайн-платформи.

В наш час існує безліч готових програм та засобів, які не потребують від користувача якихось особливих навичок.

Наприклад, платформа Salesforce містить в собі все необхідне для керування вашим сайтом або додатком, продажами, налаштування реклами, таргетингу, управління фінансами та навіть для створення віджетів [5]. Інтерфейс платформи наведено на рисунку 1.1.

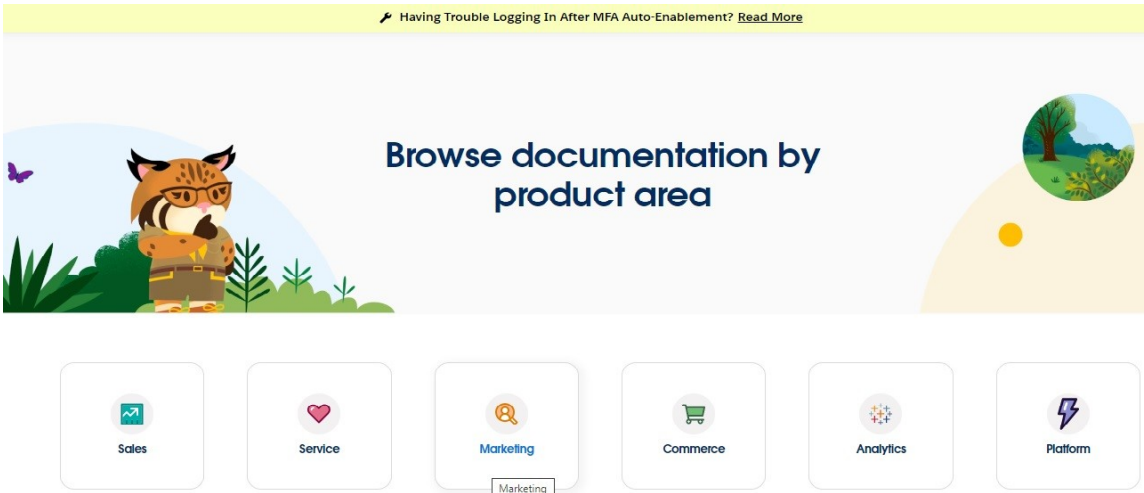


Рисунок 1.1 – Платформа Salesforce [5]

Сторінка створення віджетів у системі Salesforce наведено на рис. 1.2.

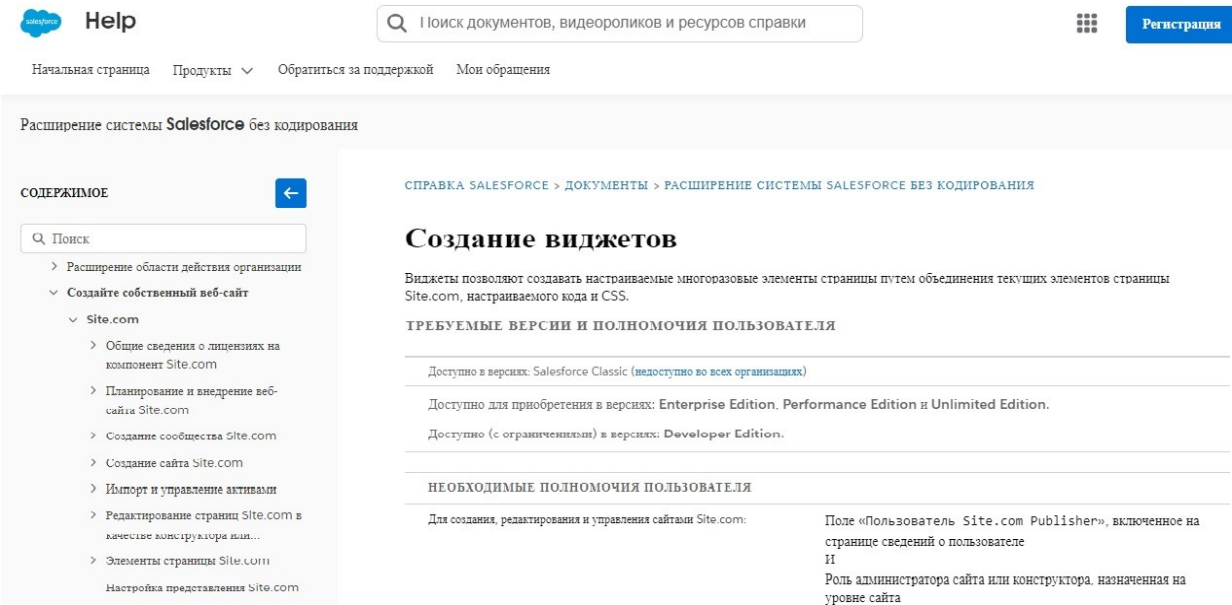


Рисунок 1.2 – Створення віджетів у системі Salesforce [5]

Також створювання віджетів можливо завдяки платформі IBM – однієї з найбільших американських електронних корпорацій та однієї із найбільших світових виробників усіх видів комп'ютерів і програмного забезпечення. Компанія об'єднує всі необхідні технології та послуги, незалежно від походження цих рішень, щоб допомогти клієнтам вирішити найактуальніші бізнес-проблеми. Сторінка створення віджетів продемонстрована на рис. 1.3.

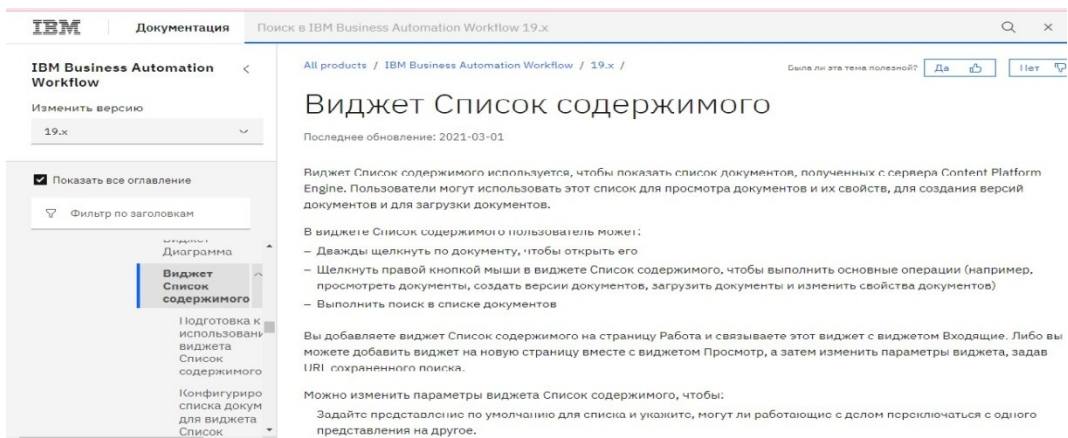


Рисунок 1.3 – Створення віджетів у системі IBM [6]

Перелік послуг компанії IBM наведено на рис.1.4.

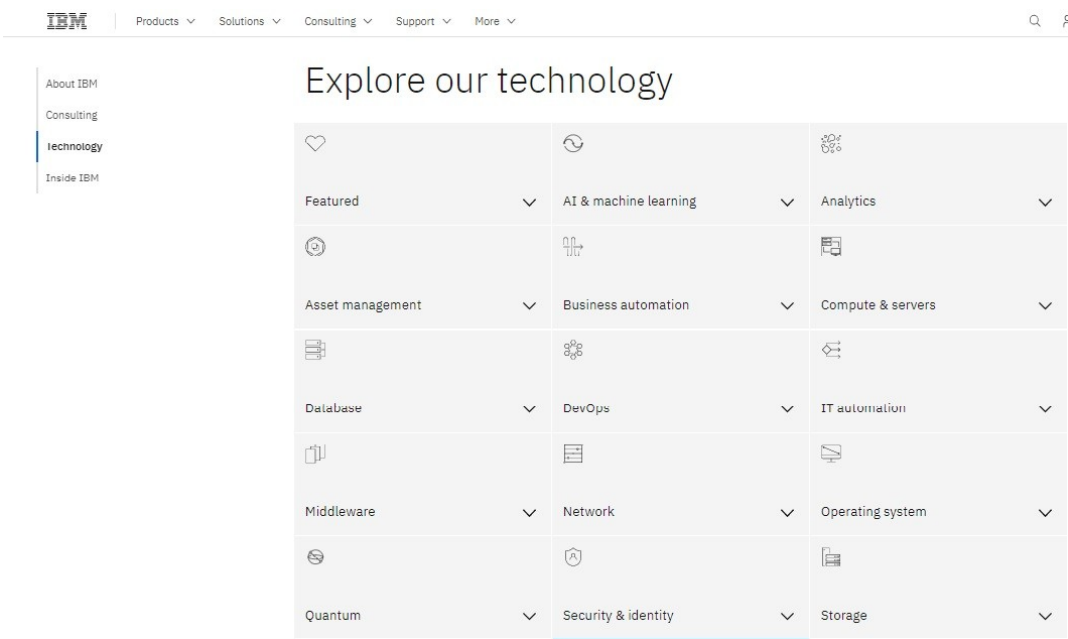


Рисунок 1.4 – Послуги компанії IBM [6]

Щодо більш простіших та дешевших засобів, то можна зазначити плагін GetSale.

Ні для кого не секрет, що розробка та супровід інтернет-магазину – це лише мала частина роботи з його просування. Основне завдання - це збільшення продажів, підвищення впізнаваності та збільшення відвідуваності сайту. Новому власнику сайту іноді складно розібратися в цілях, вирвах продажів, сегментуванні та інших премудростях. Та й просунуті веб-майстри воліють не витрачати час на рутину.

Саме тому є спеціальні сервіси для інтернет-магазинів, які покращують поведінкові характеристики сайту. До таких як раз і відноситься GetSale (рис.1.5). Та, авжеж, GetSale пропонує різноманітні віджети, які полегшують комунікацію з клієнтом, і, зрештою, дозволяють утримати його на сайті.

Ви можете налаштувати покази віджетів (в особистому кабінеті) під час виконання цілей. Наприклад, запропонувати користувачеві знижку, як тільки він виконав мету "додавання товару в кошик" (натиснув кнопку "В кошик"). Такі умови дозволяють дуже точно налаштувати показ віджетів і досягти їхньої максимальної ефективності.

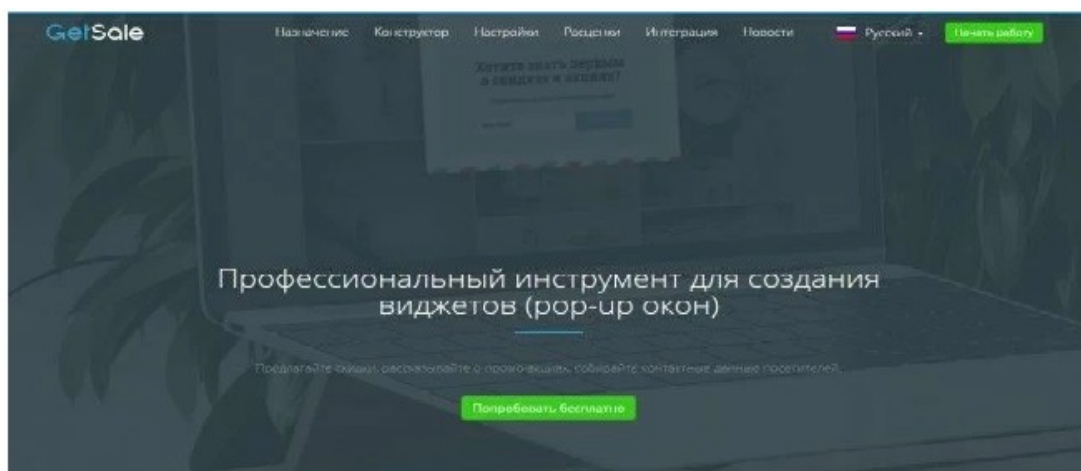


Рисунок 1.5 – Створення віджетів за допомогою сервісу GetSale

У процесі дипломного проєктування було проведено аналіз вищенаведених інформаційних систем, визначені їх можливості, виділено переваги та зазначено недоліки.

Переваги:

- сучасний, зрозумілий та зручний функціонал, особливо для керування великим бізнесом;
- простий інтерфейс;
- деякі засоби потребують лише базові знання з програмування або не потребують їх зовсім.

Недоліки:

- надто висока вартість засобів, що робить їх недоступними для більшості користувачів;
- ресурсів деяких платформ та засобів може не вистачати для вирішення усіх поставлених задач;
- створення віджетів можливе лише в межах обраної системи;
- обмежений вибір функціоналу віджетів;
- закритий код системи, що унеможливорює парсинг даних для подальших розробок.

1.3 Висновки за розділом 1

Сучасні платформи та системи мають велику кількість переваг для розвитку бізнесу, наприклад інтернет-магазинів, тощо, але мають дуже маленький функціонал для створення віджетів, які могли б покращувати інтерфейс та позитивно сприяти на розвиток бізнесу. Також вони мають свої недоліки.

Тож, беручі до уваги усе вищенаведене, можна спробувати подолати недоліки існуючих програм у процесі створення нової програмної системи, яка буде мати необхідний функціонал та набір інструментів для простого та швидкого створення якісних та корисних віджетів.

2 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ СИСТЕМИ

2.1 Вибір середовища розробки

В якості середовища розробки для даного додатку було обрано Visual Studio Code[7], який частіше називають VS Code — це сучасний редактор вихідного коду, створений Microsoft із Electron Framework для Windows, Linux і macOS. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git[8]. Користувачі мають змогу змінювати тему, комбінації клавіш, параметри та встановлювати розширення, які додають функціональність.

Особливості та переваги даної середовища розробки полягають в тому, що її можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python та ін.. Він базується на структурі Electron, яка використовується для розробки веб-додатків Node.js, що працюють на механізмі компонування Blink.

VS Code використовує той самий компонент редактора (під кодовою назвою «Monaco»), який використовується в Azure DevOps.

Visual Studio Code також постачається з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML, а також підтримує налагодження Node.js. Підтримка додаткових мов може бути забезпечена безкоштовними розширеннями на VS Code Marketplace.

Також варто додати, що Код Visual Studio дозволяє користувачам встановлювати кодову сторінку, на якій буде збережено активний документ, символ нового рядка та навіть мову програмування активного документа.

Це дозволяє використовувати його на будь-якій платформі, у будь-якій локальній мережі та для будь-якої заданої мови програмування, що, безперечно, є однією з найважливіших переваг цього редактора серед безлічі інших середовищ розробки.

Visual Studio Code збирає дані про використання та надсилає їх до Microsoft, хоча цю функцію можна вимкнути. Через відкритий вихідний код програми, код телеметрії доступний для громадськості, яка може бачити, що саме збирається. У таблиці 2.1 наведено порівняння редактора коду Visual Studio Code та не менш популярних середовищ Atom та WebStorm [9]-[10].

Таблиця 2.1 – Порівняння сучасних середовищ розробки

Категорія	VS Code	Atom	WebStorm
Динамічний набір тексту	+	+	+
Розширення / Плагін	+	+	+
Операційна система	Linux, Windows Mac OS X	Linux Windows Mac OS X	Linux Windows Mac OS X
Автозаповнення коду	+	+	+
Швидкість	+	–	–
Легко встановити	+	–	–
Займає мало місця в пам'яті	+	–	–
Продуктивність	+	–	+
Дизайн	-	+	–
Гнучкість та зручність використання	+	–	–
Підтримка більшості мов	+	–	–
Материнська компанія	Microsoft	GitHub	JetBrains

Слід зазначити, що в опитуванні розробників Stack Overflow-2022 серед 71010 респондентів Visual Studio Code назвали найпопулярнішим інструментом середовища розробника, при цьому 74,48% повідомили, що вони ним користуються.

Саме завдяки всім вищенаведеним особливостям та перевагам дана система була обрана в якості основного середовища розробки проекту.

2.2 Вибір основного браузеру в процесі розробки

В якості основного цільового браузеру, на який розробник спирається при створенні програмного продукту, було обрано браузер Google Chrome[11] – це безкоштовний веб-браузер, розроблений компанією Google[12] на основі браузеру з відкритим кодом Chromium та іншого відкритого програмного забезпечення. Можна сказати, що його програмна архітектура була створена з чистого аркуша з метою задоволення поточних потреб користувачів. Нова архітектура насамперед визнає той факт, що на сьогодні більшість вебсайтів є не просто вебсторінками, а й веб-програмами.

Заявленими перевагами цієї архітектури є підвищена стабільність, швидкість, безпека, а також чистий, простий та ефективний інтерфейс користувача.

У таблиці 2.2. буде наведено порівняння найпопулярніших веббраузерів [13].

Таблиця 2.2 – Порівняння веб-браузерів

Категорія	Google Chrome	Opera	Mozilla Firefox	Internet Explorer
Синхронізація вкладок	+	+	+	+
Приватний режим	+	+	+	+
Підтримка Flash	+	–	–	–
Режим стиснення даних	+	+	–	–
Продуктивність	+	–	+	–
Вбудовані інструменти розробника	+	–	+	–

На рис. 2.1. продемонстровано рейтинг використання веб-браузерів в Україні станом на 2017 рік.

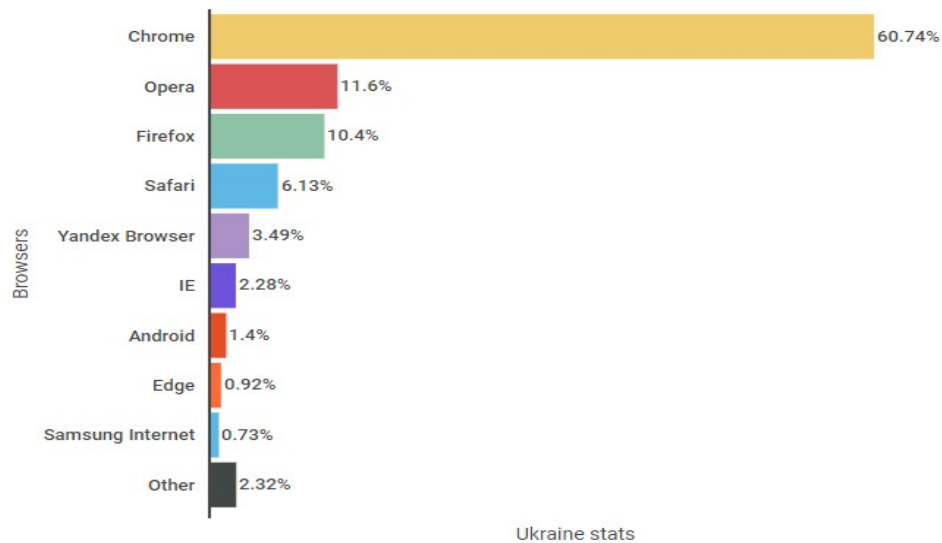


Рисунок 2.1 – Рейтинг використання веб-браузерів в Україні станом на 2017р.

В українському сегменті інтернету серед десктопних версій веб-браузерів станом на лютий 2019, браузер Chrome використовували близько 70 % користувачів.

Станом на березень 2023 року – Chrome займає 64,8% ринку.

Одна з найголовніших переваг даного браузера полягає в тому, що у Chrome для обробки сценаріїв JavaScript використовується JavaScript-рушій V8, а також у браузер включений зневаджувач(дебаггер) JavaScript, консоль JavaScript і менеджер завдань, в якому можна дізнатися, скільки пам'яті використовує кожна вкладка — і за потреби закрити найбільш ненажерливі. Саме це робить браузер Chrome найзручнішим для розробника і саме тому він використовується для даного проекту.

Тож програмний продукт повинен підтримуватись та працювати ледь не на всіх існуючих веб-браузерах, але не рідко виникають ситуації, коли якийсь функціонал працює не так як треба або не працює зовсім через

несуміжність з API, стилями або скриптами, які написані під основний браузер. Це може спричиняти відтік користувачів.

Саме тому має велике значення кросбраузерність сайту – це властивість сайту однаково відображатися та функціонувати у відповідності до поставленого завдання в усіх браузерах [14].

Тобто таку характеристику дають сайтам, дизайн яких однаковий як в Internet Explorer, так і в Google Chrome. Відображення сайту може відрізнитися лише дрібними деталями (закругленими куточками, рівнем тіні, градієнтною або суцільною заливкою), але не кольоровою гамою, розташуванням елементів, а також, що найважливіше у визначенні даного поняття, структурою сайту, адже особливо багато проблем виникає з блочною версткою. Тому дуже важливо розробляти веб-сайти та веб-додатки з урахуванням цих особливостей.

2.3 Тестування додатку на інших браузерах

Щоб впевнитись у працездатності веб-сайту або додатку, недостатньо протестувати його лише через один браузер. Функціонал додатку повинен однаково добре та без помилок працювати хоча б на трьох-чотирьох основних браузерах або платформах. Нижче буде представлено порівняння відображення інтерфейсу додатку на різних веб-браузерах.

Відображення головної сторінки додатку у браузері Google Chrome наведено на рис. 2.2, в Mozilla Firefox на рис. 2.3, та в Internet Explorer на рис. 2.4.

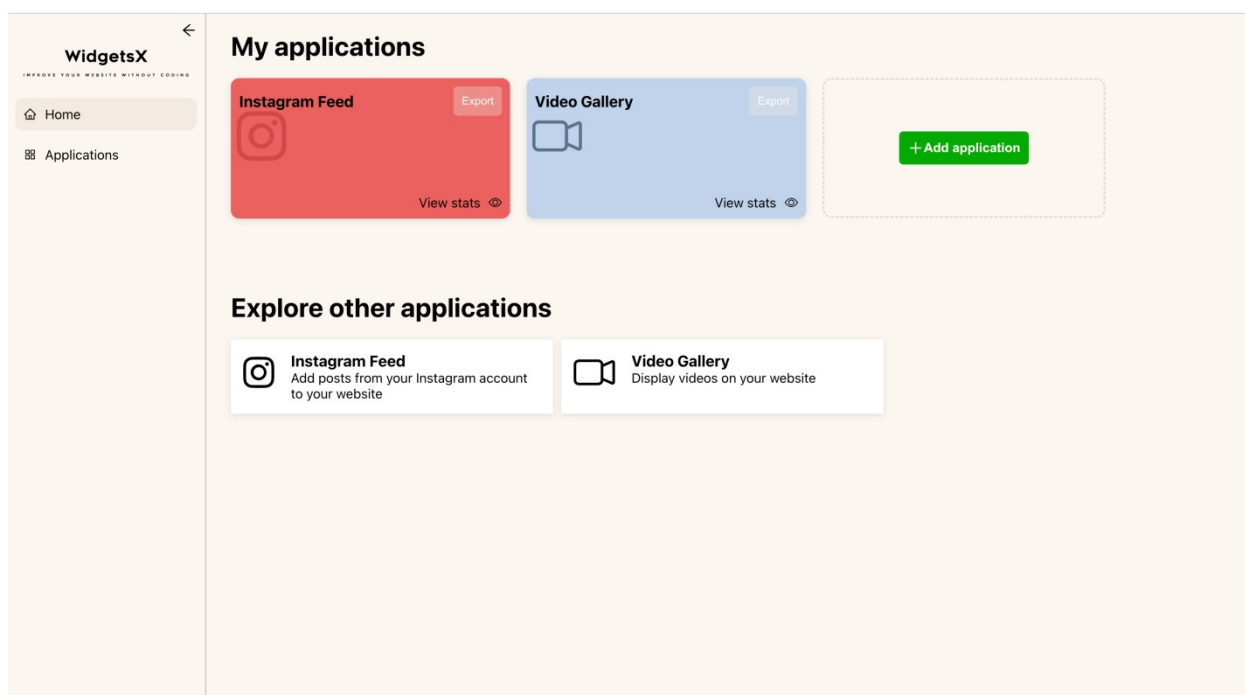


Рисунок 2.2 – Відображення головної сторінки веб-додатку у браузері Google Chrome

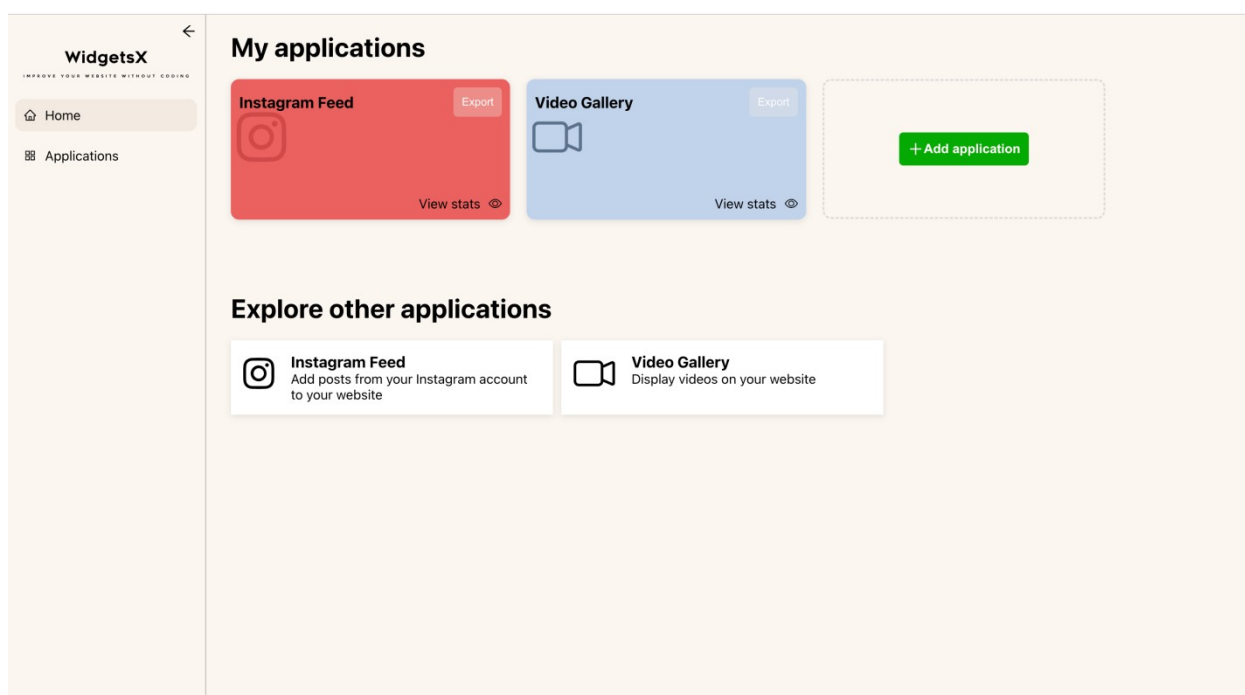


Рисунок 2.3 – Відображення головної сторінки веб-додатку у браузері Mozilla Firefox

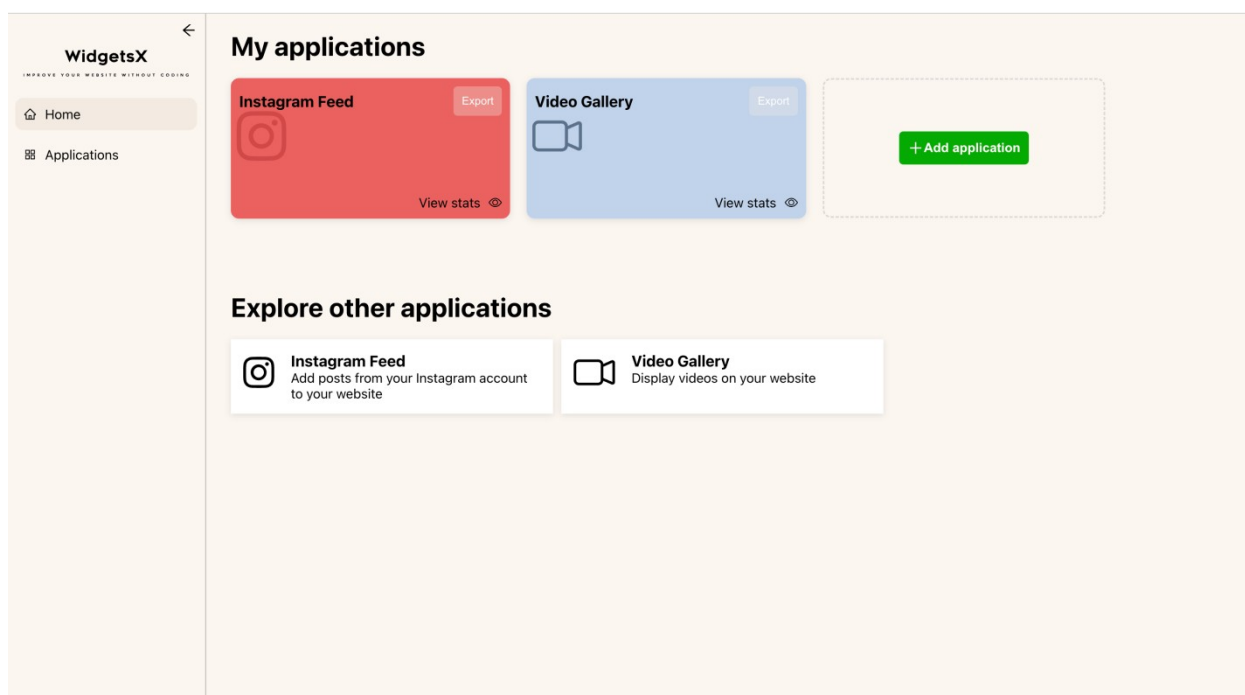


Рисунок 2.4 – Відображення головної сторінки веб-додатку у браузері Internet Explorer

Відображення сторінки авторизації у браузері Google Chrome продемонстровано на рис. 2.5.

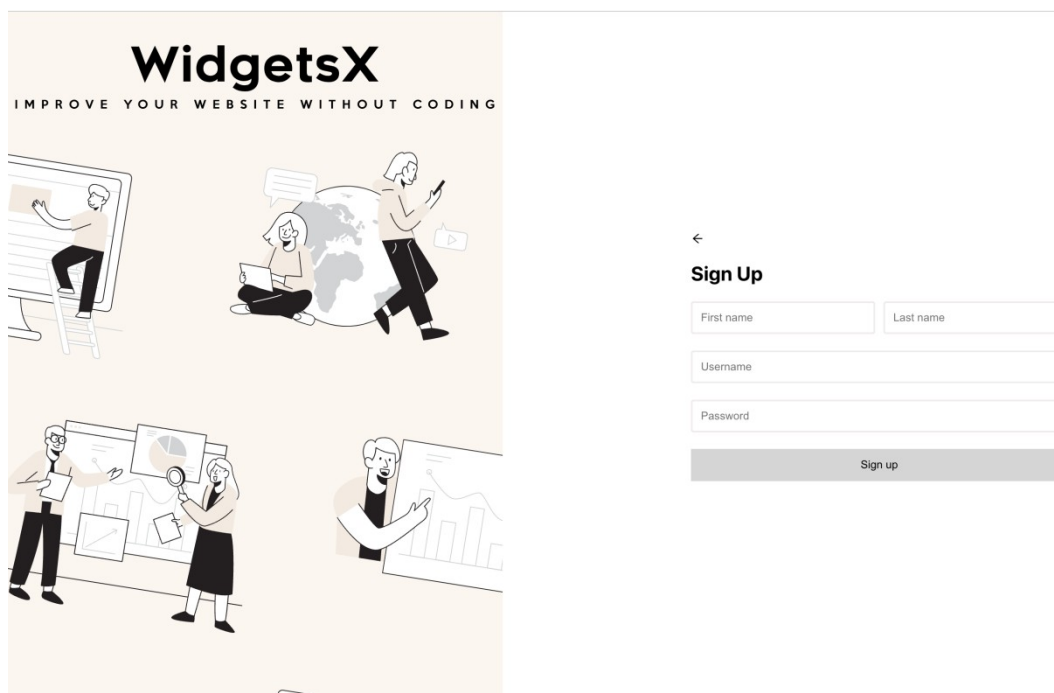


Рисунок 2.5 – Відображення сторінки авторизації у браузері Google Chrome

Відображення сторінки авторизації у браузері Internet Explorer наведено на рис. 2.6.

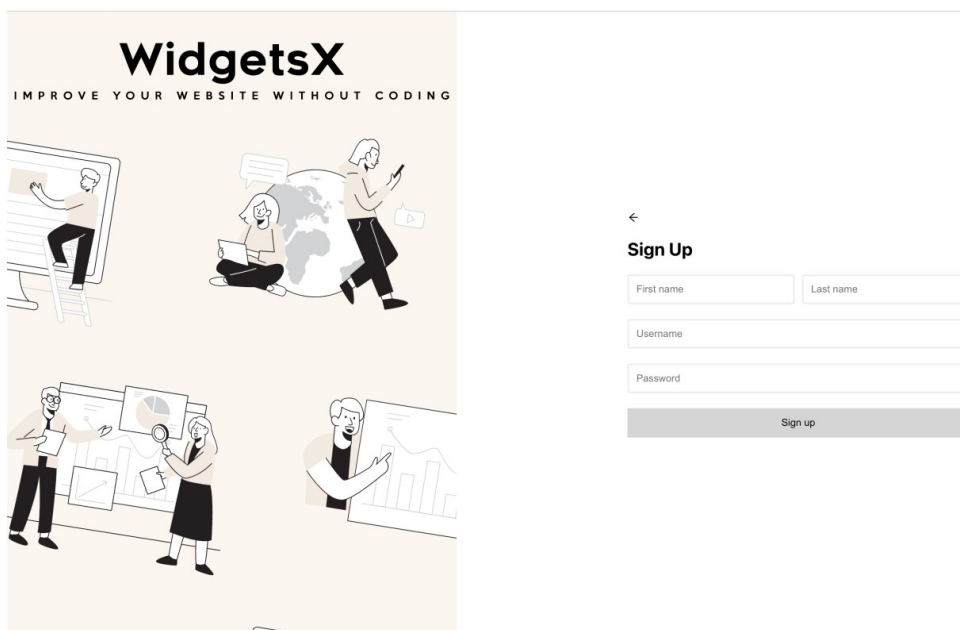


Рисунок 2.6 – Відображення сторінки авторизації у браузері Internet Explorer

Відображення сторінки авторизації у браузері Mozilla Firefox наведено на рис. 2.7.

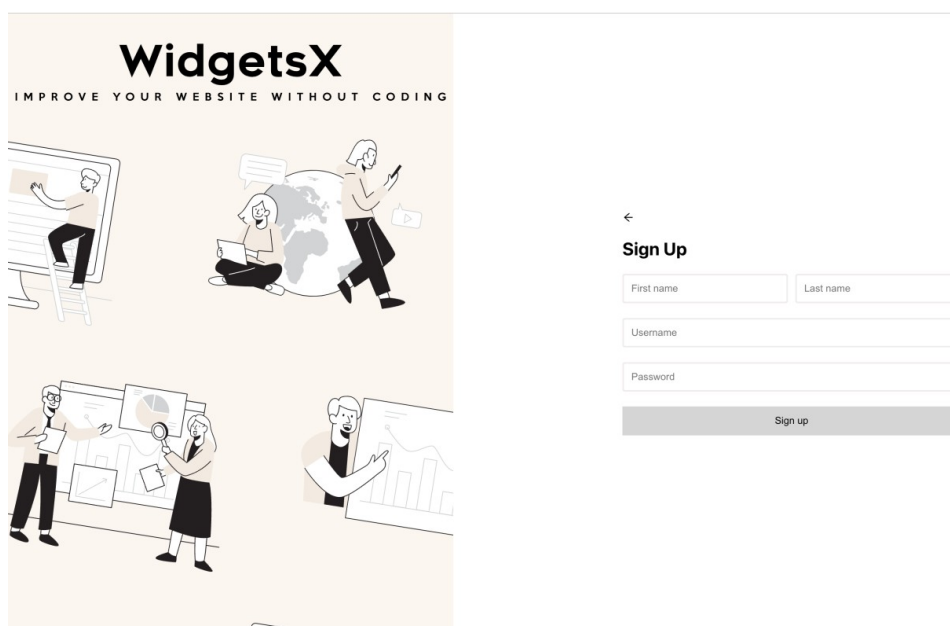


Рисунок 2.7 – Відображення сторінки авторизації у браузері Mozilla Firefox

Отже, виходячи з того, що головна сторінка веб-додатку та сторінка авторизації однаково виглядають та працюють на трьох різних браузерах, можна дійти висновку, що даний проект розроблявся з дотриманням правил та вимог кросбраузерності веб-сайту.

2.4 Порівняння роботи різних API

Іноді в браузерах API можуть працювати по-різному. Що таке API?

Інтерфейс прикладного програмування (Application Programming Interfaces, APIs[15]) – це готові конструкції мови програмування, що дозволяють розробнику будувати складну функціональність із меншими зусиллями. Вони "приховують" складніший код від програміста, забезпечуючи простоту використання.

Для JavaScript, який використовується для розробки даного застосунку, на стороні клієнта, зокрема, існує багато API. Вони не є частиною мови, а побудовані за допомогою вбудованих функцій JavaScript для того, щоб збільшити ваші можливості під час написання коду. Їх можна розділити на дві категорії:

- API браузера вбудовані в веб-браузер і здатні використовувати дані браузера та комп'ютерного середовища для більш складних дій з цими даними, наприклад, API Геолокації (Geolocation API) надає прості у використанні конструкції JavaScript для роботи з даними розташування, так що ви зможете відзначити своє розташування на карті Google Map;

- сторонні API не вбудовані в стандартний браузер, такі API та інформацію про них зазвичай потрібно шукати в інтернеті, наприклад, Twitter API дозволяє розміщувати останні твіти на вашому веб-сайті, у цьому API визначено набір конструкцій, які здійснюють запити до сервісів Twitter та повертають певні дані.

Види API наведено на рис. 2.8.

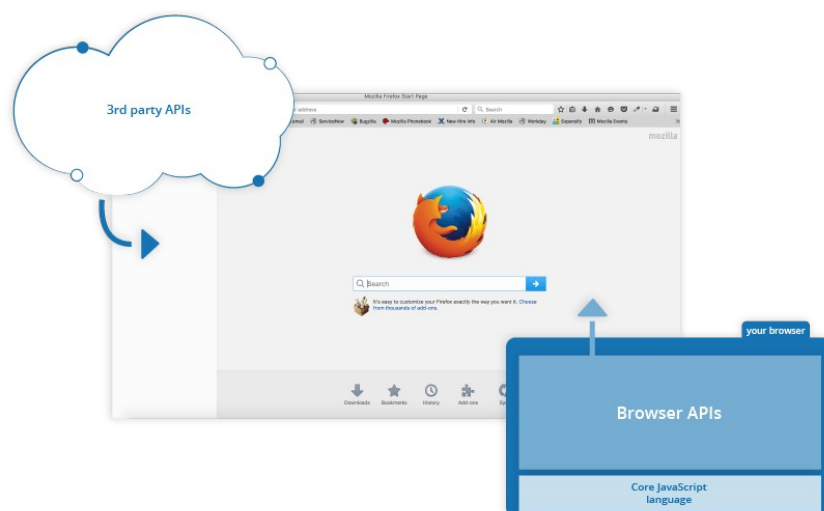


Рисунок 2.8 – Види API

Приклад роботи API продемонстровано на рис. 2.9-2.10.



Рисунок 2.9 – Як працює API

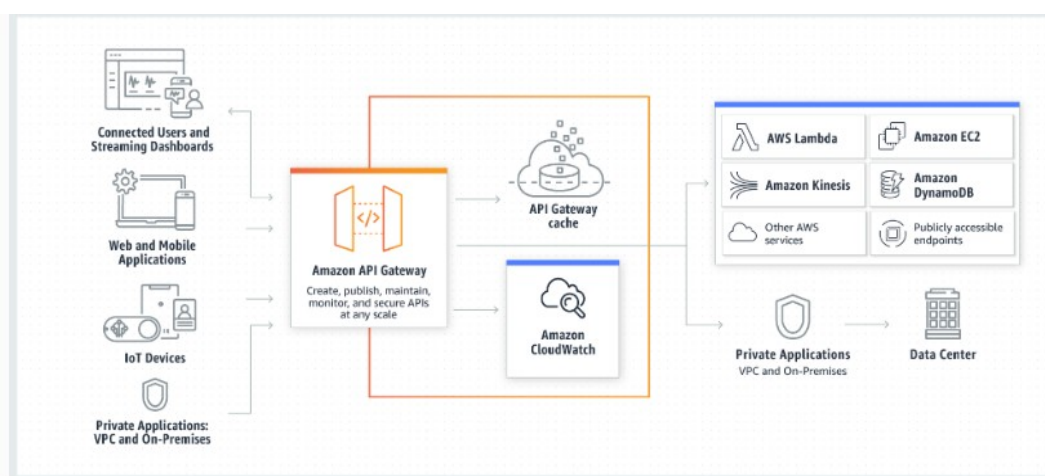


Рисунок 2.10 – Як працює API

Широке розмаїття API у сучасних браузерях дозволяє наділити вашу програму великими можливостями. Також, до найбільш популярних та часто використовуваних категорій API відносяться:

- API для роботи з документами, завантаженими у браузер, наприклад - DOM (Document Object Model) API, що дозволяє працювати з HTML та CSS - створювати, видаляти та змінювати HTML, динамічно змінювати вигляд сторінки і т.д;

- API, які приймають дані від сервера, часто використовуються для оновлення невеликих частин веб-сторінки, список API, завдяки яким це можливо, включає: XMLHttpRequest та Fetch API. Ви також могли зустріти термін Ajax, який описує цю технологію;

- API для роботи з графікою широко підтримуються браузерами, найпопулярніші з них: Canvas та WebGL та інші.

2.5 Висновки за розділом 2

Робота різних JavaScript API трохи відрізняється, але в основному в них схожі функції та принцип роботи.

Взаємодія з API у коді відбувається через один або більше об'єктів JavaScript, які служать контейнерами для інформації, з якою працює API, та реалізують функціональність, яку надає API.

Отже, API – це дуже потужний та корисний інструмент у сучасній розробці веб-застосунків, який надає багато можливостей для розширення функціоналу.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Вибір стеку технологій для розробки

При розробці даного програмного забезпечення було використано одну з найпопулярніших мов програмування JavaScript та TypeScript.

JavaScript в основному використовується для створення інтерактивних веб-сторінок. Виходячи з доступних вакансій, JavaScript є третьою найбільш затребуваною мовою 2019 року. Рейтинг мов програмування наведено на рис. 3.1.

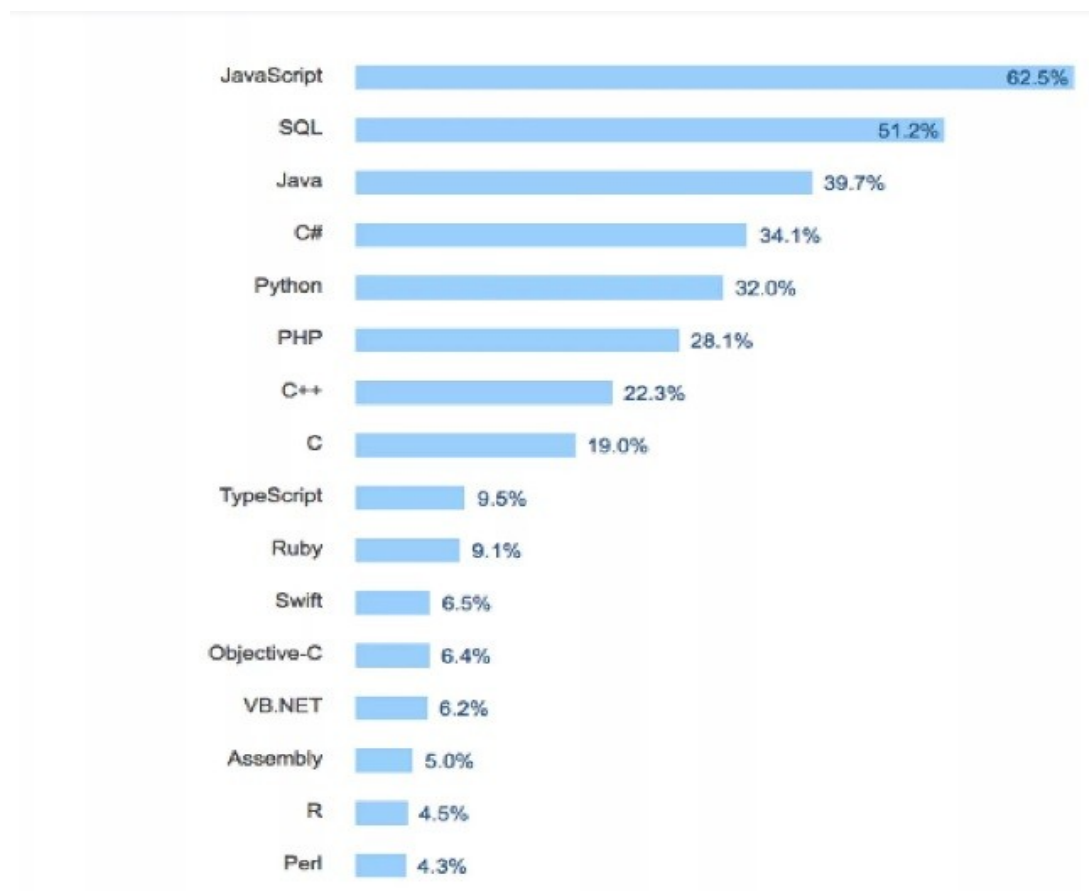


Рисунок 3.1 – Рейтинг популярності JavaScript серед розробників на 2023 рік

TypeScript — мова програмування, представлена Microsoft восени 2012 року; позиціонується як засіб розробки вебзастосунків, що розширює можливості мови програмування JavaScript [16].

TypeScript є зворотно сумісним із JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому

сучасному браузері або використовувати спільно із серверною платформою Node.js.

Переваги над JavaScript:

- можливість явного визначення типів (статична типізація);
- підтримка використання повноцінних класів (як у традиційних об'єктно-орієнтованих мовах);
- підтримка підключення модулів;
- за задумом ці нововведення мають підвищити швидкість розробки, прочитність, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Підтримка динамічної типізації зберігається — компілятор TypeScript успішно обробить і не модифікований код на JavaScript.

Основний принцип мови — будь-який код на JS сумісний з TS, тобто в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені напрацювання.

Ба більше, можна залишити наявні JavaScript-проекти в незмінному вигляді, а дані про типізації розмістити у вигляді анотацій, які можна помістити в окремі файли, які не заважатимуть розробці і прямому використанню проекту (наприклад, подібний підхід зручний при розробці JavaScript-бібліотек).

Нижче буде наведено порівняння особливостей мов програмування JS та TS (таблиця 3.1).

Дизайн та стилізація клієнтської частини додатку виконані за допомогою CSS-модулей, головна особливість яких — це унікальний хеш для кожної назви класу або id, тобто стилі можуть мати однакові назви та не мати при цьому конфліктів.

Авторизація на серверній частині додатку виконана за допомогою JWT — JSON Web Tokens[17]. Веб-токен JSON (JWT) — це відкритий стандарт який визначає компактний і самодостатній спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Цю інформацію можна

перевірити та довіряти їй, оскільки вона має цифровий підпис. JWT можна підписати за допомогою секрету (з алгоритмом HMAC) або пари відкритих/приватних ключів.

Таблиця 3.1 – Порівняння мов JavaScript та TypeScript.

Категорія	JavaScript	TypeScript
Легкість вивчення	+	–
Підтримка модулів	–	+
Підтримка статичної типізації	–	+
Підтримка функції необов’язкового параметру	–	+
Велика кількість розробників та користувачів	+	–

Приклад структури JWT наведено на рис. 3.2.

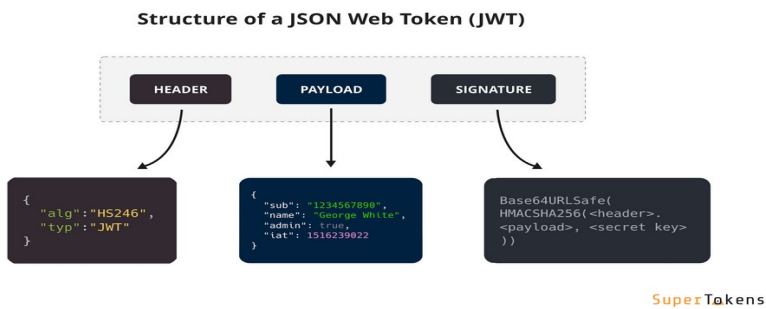


Рисунок 3.2 – Структура JWT

JSON веб-токен (JWT) призначений для передачі підписаних «заявок» (claims) між службами (як зовнішніми, так і внутрішніми для вашого застосунку/сайту). «Заявки» — частина інформації, яку інші можуть переглядати та/або перевіряти, але не змінювати.

Щоб ідентифікувати/здійснити автентифікацію користувача у вашому (веб/мобільному) застосунку, розмістіть стандартизований токен у заголовок

або url сторінки (або кінцеву точку API). Таким чином ми засвідчуємо, що користувач авторизувався та може переглядати бажаний контент.

Саме ці переваги обумовлюють необхідність використовувати JWT при розробці даного програмного продукту.

В якості системи керування базами даних було вирішено використовувати MySQL.

MySQL — вільна система керування реляційними базами даних, яка була розроблена компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам.

MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних.

Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Робота з базами даних MySQL виконується через середовище MySQL Workbench, інтерфейс якого наведено на рис. 3.3, — інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безкоштовне оточення для системи баз даних MySQL. Є наступником DBDesigner 4 з FabForce.

Асинхронність обміну даними (AJAX) забезпечує Axios(рис. 3.4)— це HTTP-клієнт, заснований на Promise для node.js та браузера. Він ізоморфний (= він може працювати у браузері та node.js з тією ж базою кодів). На стороні сервера він використовує нативний node.js http-модуль, тоді як стороні клієнта (браузер) він використовує XMLHttpRequests.

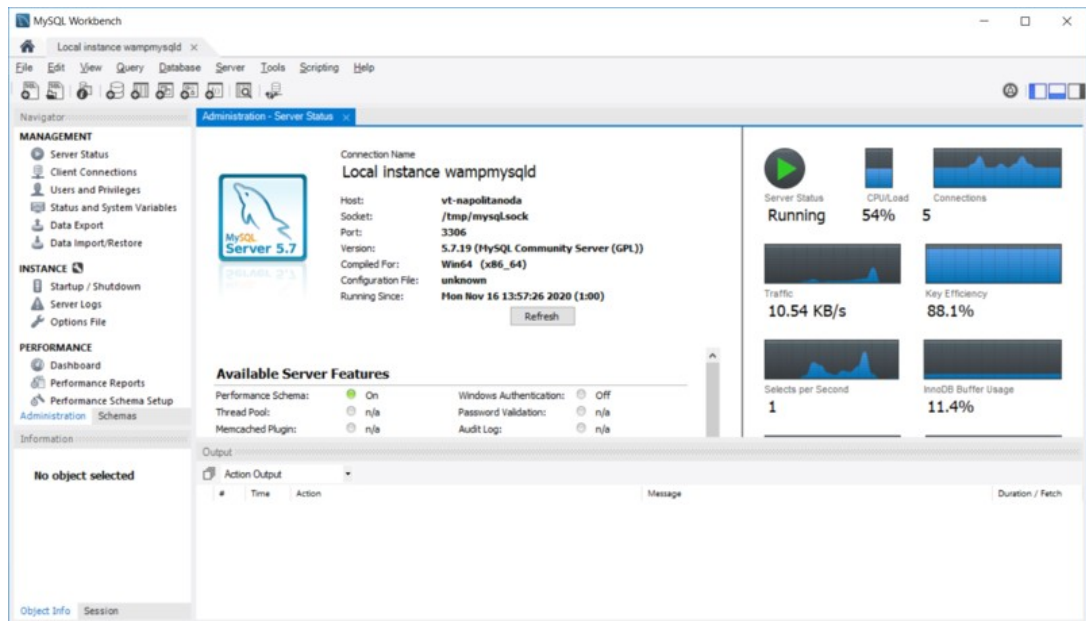


Рисунок 3.3 – Структура середовища MySQL Workbench

Тож можна підсумувати, що всі вищенаведені програмні засоби можуть забезпечити швидку розробку додатку, а також зпростити та покращити сам процес розробки.

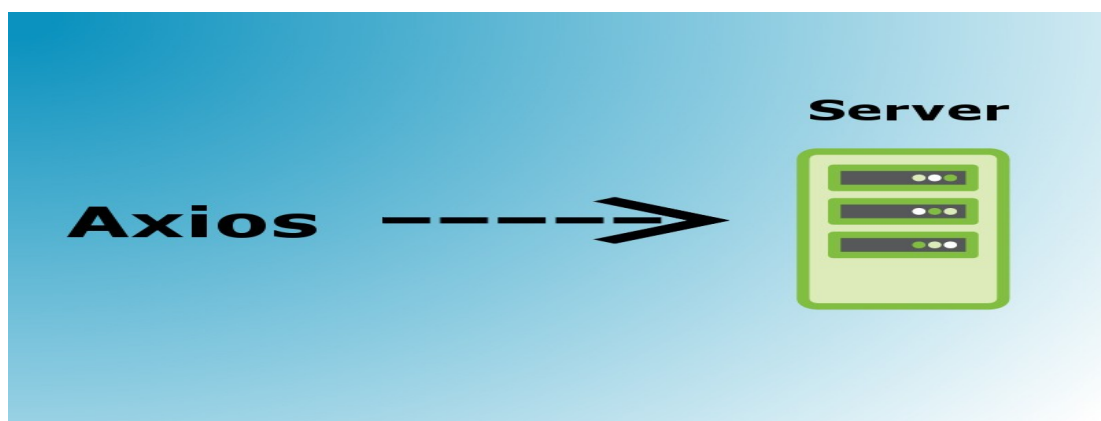


Рисунок 3.4 – HTTP-клієнт Axios

3.2 Проектування архітектури системи

Першим та дуже важливим кроком у створенні програмного засобу є проектування архітектури системи.

За основу в проектуванні архітектури було взято Мікросервіси [18] — архітектурний стиль, за яким єдиний застосунок будується як сукупність невеличких сервісів, кожен з яких працює у своєму власному процесі та спілкується з рештою, використовуючи прості та швидкі протоколи передачі даних, зазвичай HTTP.

Ці сервіси будуються навколо бізнес-потреб і розгортаються незалежно один від одного з використанням зазвичай повністю автоматизованого середовища. Існує абсолютний мінімум централізованого керування цими сервісами. Самі по собі вони можуть бути написані з використанням різних мов програмування і технологій зберігання даних.

Мікросервісна архітектура зручна для реалізації процесу безперервної поставки програмного продукту, на відміну від сервіс-орієнтовної архітектури, мікросервісна спрямована на створення одного застосунка, в той час як сервісно-орієнтована система являє собою множину застосунків, які взаємодіють між собою.

Основні властивості:

- високий рівень незалежності: незалежна розробка, незалежне розгортання;
- незалежне масштабування;
- невелика кодова база зменшує кількість конфліктів та дозволяє швидко залучати нових розробників;
- простота заміни однієї реалізації сервісу іншою;
- простота додавання нової функціональності в систему;
- ефективне використання ресурсів;
- еластичність: вихід з ладу одного сервісу зазвичай не призводить до виходу з ладу всієї системи;
- сервіси організовані відносно бізнес логіки яку вони виконують;
- кожен сервіс незалежно від інших може бути реалізований за допомогою будь-якої мови програмування, СУБД, та ін.;

– архітектурно побудовані за симетричним принципом (виробник-споживач).

Приклад мікросервісної архітектури наведено на рис. 3.5.

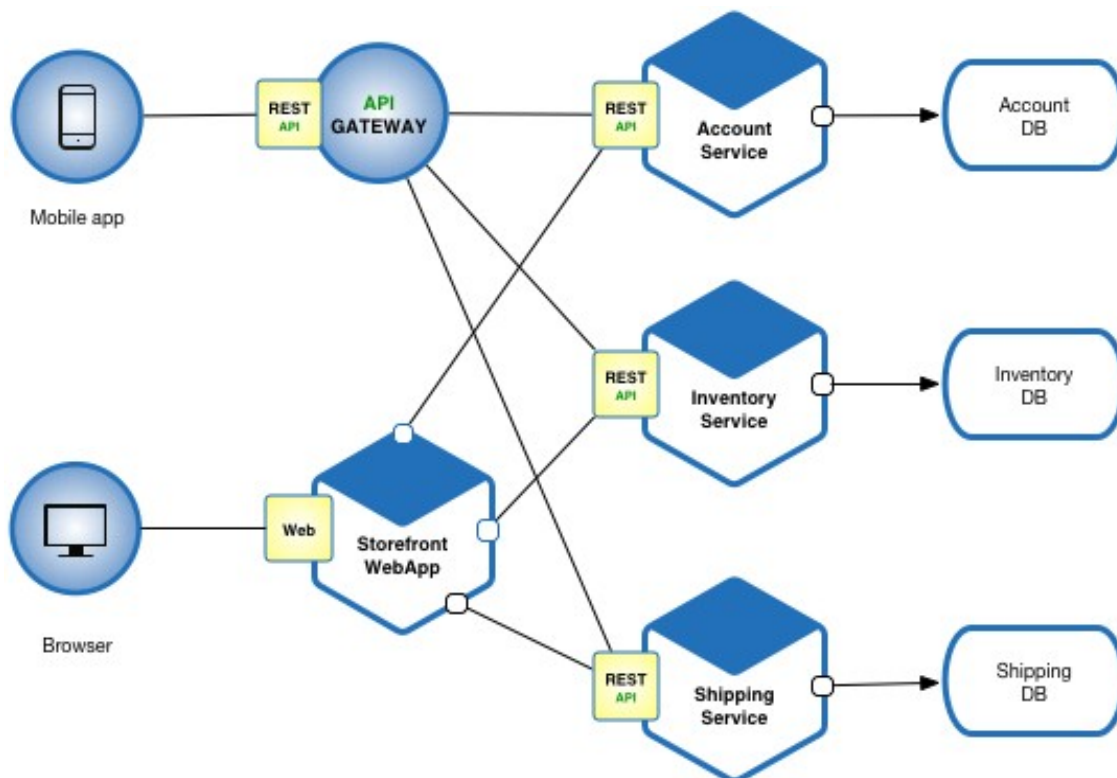


Рисунок 3.5 – Приклад схеми мікросервісної архітектури

Саме завдяки всім вищенаведеним особливостям було обрано саме мікросервісну архітектуру для проектування архітектури даного програмного продукту.

Схему проектування архітектури даної системи продемонстровано на рисунку 3.6.

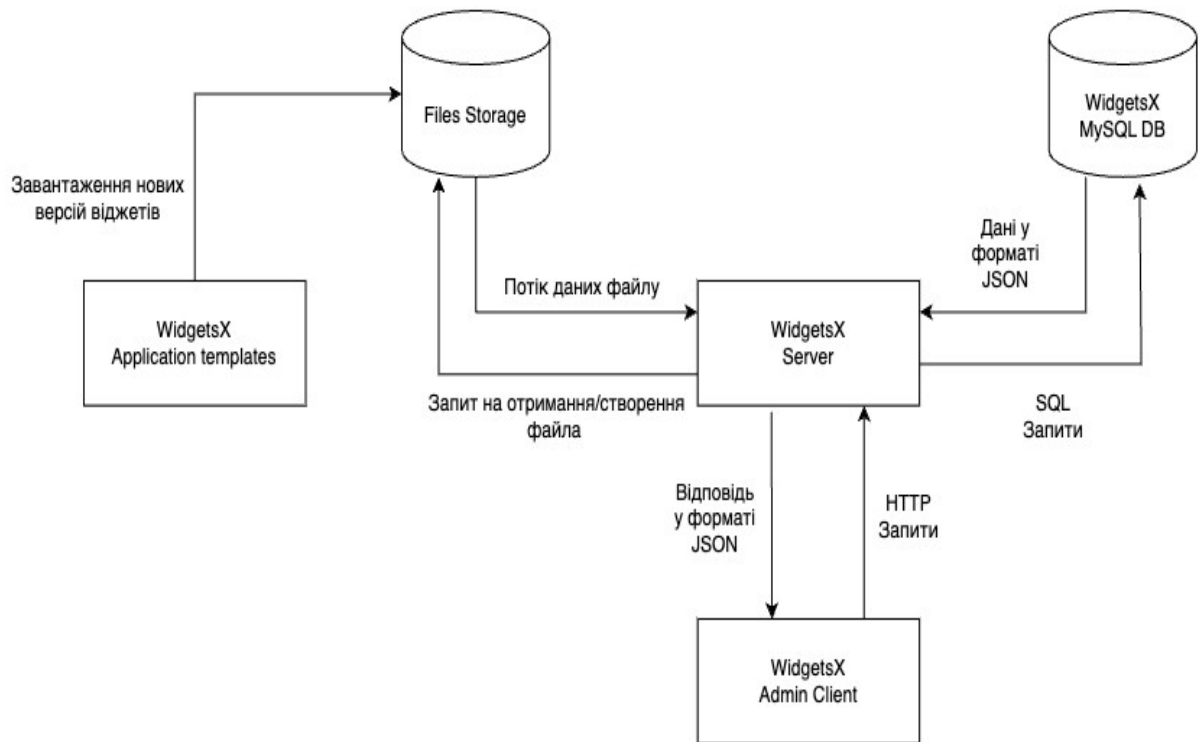


Рисунок 3.6 – Архітектура системи

3.3 Проектування графічного інтерфейсу

Наступним кроком є розробка графічного інтерфейсу програми. При розробці програмного забезпечення для створення віджетів були враховані наступні функціональні вимоги до програми:

- авторизація та реєстрація користувачів програми для створення віджетів;
- створення нових віджетів;
- перегляд створених віджетів;
- можливість редагувати створені віджети.

Сторінку авторизації у програмі для створення віджетів зображено на рис. 3.7. Тут користувач повинен авторизуватися або створити новий акаунт у додатку, аби розпочати роботу з додатком.

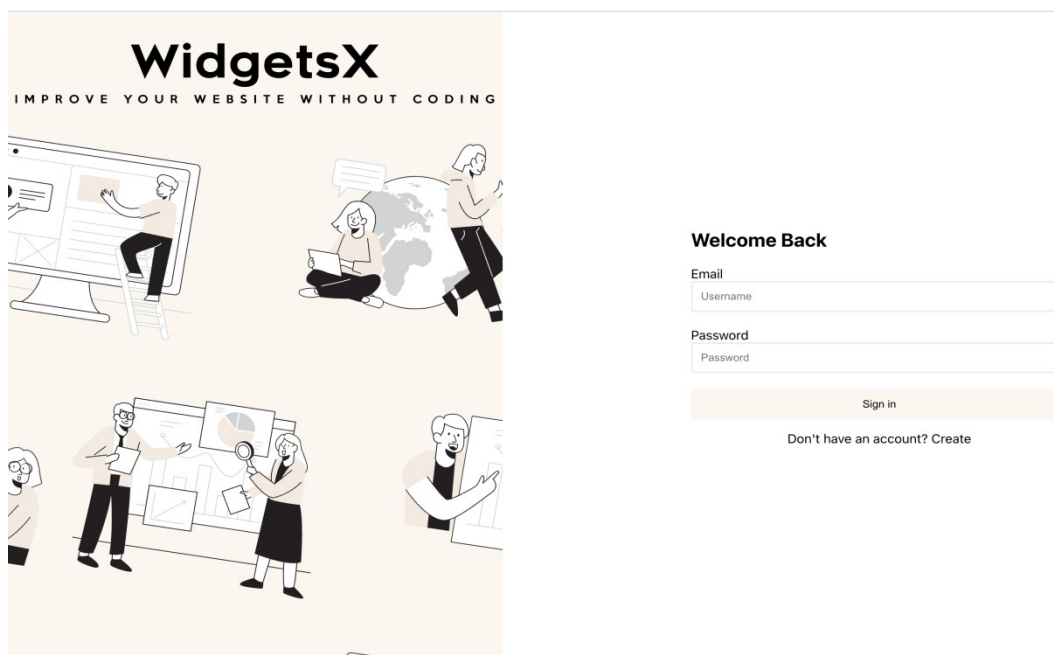


Рисунок 3.7 – Сторінка авторизації у програмі для створення віджетів

Після авторизації або реєстрації користувач потрапляє на головну сторінку додатку. На цій сторінці він може створити нові віджети, переглянути їх або редагувати, або підключити сторонні застосунки. Інтерфейс головної сторінки продемонстровано на рис. 3.8.

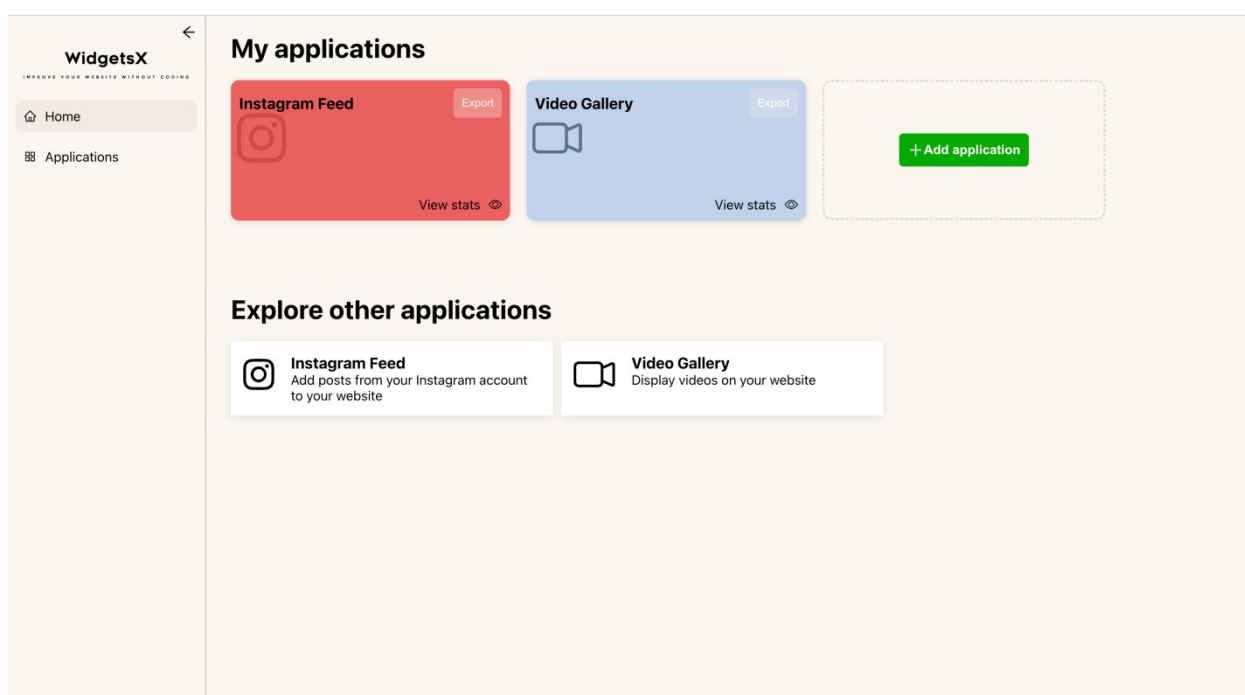


Рисунок 3.8 – Головна сторінка програми для створення віджетів

Редагувати існуючий віджет можна після переходу на його сторінку с головної. Існує тип віджету під назвою «Instagram Галерея».

Сторінка редагування віджету типу «Instagram Галерея» наведена на рис. 3.9.

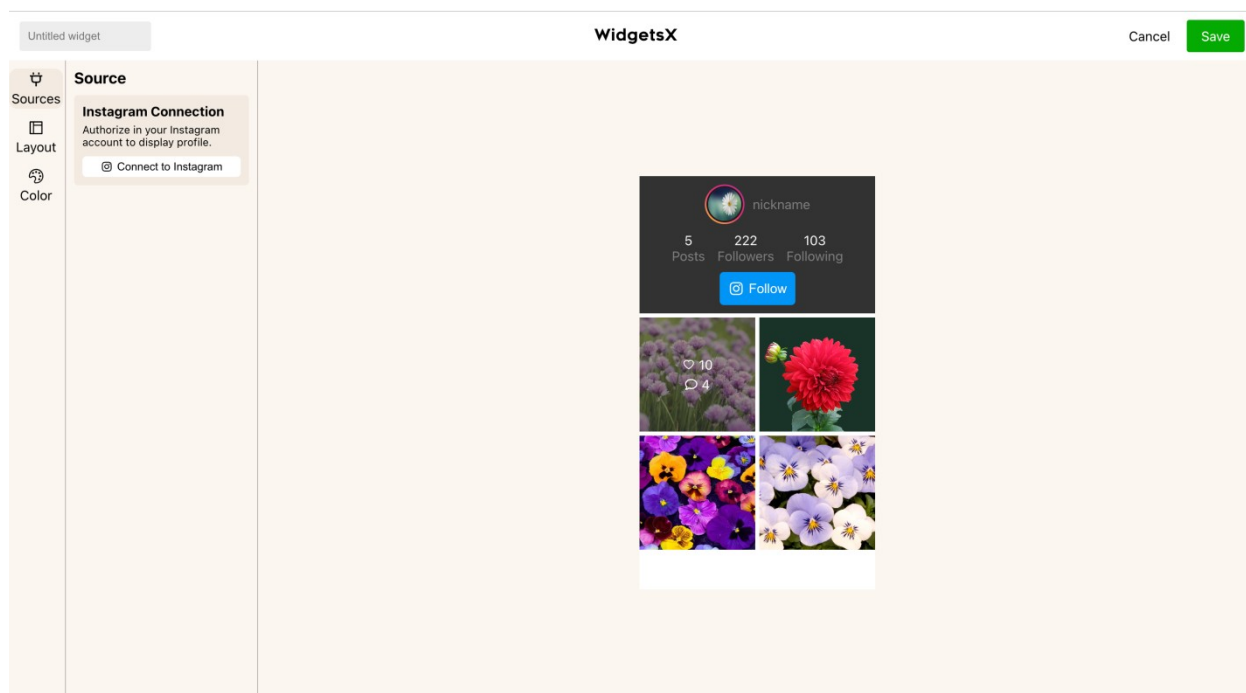


Рисунок 3.9 – Сторінка редагування віджету типу «Instagram Галерея»

Для створення деяких типів віджетів необхідна можливість надати доступ до даних зі сторонніх додатків. Для цього було прийнято рішення використовувати стандарт «відкритої авторизації» OAuth.

OAuth 2 або «Відкрита авторизація» – це стандарт, розроблений для надання сайту або додатку доступу до ресурсів, розміщених іншими веб-додатками. OAuth 2.0 є протоколом авторизації і служить засобом надання доступу до набору ресурсів (віддалені API або даними користувача).

В якості підтвердження наданого доступу використовується токени у форматі JSON Web Token (JWT), які можуть мати обмежений термін дії.

Для оновлень прав доступу використовується токен оновлення, за допомогою якого можна запитувати оновлений токен доступу.

Також можна створити віджет у відео-форматі. Сторінку створення відео-віджету продемонстровано на рис. 3.10.

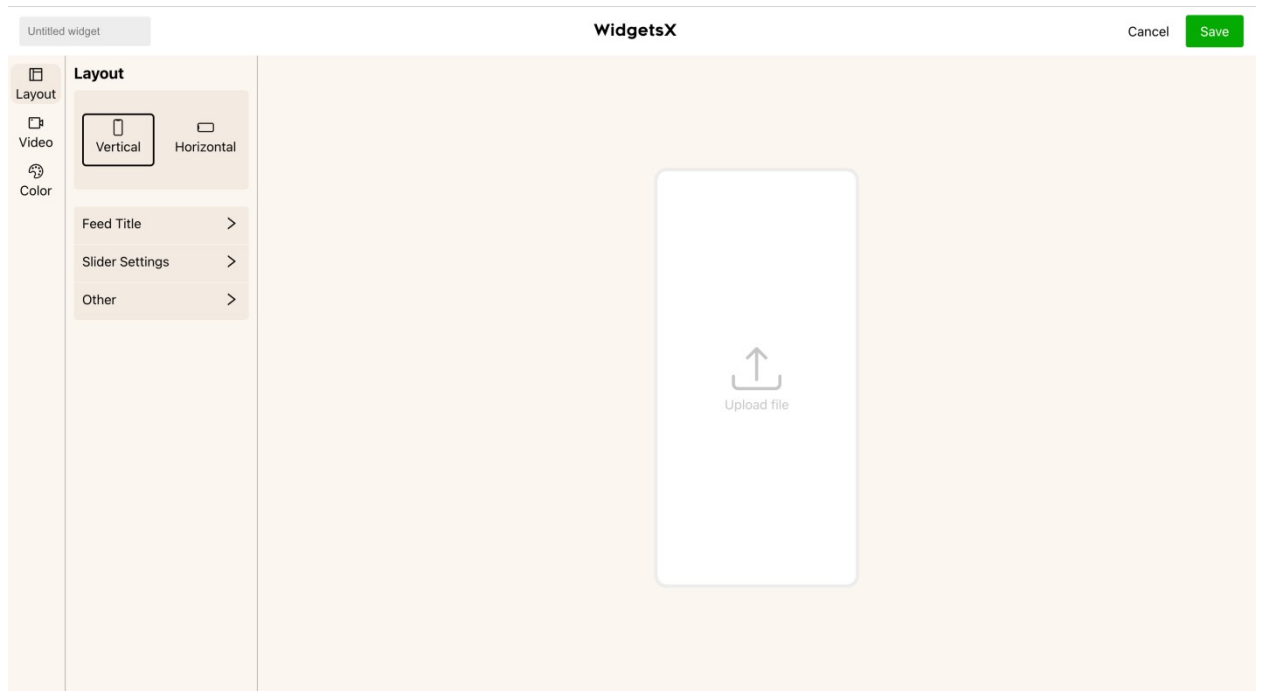


Рисунок 3.10 – Сторінка створення відео-віджету

Таким чином був розроблений інтерфейс даної програми, базуючись на основних функціональних вимогах до програмного забезпечення.

3.4 Проектування архітектури бази даних

Наступним не менш важливим кроком стало проектування архітектури бази даних MySQL, тому як велика кількість даних повинна збиратись, зберігатись та змінюватись.

Наявність бази даних облегшує пошук та керування даними, спрощує та пришвидшує їх обробку.

Структуру бази даних програми продемонстровано на рисунку 3.11.

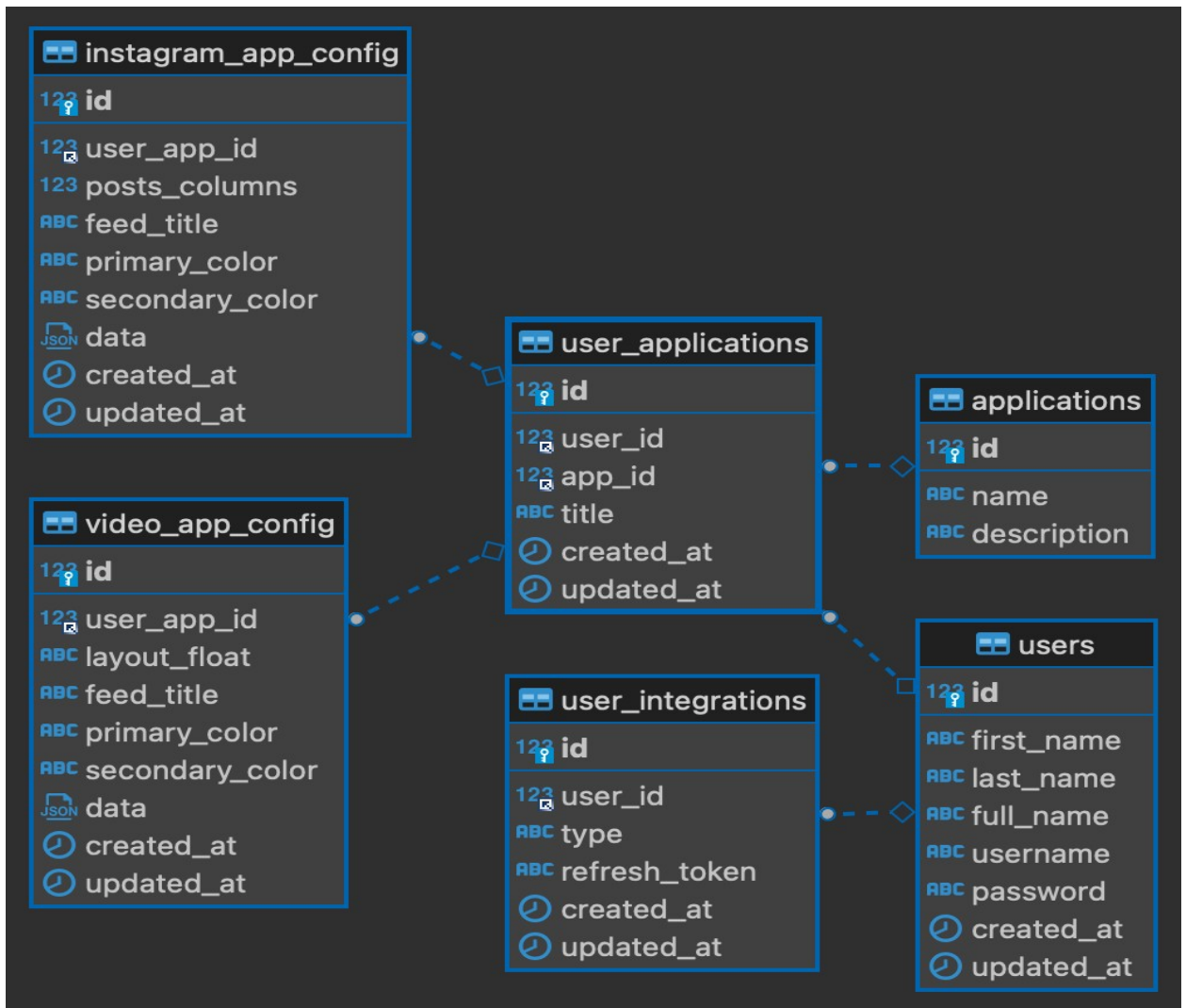


Рисунок 3.11 – База даних програми

База даних містить такі таблиці:

- користувачі (users) – містить публічну інформацію користувача і облікові дані для входу в систему;
- інтеграції користувача із сервісами "третьої сторони" (user_integrations), дані для отримання OAuth токена користувача необхідні для авторизації та надання прав доступу до даних у додатках "третьої сторони";
- програми (applications) – містить в собі список доступних типів віджетів;
- програми користувача (user_applications) – створені програми(віджети) користувача;

- конфігурація та дані програми типу "Відео" (video_app_config), вибрані параметри кастомізації основного/вторинного кольору, позиції відео плеєра (горизонтальна, вертикальна), опціональний заголовок, дані відео у форматі JSON;

- конфігурація та дані програми типу "Інстаграм галерея" (instagram_app_config), вибрані параметри кастомізації основного/вторинного кольору, кількість колонок публікацій, опціональний заголовок, дані профілю користувача та публікацій у форматі JSON.

Також база даних має такі зв'язки між таблицями:

- таблиця "Користувачі" (users) пов'язана з таблицями "Додатки користувача" (user_applications) та "Інтеграціями користувача" (user_integrations) полями "user_id";

- таблиця "Програми користувача", крім вищезгаданого зв'язку з таблицею користувачів, має зв'язки з таблицями "Конфігурація програми відео" та "Конфігурація програми Інстаграм галереї" за допомогою полів "user_app_id", також таблиця має зв'язок з таблицею "Додатки" полем "app_id".

3.5 Висновки до розділу 3

Для розроблюваного програмного продукту в даному розділі було обрано стек технологій та програмних засобів для розробки, спроектовано загальну архітектуру системи, засновану на архітектурному стилі Мікросервісів, а також графічний інтерфейс програми.

Розроблено та створену базу даних для зберігання та керування даними програмного продукту.

4 КЕРІВНИЦТВО З ВИКОРИСТАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Призначення та умови використання програмного продукту

Даний програмний продукт представляє собою систему, яка призначена для користувачів, яким необхідно створювати власні віджети.

Програмний продукт написано на мові програмування JavaScript з використанням TypeScript, що дозволяє швидко та ефективно створювати вебзастосунки, а також з використанням бази даних MySQL для створення, зберігання та швидкого керування великою кількістю даних.

Даний вебдодаток може використовуватись майже на будь-якому пристрої платформи MacOS або Windows(7,10), що має хоча б один сучасний веббраузер. Не потребує встановлення будь-яких інших компонентів.

4.2 Функції системи

Програмне забезпечення для створення віджетів забезпечує виконання таких функцій:

- авторизація та реєстрація користувачів програми для створення віджетів;
- створення нових віджетів;
- перегляд створених віджетів;
- можливість редагувати створені віджети.

4.3 Вхідні та вихідні дані

Вхідними даними до програмного забезпечення для створення віджетів є особисті дані про користувача:

- ім'я;
- прізвище;

- ім'я користувача;
- пароль;
- адреса електронної пошти;
- особисті дані користувача зі сторонніх застосунків;
- файли, які користувач завантажує до програми.

Вихідними даними є оброблена інформація про користувача, а також створений віджет та посилання на нього для подальшого використання у програмному коді.

4.4 Взаємодія із програмним продуктом

Для того щоб розпочати роботу з програмою, користувач повинен зареєструватися. Користувачеві треба ввести своє ім'я, прізвище, ім'я користувача та придумати пароль для подальшої авторизації у вебзастосунку.

На рисунку 4.1 буде продемонстровано сторінку реєстрації.

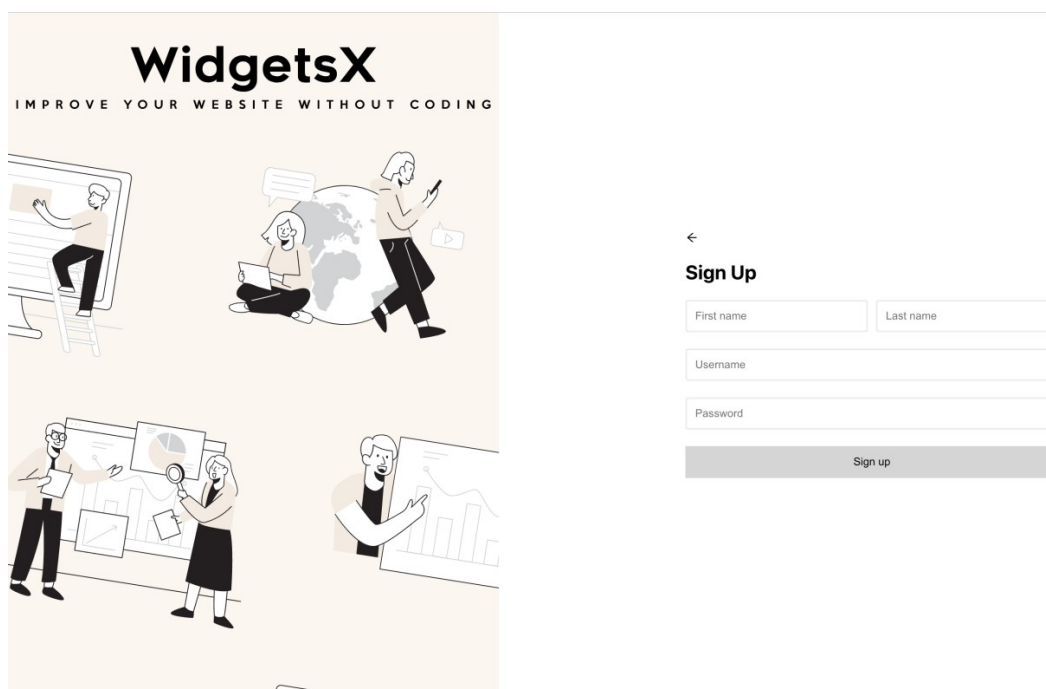


Рисунок 4.1 – Сторінка реєстрації у додатку

Якщо користувач вже має акаунт у за стосунку, то він має можливість авторизуватися. Для цього потрібно ввести все створене ім'я користувача та свій пароль. Сторінку авторизації буде наведено на рис. 4.2

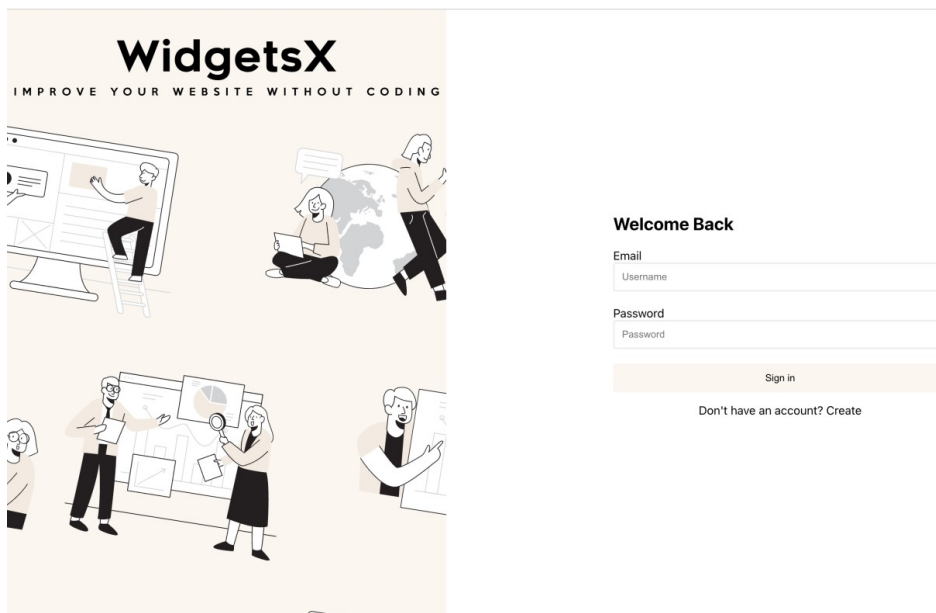


Рисунок 4.2 – Сторінка авторизації

Після авторизації або реєстрації користувач потрапляє на головну сторінку – сторінку «Мої програми». Інтерфейс сторінки буде наведено на рисунку 4.3.

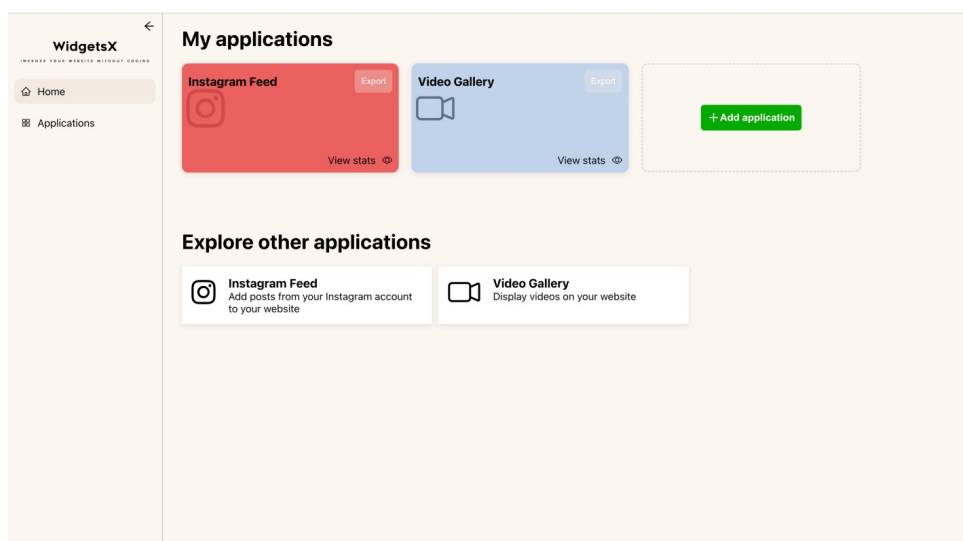


Рисунок 4.3 – Сторінка «Мої програми»

Натиснувши на поле «Instagram Feed» («Інстаграм Галерея»), користувач переходить на сторінку створення віджету «Інстаграм Галерея». Інтерфейс сторінки продемонстровано на рис. 4.4.

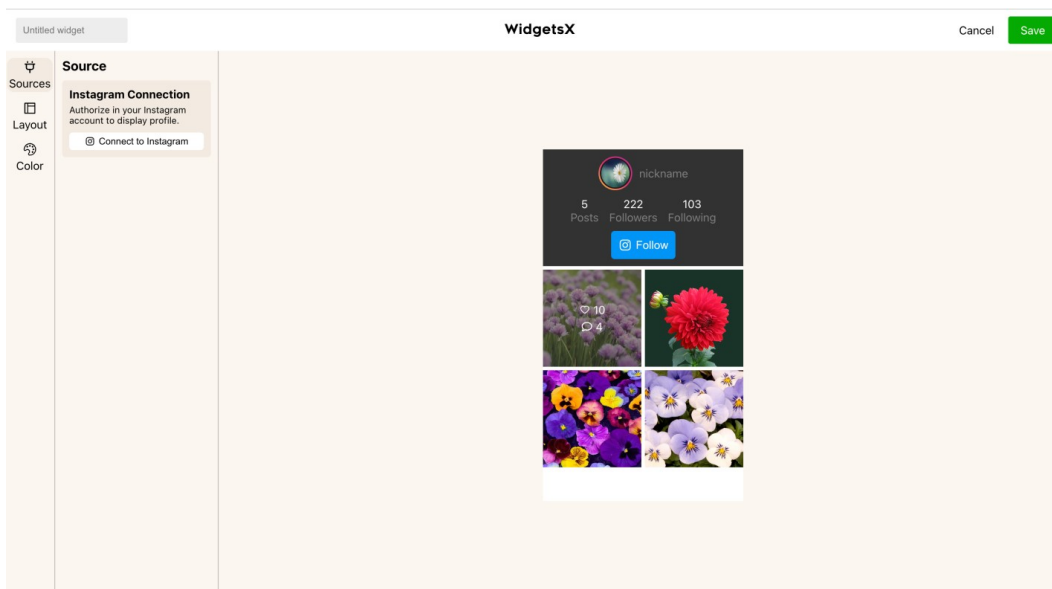


Рисунок 4.4 – Сторінка створення віджету «Інстаграм Галерея»

Натискання кнопки «Connect to Instagram», яка зображена на рисунку 4.5, відкриває вікно підтвердження підключення додатку до акаунту Instagram.

Після підтвердження відкривається модальне вікно авторизації у сторонньому додатку Instagram, для того щоб підключити свій акаунт та надати доступ до персональних даних для подальшої роботи зі створення віджету.

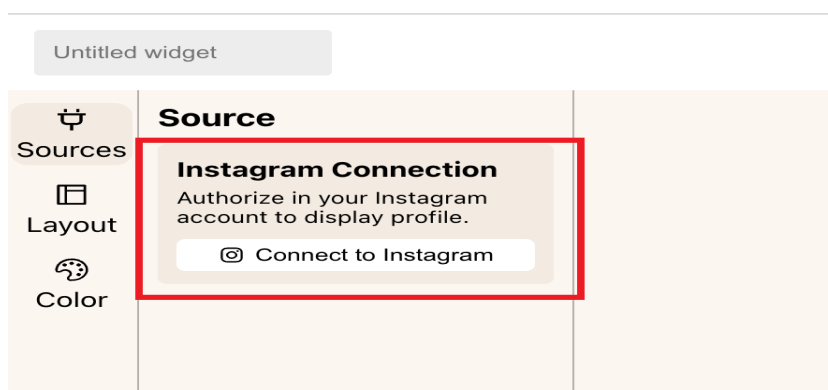


Рисунок 4.5 – Кнопка «Connect to Instagram»

Вікно підтвердження підключення зображено на рисунку 4.6.

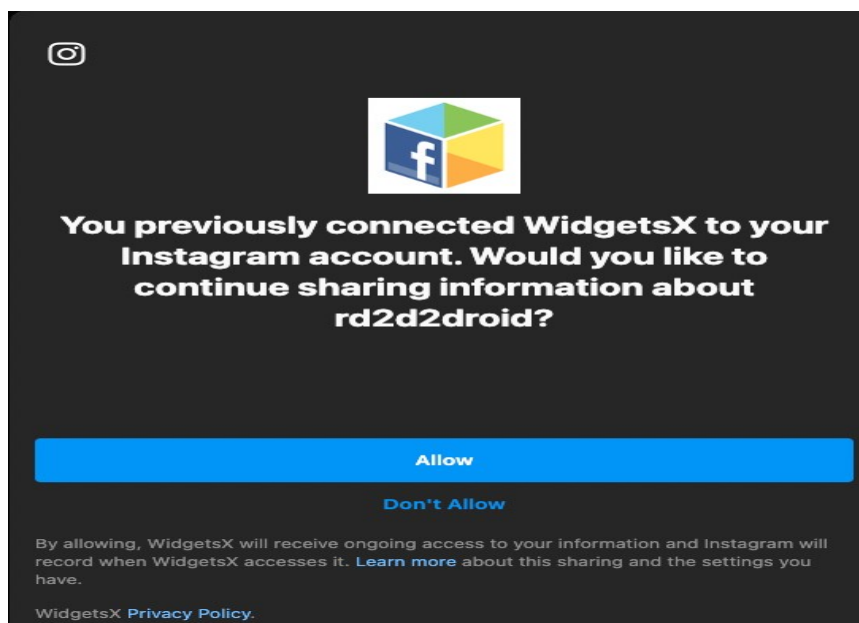


Рисунок 4.6 – Вікно підтвердження підключення до Instagram

Модальне вікно авторизації в соцмережі Instagram продемонстровано на рис. 4.7.

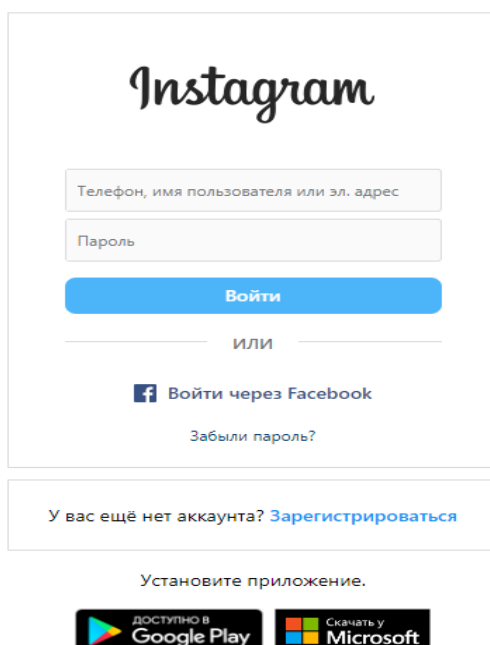


Рисунок 4.7 – Вікно авторизації у соцмережі Instagram

Якщо користувач авторизувався в Instagram, на екрані з'явиться повідомлення що його акаунт успішно підключено до застосунку. Вікно повідомлення продемонстровано на рисунку 4.8.

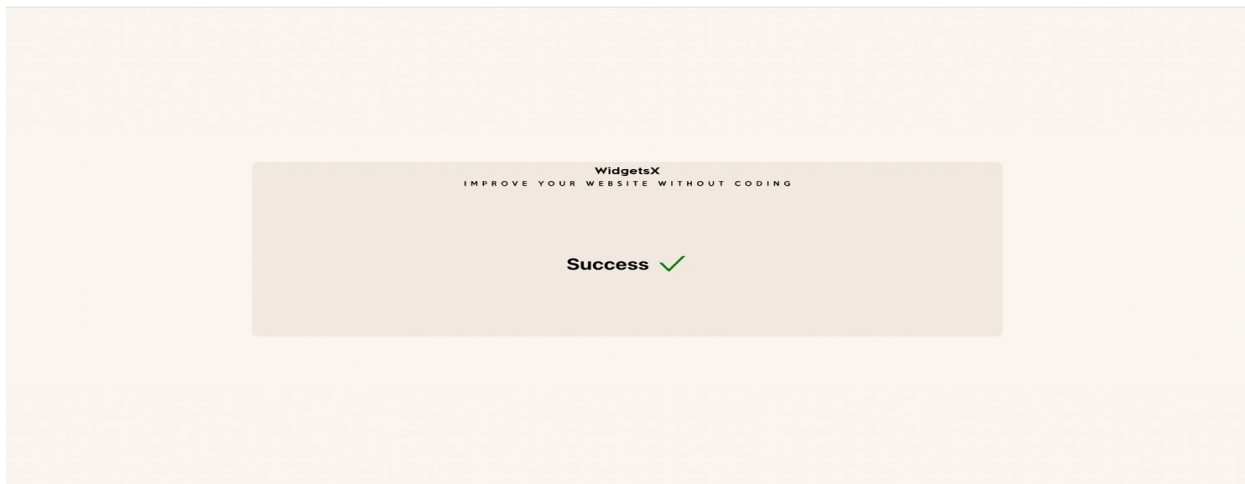


Рисунок 4.8 – Вікно повідомлення про успішне підключення

Після авторизації профіль Instagram інтегрується в додаток. Приклад інтегрованого профілю наведено на рисунку 4.9.

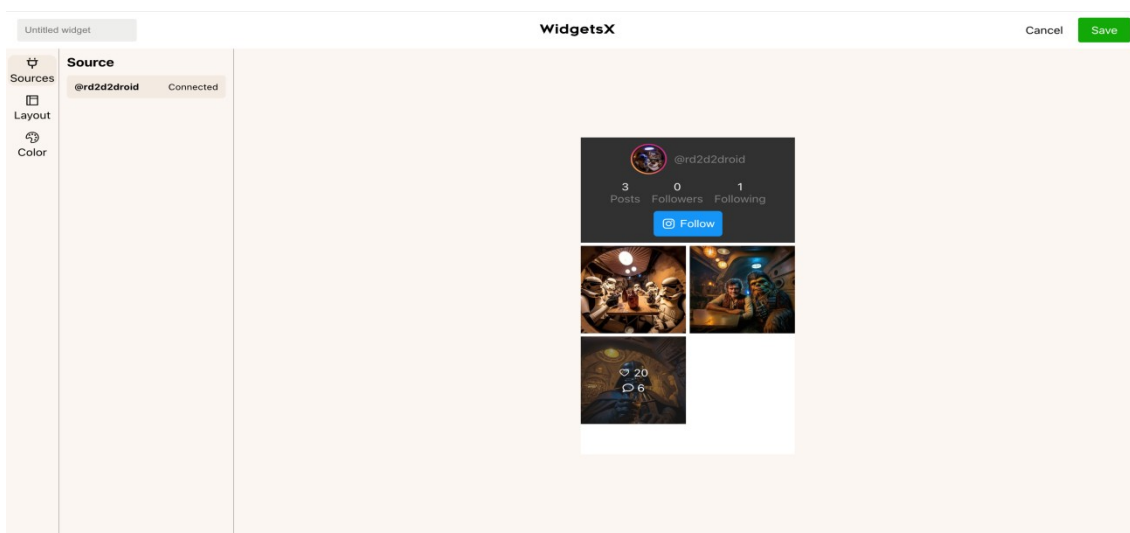


Рисунок 4.9 – Профіль Instagram інтегрований в додаток

Після натискання кнопки «Save» на екрані з'являється спливаюче повідомлення, що містить в собі скрипт.

Цей скрипт користувач повинен скопіювати для подальшої роботи: наприклад, додати його на свій сайт.

Інтерфейс спливаючого повідомлення наведено на рис. 4.10.

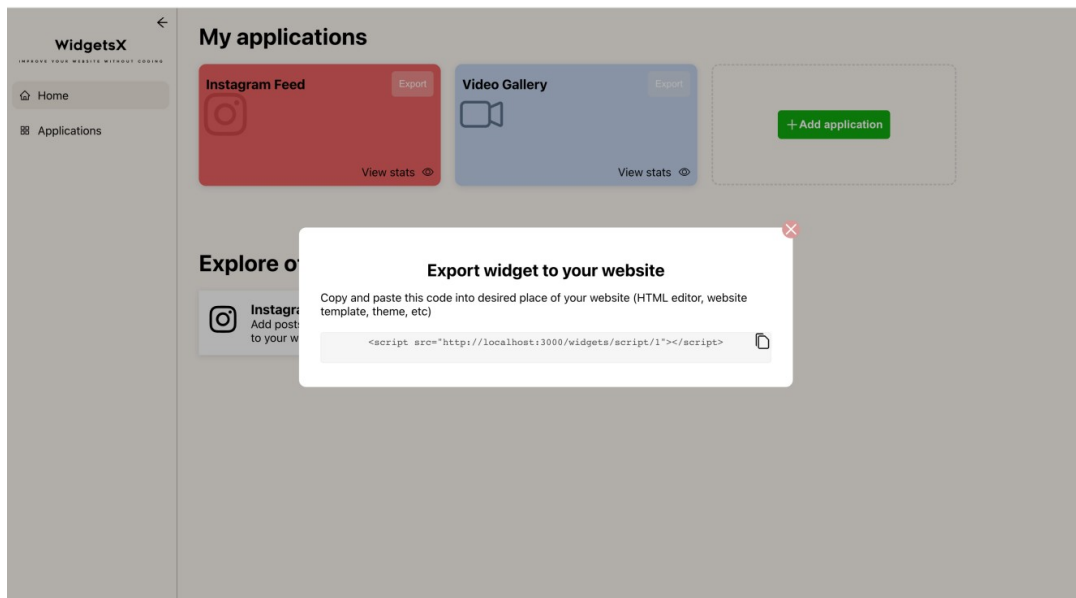


Рисунок 4.10 – Спливаюче повідомлення зі скриптом

Приклад сгенерованого програмою віджету на сайті продемонстровано на рис. 4.11.

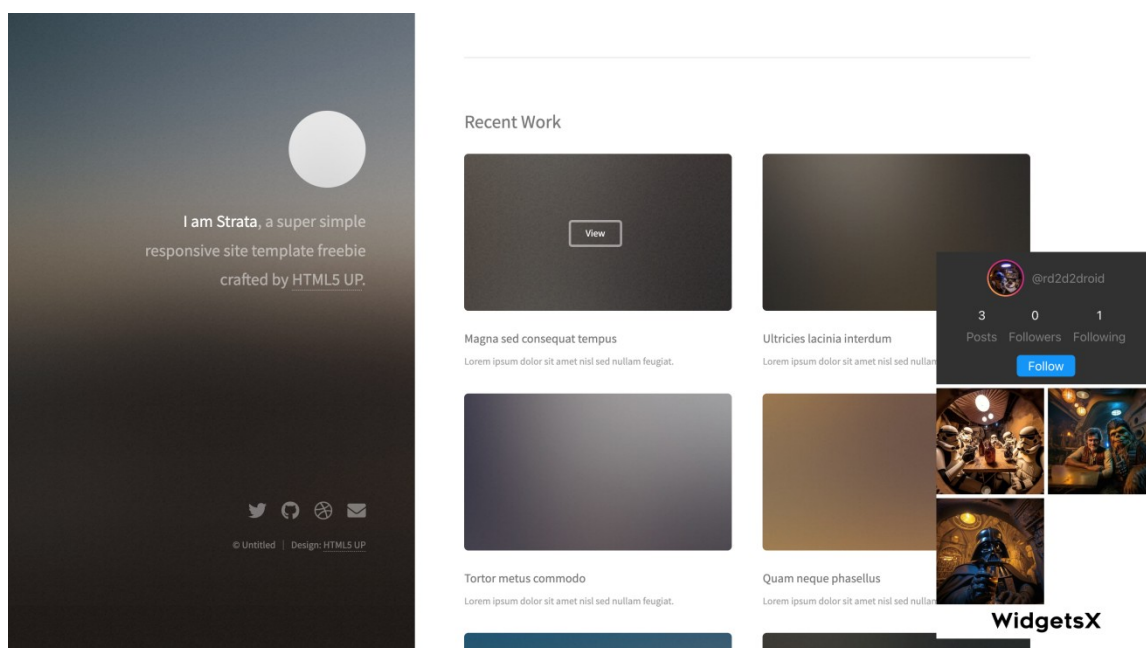


Рисунок 4.11 – Вигляд сгенерованого віджету на сайті

Натиснувши на поле «Video Gallery» (Відео галерея) головної сторінки, користувач переходить на сторінку створення відео-віджету. Тут сторінці користувач має змогу завантажити відео-файл у форматі .mp4. Інтерфейс сторінки відео-віджету наведено на рис. 4.12.

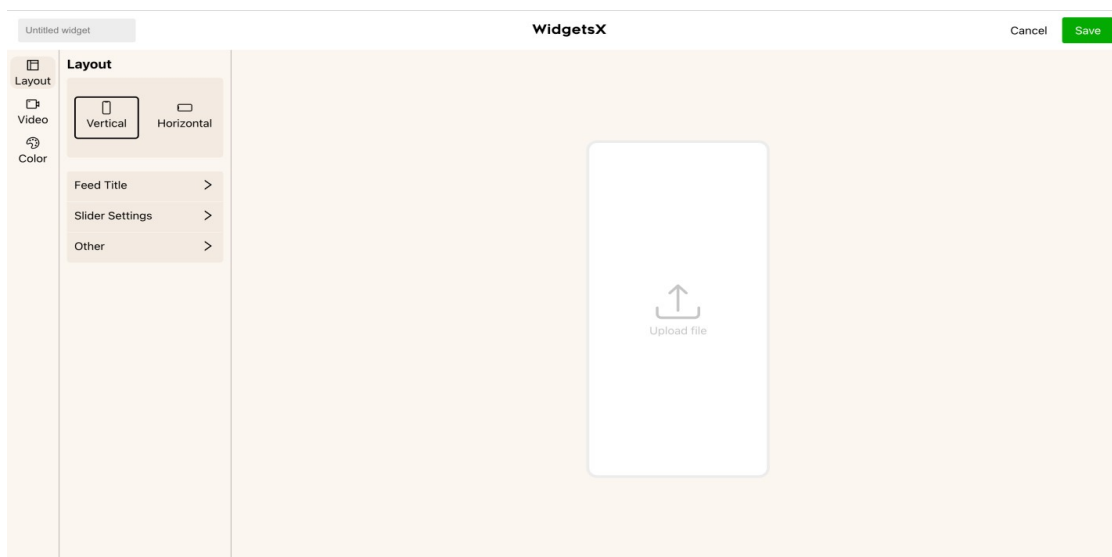


Рисунок 4.12 – Сторінка створення відео-віджету

Процес завантаження відео-файлу для створення відео-віджету зображено на рисунку 4.13.

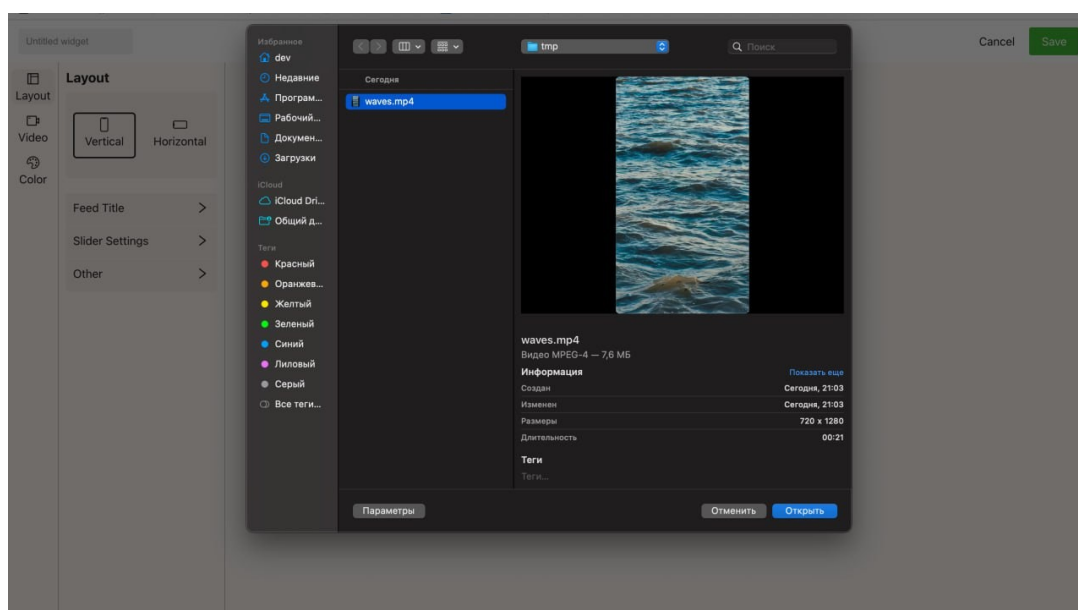


Рисунок 4.13 – Завантаження відео-файлу

Після успішного завантаження обраного файлу користувач має можливість побачити його на сторінці відео-віджету та провести певні маніпуляції з даним віджетом, наприклад, міняти положення самого відео з вертикального на горизонтальне та навпаки, додати написи та інше.

Завантажений відео-файл на сторінці створення зображено на рисунку 4.14.

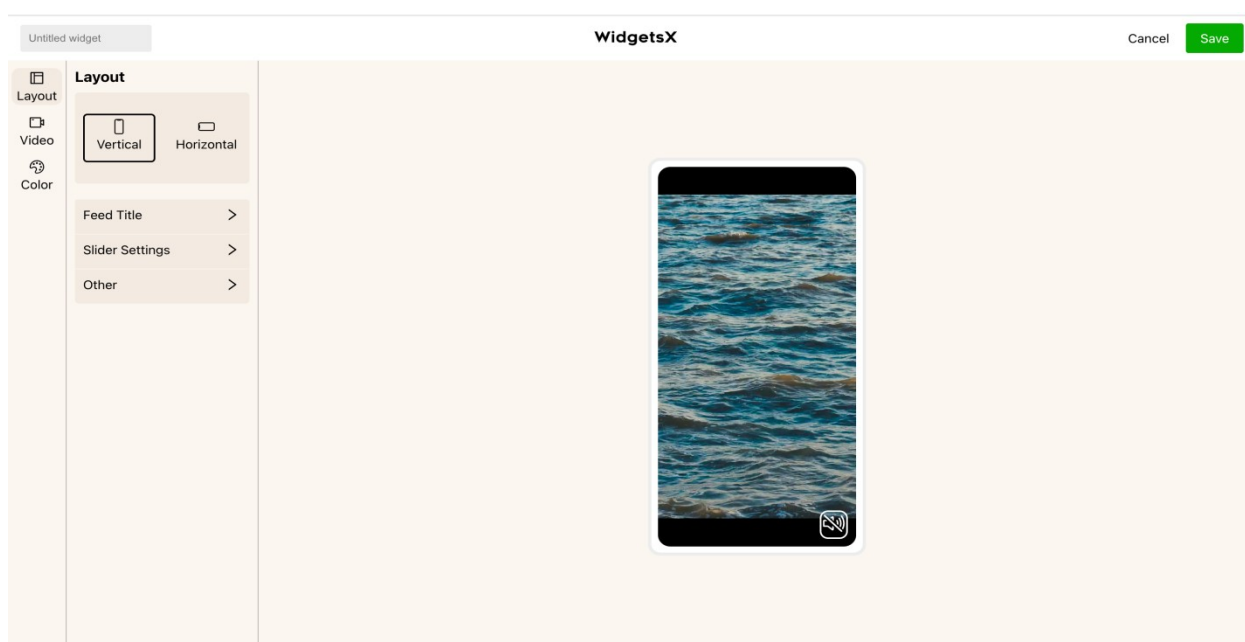


Рисунок 4.14 – Завантажений відео-файл на сторінці створення віджету

Після зберігання віджету користувач знову отримує спливаюче повідомлення з посиланням на створений віджет, яке він може надалі використовувати у своїх цілях. Наприклад, як продемонстровано на рисунку 4.15, додати відео на свій сайт.

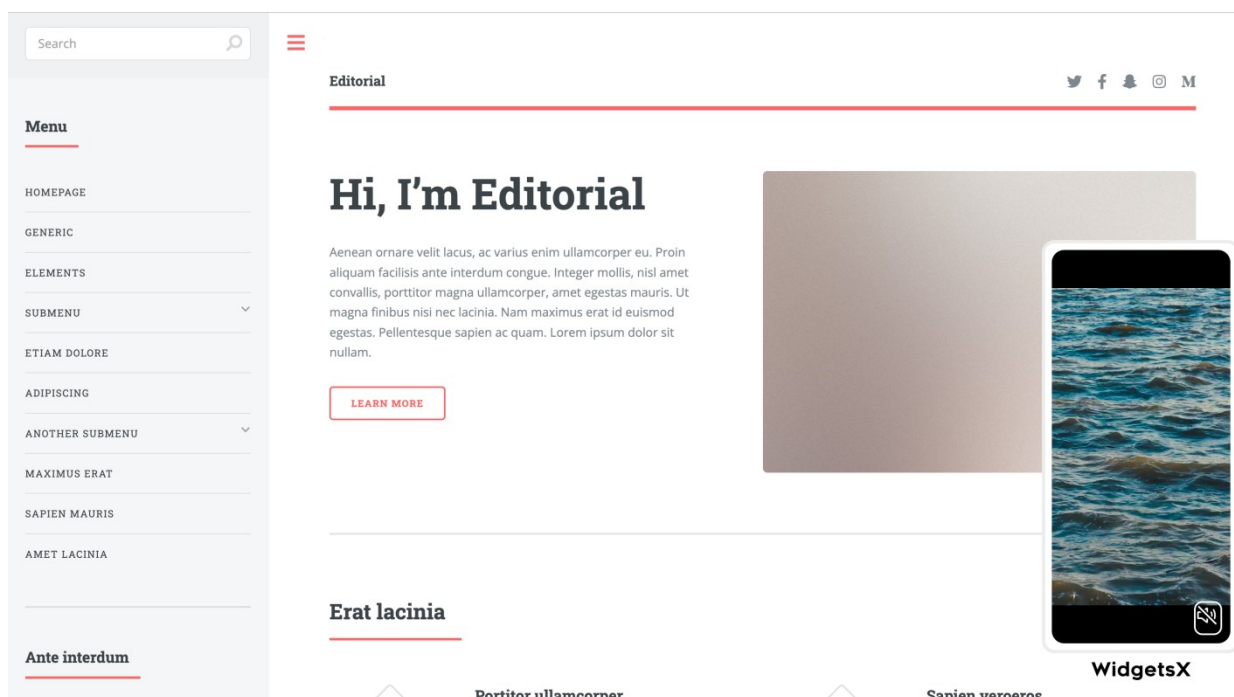


Рисунок 4.15 – Вигляд створеного відео-віджету на сайті

4.5 Висновки до розділу 4

У даному розділі наведено призначення програмного продукту, а також умови використання продукту, його вхідні та вихідні дані. Наведено опис основних функцій системи та надано детальну інструкцію користувача щодо використання даного програмного забезпечення.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для створення віджетів.

Проаналізовано існуючі програмні засоби для створення віджетів, оцінені їх можливості, їх переваги та недоліки, а також їх доступність для користувачів. Було виявлено, що більшість існуючих застосунків відповідають вимогам багатьох користувачів.

Також було проведено порівняльний аналіз та обґрунтування використання інструментів для розробки подібних додатків, в результаті чого мовою для розробки додатку для створення віджетів було обрано JavaScript з використанням TypeScript. Проведені дослідження були зведені у порівняльних таблицях.

Для розроблюваного програмного продукту було спроектовано загальну архітектуру системи та графічний інтерфейс. Також було спроектовано архітектуру бази даних для зручного та швидкого керування даними.

Наведено призначення системи, умови використання розробленого програмного продукту, його вхідні та вихідні дані, а також його можливості. Також було наведено основні функції системи. Крім того, надається детальна інструкція для користувача щодо використання програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Bootstrap? [Electronic resource]. – Access mode : <https://getbootstrap.com/>.
2. What is Javascript? [Electronic resource]. – Access mode https://developer.mozilla.org/enUS/docs/Learn/JavaScript/First_steps/What_is_JavaScript.
3. What is HTML5 Canvas? [Electronic resource]. – Access mode : https://www.w3schools.com/html/html5_canvas.asp.
4. What is the widget? [Electronic resource]. – Access mode : <https://www.techtarget.com/whatis/definition/widget>.
5. Salesforce: The Customer Company [Electronic resource]. – Access mode : <https://www.salesforce.com/>.
6. IBM Company [Electronic resource]. – Access mode : <https://www.ibm.com/us-en/>.
7. Why Visual Studio Code? [Electronic resource]. – Access mode <https://code.visualstudio.com/docs/editor/whyvscode>.
8. Git [Electronic resource]. – Access mode : <https://git-scm.com/>.
9. Atom vs VSCode [Electronic resource]. – Access mode : <https://www.crazyegg.com/blog/atom-vs-visual-studio-code/>.
10. Atom vs WebStorm [Electronic resource]. – Access mode : <https://stackshare.io/stackups/atom-vs-webstorm>.
11. Top 5 Reasons You Should Use Chrome [Electronic resource]. – Access mode : <https://www.ditoweb.com/2014/02/top-5-reasons-you-should-use-google/#:~:text=Chrome%20is%20designed%20to%20be,WebKit%20open%20source%20rendering%20engine>.
12. Google. American Company [Electronic resource]. – Access mode : <https://www.britannica.com/topic/Google-Inc>.
13. Most Popular Web Browsers In 2023 [Electronic resource]. – Access mode: <https://www.oberlo.com/statistics/browser-market-share#:~:text=As%20of>

%20February%202023%2C%20Google's,%25%2C%2046.88%20percentage
%20points%20behind.

14. CrossBrowser Testing [Electronic resource]. – Access mode :
<https://www.browserstack.com/cross-browser-testing>

15. What is API [Electronic resource]. – Access mode :
https://aws.amazon.com/what-is/api/?nc1=h_ls

16. TypeScript [Electronic resource]. – Access mode :
<https://www.typescriptlang.org/>

17. JSON Web Token [Electronic resource]. – Access mode:
<https://jwt.io/>.

18. MicroService Architecture [Electronic resource]. – Access mode:
[https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/
microservices](https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices)

ДОДАТОК А
Текст програми

```

import './root/index.css';
import ReactDOM from 'react-dom/client';
import App from './root/App';
import { BrowserRouter } from 'react-router-dom';
import { AuthorizationWrapper } from
'./components/Authorization/AuthorizationWrapper';
import { Provider } from 'react-redux';
import { store } from './state/store';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <BrowserRouter basename='WidgetsX'>
    <AuthorizationWrapper>
      <Provider store={store}>
        <App/>
      </Provider>
    </AuthorizationWrapper>
  </BrowserRouter>
);
import styles from './App.module.css';
import { AppRoutes } from 'src/components/General/AppRoutes';
import { NavMenu } from '../components/Menu/NavMenu';

function App() {
  return (
    <>
      <div className={styles.container}>
        <NavMenu />
        <AppRoutes />
      </div>
    </>
  );
}

export default App;

import axios, { AxiosInstance } from 'axios'
export class APIClient {
  private instance: AxiosInstance = null

```

```

    constructor(protected baseUrl: string, private token?: string){
        this.instance = axios.create({
            baseUrl: this.baseUrl,
            headers: this.token
                ? { Authorization: `Bearer ${this.token}` }
                : {}
        })
    }

    protected get<T>(url: string, params?: any){
        return this.instance.get<T>(url, { params })
    }

    protected post<T>(url: string, body?: any){
        return this.instance.post<T>(url, body)
    }

    protected put<T>(url: string, body?: any){
        return this.instance.put<T>(url, body)
    }

    protected delete<T>(url: string){
        return this.instance.delete<T>(url)
    }
}

import { APIClient } from "src/lib/utils/api.client";
import { IGenerateAccessTokenResponse } from "../interfaces/instagram-api.interface";

class InstagramAPI extends APIClient {
    constructor(baseUrl: string, token: string){
        super(baseUrl, token)
    }

    public async getAuthorizeLink(): Promise<string>{
        return this.get<{link: string}>('/integrations/instagram/authorization-link')
            .then((d) => d.data.link)
    }
}

```

```

    public async generateAccessToken(code: string) {
        return
this.get<IGenerateAccessTokenResponse>('/integrations/instagram/exchange-
code', { code })
        .then((d) => d.data)
        .catch((e) => {
            throw new Error(e)
        })
    }

    public async getUserMediaData(profileID: string){
        return this.get<any>('/integrations/instagram/profile-data',
{ external_user_id: profileID })
        .then((d) => d.data)
    }
}

let _client: InstagramAPI = null

export function getInstagramAPIClient(token: string){
    if(!_client){
        _client = new InstagramAPI(
            process.env.REACT_APP_API_URL,
            token
        )
    }

    return _client
}

import { APIClient } from "src/lib/utils/api.client"
import { ICheckUsernameExistsRes, ISubmitUserDataRes, IUser, IUserCreds,
IValidateAccessTokenRes } from "../interfaces/user-auth-api.interfaces"

class UserAuthAPI extends APIClient {
    constructor(baseUrl: string){
        super(baseUrl)
    }

    public async validateAccessToken(token: string){

```

```

        return
this.get<IValidateAccessTokenRes>('/user-authorization/validate/jwt', { token
})
    .then((res) => res.data)
}
public async checkUsernameExistis(name: string){
    return this.get<ICheckUsernameExistsRes>('/user-authorization/check-
username', { name })
    .then((res) => res.data)
}

public async submitUserCreds(payload: IUserCreds){
    return this.post<ISubmitUserDataRes>('/user-authorization/sign-in',
payload)
    .then((res) => res.data)
}

public async submitNewUser(payload: Pick<IUser, "first_name" | "last_name"
| "username" | "password">){
    return this.post<ISubmitUserDataRes>('/user-authorization/sign-up',
payload)
    .then((res) => res.data)
}
}

let _client: UserAuthAPI = null

export function getUserAuthAPIClient(){
    if(!_client){
        _client = new UserAuthAPI(process.env.REACT_APP_API_URL)
    }

    return _client
}

export interface IGenerateAccessTokenResponse {
    id: number
    user_id: number
    external_user_id: string
    type: string
    access_token: string

```

```

    refresh_token: string
    token_renewed_at: string
    created_at: string
    updated_at: string
  }
export interface IUser {
  id: number
  first_name: string
  last_name: string
  full_name: string
  username: string
  password: string
  created_at: string
  updated_at: string
}

export interface IUserCreds {
  username: string;
  password: string;
}

export interface ICheckUsernameExistsRes {
  exist: boolean;
}

export interface IValidateAccessTokenRes {
  valid: boolean;
  error: string;
  token: string;
}

export interface ISubmitUserDataRes {
  user: IUser;
  token: string;
}

import styles from './NavMenu.module.css'
import { AiOutlineArrowLeft } from 'react-icons/ai'
import { BrandLogo } from '../General/BrandLogo'
import { Link, useLocation } from 'react-router-dom'
import { routesDefinition } from '../General/AppRoutes'

```

```

import { useState } from 'react'
import { useAppSelector } from 'src/lib/hooks/redux'

export function NavMenu(){
  const hiddeNavMenu = useAppSelector((s) => s.appLayout.hiddeNavMenu)
  const [ showNavMenu, setShowNavMenu ] = useState(true)
  const { pathname: currPath } = useLocation()
  return (
    !hiddeNavMenu &&
    <div className={` ${styles.container} ${!showNavMenu ?
styles.container_hidden : ''}`>
      <div
        className={` ${styles.nav_toggle} ${ !showNavMenu ?
styles.nav_toggle_off : '' }`}
        onClick={() => setShowNavMenu(!showNavMenu)}
      >
        <AiOutlineArrowLeft size='20px' />
      </div>
      {showNavMenu
        ? <BrandLogo logoWidth='100px' taglineWidth='200px' />
        : <></>
      }
      <ul className={styles.nav_container}>
        {routesDefinition
          .filter((def) => !def.hidden)
          .map((def, i) =>
            <li key={i}>
              <Link to={def.path} style={{textDecoration: 'none', color:
'black'}}>
                <div
                  className={
                    `${styles.nav_item} ${currPath === def.path ?
styles.nav_item_active : '' }`
                  }
                >
                  {def.icon}
                  {showNavMenu ? def.name : ''}
                </div>
              </Link>
            </li>
          )
        )
      }
    )
  )

```

```

    }
  </ul>
</div>
)
}

import style from './Authorization.module.css'
import _ from 'lodash'
import { useEffect, useState } from 'react'
import { getUserAuthAPIClient } from 'src/lib/services/api/user-auth-api.client'
import { setUsernameExists } from 'src/lib/hooks/check-username'
import { IAuthorizationProps } from './Authorization'

interface ISignInProps extends IAuthorizationProps {
  hidde: boolean
}

export const SignIn = ({ hidde, setToken }: ISignInProps) => {
  const [login, setLogin] = useState<string>(null)
  const [password, setPassword] = useState<string>(null)
  const [isInputWrong, setWrongInput] = useState(false)

  const userAuthAPIClient = getUserAuthAPIClient()
  const { username, exists } = setUsernameExists(login)

  const submitUserCreds = async() => {
    if(!_.isEmpty(login) && !_.isEmpty(password)){
      try {
        const { token } = await userAuthAPIClient.submitUserCreds({ username:
login, password })
        setToken(token)
      } catch (error) {
        window.alert(_.get(error, 'data.message') ?? _.get(error, 'message'))
      }
    }
  }

  useEffect(() => {
    if(_.isEmpty(username)){
      setWrongInput(false)
    }
  })
}

```

```

    return
  }

  setWrongInput(!exists)
}, [username, exists])

if(hidde){
  return <></>
}
return (
  <>
    <div className={style.signin_container}>
      <div className={style.signin_container_items}>
        <h2>Welcome Back</h2>
        <div>
          <p>Email</p>
          <input
            type='text'
            className={style.signin_container_items_input, isInputWrong ?
style.err_input : ''}.join(' ')}
            onChange={(e) => setLogin(e.target.value)}
            placeholder='Username'
          />
        </div>
        <div>
          <p>Password</p>
          <input
            type='password'
            onChange={(e) => setPassword(e.target.value)}
            className={style.signin_container_items_input}
            placeholder='Password'
          />
        </div>
        <button
          onClick={submitUserCreds}
          className={style.authorization_button, isInputWrong ?
style.button_disabled : '' }.join(' ')}
        >
          Sign in
        </button>
      </div>
    </>
  )

```

```

        </div>
    </>
)
}
/* eslint-disable react-hooks/exhaustive-deps */
import style from './Authorization.module.css'
import _ from 'lodash'
import { Dispatch, useEffect, useState } from 'react'
import { useUsernameExists } from 'src/lib/hooks/check-username'
import { getUserAuthAPIClient } from 'src/lib/services/api/user-auth-api.client'
import { AiOutlineArrowLeft } from 'react-icons/ai'
import { IAuthorizationProps } from './Authorization'

interface ISignUpProps extends IAuthorizationProps {
    hidde: boolean;
    setShowSignUpForm: Dispatch<React.SetStateAction<boolean>>;
}

export const SignUp = ({ hidde, setShowSignUpForm, setToken }: ISignUpProps)
=> {
    const [username, setUsername] = useState<string>(null)
    const [firstName, setFirstName] = useState<string>(null)
    const [lastName, setLastName] = useState<string>(null)
    const [password, setPassword] = useState<string>(null)

    const [isInputWrong, setWrongInput] = useState(false)
    const [allowToSubmit, setAllowToSubmit] = useState(false)

    const { username: debouncedUsername, exists } = useUsernameExists(username)

    const submitNewUser = async() => {
        if(!allowToSubmit){
            return
        }

        try {
            const { token } = await getUserAuthAPIClient()
                .submitNewUser({
                    username,
                    password: password,

```

```

        first_name: firstName,
        last_name: lastName,
    })

    setToken(token)
  } catch (error) {
    window.alert(_.get(error, 'data.message') ?? _.get(error, 'message'))
  }
}

useEffect(() => {
  if(_.isEmpty(username)){
    setWrongInput(false)
    return
  }

  setWrongInput(exists)
}, [debouncedUsername, exists])

useEffect(() => {
  const isFormCompleted = !_.isEmpty(debouncedUsername) && !exists && !
_.isEmpty(password)

  setAllowToSubmit(isFormCompleted)
}, [debouncedUsername, password, exists])

if(hidde){
  return <></>
}

return (
  <>
    <div className={style.signup_container}>
      <div className={style.signup_container_items}>
        <AiOutlineArrowLeft style={{cursor: 'pointer'}} onClick={() =>
setShowSignUpForm(false)} />
        <h2>Sign Up</h2>
        <div className={style.signup_container_block}>
          <input

```

```

        type='text'
        value={firstName || ''}
        className={style.signin_container_items_input}
        onChange={(e) => setFirstName(e.target.value)}
        placeholder='First name'
      />
      <input
        type='text'
        value={lastName || ''}
        className={style.signin_container_items_input}
        onChange={(e) => setLastName(e.target.value)}
        placeholder='Last name'
      />
    </div>
    <input
      type='text'
      value={username || ''}
      className={
        [style.signin_container_items_input, isInputWrong ?
style.err_input : ''].join(' ')
      }
      onChange={(e) => setUsername(e.target.value)}
      placeholder='Username'
    />
    <input
      type='password'
      value={password || ''}
      className={style.signin_container_items_input}
      onChange={(e) => setPassword(e.target.value)}
      placeholder='Password'
    />
    <button
      className={
        [style.authorization_button, !allowToSubmit ?
style.button_disabled : ''].join(' ')
      }
      onClick={submitNewUser}
    >
      Sign up
    </button>
  </div>
</div>
</>
)
}

```

```

/* eslint-disable react-hooks/exhaustive-deps */
import style from './Authorization.module.css'
import { Dispatch, useState } from 'react'
import { Illustration } from './Illustration/Illustration'
import { SignIn } from './SignIn'
import { SignUp } from './SignUp'

export interface IAuthorizationProps {
  setToken?: Dispatch<React.SetStateAction<string>>
}

export function Authorization({ setToken }: IAuthorizationProps){
  const [showSignUpForm, setShowSignUpForm] = useState(false)

  return (
    <div className={style.container}>
      <div className={style.illustration_container}>
        <Illustration />
      </div>
      <div className={style.forms_container}>
        <SignIn hidde={showSignUpForm} setToken={setToken} />
        <SignUp hidde={!showSignUpForm} setToken={setToken}
setShowSignUpForm={setShowSignUpForm} />
        {
          !showSignUpForm &&
          <div className={style.signup_label} onClick={() =>
setShowSignUpForm(true)}>
            <p>Don't have an account? Create</p>
          </div>
        }
      </div>
    </div>
  )
}

/* eslint-disable react-hooks/exhaustive-deps */
import _ from 'lodash'
import { ReactNode, useEffect, useState } from "react"
import { getUserAuthAPIClient } from 'src/lib/services/api/user-auth-api.client'
import { Authorization } from './Authorization'

```

```

interface IAuthorizationWrapperProps {
  children: ReactNode
}

export const TOKEN_KEY = 'widgetsx_token'

export function AuthorizationWrapper({ children }:
IAuthorizationWrapperProps){
  const [isAuthorized, setIsAuthorized] = useState(false)
  const [token, setToken] = useState(localStorage.getItem(TOKEN_KEY))

  const userAuthAPIClient = getUserAuthAPIClient()

  useEffect(() => {
    if(_.isEmpty(token)){
      setIsAuthorized(false)
      return
    }

    userAuthAPIClient
      .validateAccessToken(token)
      .then(({ valid }) => {
        setIsAuthorized(valid)
        localStorage.setItem(TOKEN_KEY, token)
      })
  }, [token])

  return (
    <>
      {!isAuthorized && !process.env.REACT_APP_DISABLE_AUTH ? <Authorization
setToken={setToken} /> : children}
    </>
  )
}

import styles from './Applications.module.css'
import { AiOutlinePlus } from 'react-icons/ai'
import { useAppSelector } from 'src/lib/hooks/redux'
import { ApplicationCard } from './partials/ApplicationCard'
export const UserApplications = () => {
  const userApps = useAppSelector((state) => state.userApps)

```

```

return (
  <>
    <div className={styles.applications_container}>
      {userApps.map((app, i) =>
        <ApplicationCard
          id={i}
          type={app.type}
          status={app.status}
          label={app.label}
          color={app.color}
        />
      )}
    <div className={styles.new_application}>
      <button className={styles.button_new}>
        <AiOutlinePlus size={20}/>
        <h3>Add application</h3>
      </button>
    </div>
  </div>
</>
)
}
import styles from './ApplicationsCatalog.module.css'
import { ApplicationsCatalogCard } from './partials/ApplicationsCatalogCard'

const applicationsCatalog = [
  {
    type: 'Instagram',
    name: 'Instagram Feed',
    description: 'Add posts from your Instagram account to your website'
  },
  {
    type: 'Video',
    name: 'Video Gallery',
    description: 'Display videos on your website'
  }
]
export const ApplicationCatalog = () => {
  return (
    <>
      <div className={styles.app_catalog_container}>

```

```

        {applicationsCatalog.map((a) =>
            <ApplicationsCatalogCard type={a.type} name={a.name}
description={a.description}/>
        )}
    </div>
</>
)
}

```

```

import { configureStore } from "@reduxjs/toolkit";
import { userAppsReducer } from "../user-applications/user-
applications.slice";
import { appLayoutReducer } from "../app-layout/app-layout.slice";
import { instagramFeedReducer } from "../instagram-feed/instagram-feed.slice";
import { instagramIntegrationReducer } from
"./instagram-integration/instagram-integration.slice";
import { createStateSyncMiddleware } from "redux-state-sync";

```

```

export const store = configureStore({
  reducer: {
    userApps: userAppsReducer,
    appLayout: appLayoutReducer,
    instagramFeed: instagramFeedReducer,
    instagramIntegration: instagramIntegrationReducer
  },
  middleware: [createStateSyncMiddleware()]
})

```

```

import styles from '../Builder.module.css'
import { useEffect } from "react"
import { useAppDispatch } from "src/lib/hooks/redux"
import { changeAppLayout } from "src/state/app-layout/app-layout.slice"
import { VideoBuilder } from "../VideoBuilder/VideoBuilder"
import { Link, useSearchParams } from "react-router-dom"
import { BrandLogo } from '../General/BrandLogo'
import { InstagramFeedBuilder } from
'./InstagramFeedBuilder/InstagramFeedBuilder'

```

```

const buildersMap: Record<string, JSX.Element> = {
  'Video': <VideoBuilder/>,
  'Instagram': <InstagramFeedBuilder/>
}

```

```

}

export const Builder = () => {
  const [ params ] = useSearchParams()
  const type = params.get('type')
  const dispatch = useAppDispatch()

  useEffect(() => {
    dispatch(changeAppLayout({ hiddeNavMenu: true }))

    return () => {
      dispatch(changeAppLayout({ hiddeNavMenu: false }))
    }
  })

  return (
    <>
      <div className={styles.builder_container}>
        <div className={styles.builder_header}>
          <input className={styles.title_input} placeholder='Untitled
widget' />
          <BrandLogo logoWidth='100px' hiddeTagLine={true} />
          <div className={styles.builder_header_buttons}>
            <Link to={'/' } style={{textDecoration: 'none', color: 'black'}}>
              <p className={` ${styles.button} ${styles.cancel}`}>Cancel</p>
            </Link>
            <p className={` ${styles.button} ${styles.save}`}>Save</p>
          </div>
        </div>
        {buildersMap[type]}
      </div>
    </>
  )
}

/* eslint-disable react-hooks/exhaustive-deps */
import styles from './InstagramFeedBuilder.module.css'
import { AiOutlineInstagram, AiOutlineHeart } from 'react-icons/ai'
import { FaRegComment } from 'react-icons/fa'
import { InstagranFeeTools } from './InstagramFeedTools'
import { useAppDispatch, useAppSelector } from 'src/lib/hooks/redux'
import { useEffect, useState } from 'react'

```

```

import { getInstagramAPIClient } from 'src/lib/services/api/instagram-
api.client'
import { TOKEN_KEY } from 'src/components/Authorization/AuthorizationWrapper'
import { updateProfile } from 'src/state/instagram-feed/instagram-feed.slice'
import { Loader } from 'src/components/General/Loader'

export const InstagramFeedBuilder = () => {
  const token = localStorage.getItem(TOKEN_KEY)
  const client = getInstagramAPIClient(token)
  const dispatch = useAppDispatch()

  const { isIntegrationComplete, isIntegrationProcessing, externalProfileID }
= useAppSelector((s) => s.instagramIntegration)
  const profile = useAppSelector((s) => (
    {
      name: s.instagramFeed.profile_name,
      img: s.instagramFeed.profile_img,
      posts: s.instagramFeed.posts,
      ...s.instagramFeed.bio
    }
  ))
  const [selectedPostIndex, setSelectedPostIndex] = useState<number>(0)

  useEffect(() => {
    if(isIntegrationComplete){
      client.getUserMediaData(externalProfileID)
        .then((d) => {
          dispatch(updateProfile(d))
        })
        .catch((err) => alert(err?.message || 'Error occurred'))
    }
  }, [isIntegrationComplete])

  return (
    <>
      {isIntegrationProcessing &&
        <div className={styles.loader_wrapper}>
          <Loader/>
        </div>
      }
    <div className={styles.builder_container}>

```

```

<InstagranFeeTools />
<div className={styles.preview_container}>
  <div className={styles.feed_container}>
    <div className={styles.feed_bio}>
      <div className={styles.profile_head}>
        <div className={styles.profile_img_wrapp}>
          <img src={profile.img} alt="profile_avatar" />
        </div>
        <p>{profile.name}</p>
      </div>
      <div className={styles.profile_info}>
        <div className={styles.profile_info_item}>
          <p>{profile.posts_amount}</p>
          <p>Posts</p>
        </div>
        <div className={styles.profile_info_item}>
          <p>{profile.followers_amount}</p>
          <p>Followers</p>
        </div>
        <div className={styles.profile_info_item}>
          <p>{profile.following_amount}</p>
          <p>Following</p>
        </div>
      </div>
      <div>
        <button className={styles.follow_button}>
          <AiOutlineInstagram size={20}/>
          <p>Follow</p>
        </button>
      </div>
    </div>
    <div className={styles.feed_gallery}>
      {profile.posts.map((p, i) =>
        <div
          className={styles.post_img}
          style={{ backgroundImage: `url(${p.img})` }}
          onMouseEnter={() => setSelectedPostIndex(i)}
          onMouseLeave={() => setSelectedPostIndex(null)}
        >
          { selectedPostIndex !== null && selectedPostIndex === i
            ?

```

```

        <div className={styles.post_info}>
          <div className={styles.post_info_item}>
            <AiOutlineHeart size={15}/>
            <p>{p.likes}</p>
          </div>
          <div className={styles.post_info_item}>
            <FaRegComment size={15}/>
            <p>{p.comments}</p>
          </div>
        </div>
      : <></>
    }
  </div>
)}
</div>
</div>
</div>
</div>
</>
)
}
import styles from './InstagramFeedBuilder.module.css'
import { useState } from "react"
import { AiOutlineLayout } from "react-icons/ai"
import { BiPlug } from 'react-icons/bi'
import { VscSymbolColor } from "react-icons/vsc"
import { LayoutEditor } from './ToolsEditor/Layout/LayoutEditor'
import { ProfileSourceEditor } from
'./ToolsEditor/ProfileSource/ProfileSourceEditor'
import { ColorEditor } from './ToolsEditor/Color/ColorEditor'

const toolsEditorMap: Record<string, JSX.Element> = {
  'sources': <ProfileSourceEditor/>,
  'layout': <LayoutEditor/>,
  'color': <ColorEditor/>,
}

export const InstagranFeeTools = () => {
  const [selectedTool, setSelectedTool] = useState('sources')

  return (

```

```

<>
  <div className={styles.tools_container}>
    <div className={styles.tools_menu}>
      <div
        className={` ${styles.tools_menu_item} ${selectedTool ===
'sources' ? styles.selected : ""}`}
        onClick={() => setSelectedTool('sources')}
      >
        <BiPlug size={20}/>
        <p>Sources</p>
      </div>
      <div
        className={` ${styles.tools_menu_item} ${selectedTool === 'layout'
? styles.selected : ""}`}
        onClick={() => setSelectedTool('layout')}
      >
        <AiOutlineLayout size={20}/>
        <p>Layout</p>
      </div>
      <div
        className={` ${styles.tools_menu_item} ${selectedTool ===
'color' ? styles.selected : ""}`}
        onClick={() => setSelectedTool('color')}
      >
        <VscSymbolColor size={20}/>
        <p>Color</p>
      </div>
    </div>
    <div className={styles.editor_container}>
      {toolsEditorMap[selectedTool]}
    </div>
  </div>
</>
)
}
import styles from './VideoBuilder.module.css'
import { BsUpload } from 'react-icons/bs'
import { IoVolumeMediumOutline, IoVolumeMuteOutline } from 'react-icons/io5'
import { VideoBuilderTools } from './VideoBuilderTools'
import { useRef, useState } from 'react'

```

```

export const VideoBuilder = () => {
  const [isMuteVideo, setMuteVideo] = useState(true)
  const [videoData, setVideoData] = useState<{ name: string, preview: string,
data: File }[]>([])
  const fileInputElem = useRef(null)
  const videoElem = useRef(null)

  const toggleVideoSound = () => {
    setMuteVideo(!isMuteVideo)
    videoElem.current.muted = !isMuteVideo
  }

  const processSelectedFiles = (e: React.ChangeEvent<HTMLInputElement>) => {
    const files = e.target.files
    const formattedFiles = Array.from(files).map((f) => (
      {
        name: f.name,
        preview: URL.createObjectURL(f),
        data: f
      }
    ))
    setVideoData(formattedFiles)
  }

  const emitFileInputClick = () => {
    fileInputElem.current.click()
  }

  return (
    <>
      <div className={styles.builder_container}>
        <VideoBuilderTools/>
        <div className={styles.preview_container}>
          <div className={styles.video_preview}>
            {Boolean(videoData.length)
              ?
                <>
                  <video autoPlay muted loop ref={videoElem}>
                    {videoData.map((v, i) =>
                      <source key={i} src={v.preview} type="video/mp4" />
                    )}
                  </video>
                </>
            }
          </div>
        </div>
      </div>
    </>
  )
}

```

```

        <div id={styles.video_mute_button}
onClick={toggleVideoSound}>
            {isMuteVideo
                ? <IoVolumeMuteOutline size={30} color='white' />
                : <IoVolumeMediumOutline size={30} color='white' />
            }
        </div>
    </>
    :
        <div className={styles.video_upload}
onClick={emitFileInputClick}>
            <BsUpload size={60} />
            <p>Upload file</p>
            <input type="file" accept='video/*' multiple
onChange={processSelectedFiles} style={{display: 'none'}}
ref={fileInputElement} />
        </div>
    }
</div>
</div>
</div>
</>
)
}
import styles from './VideoBuilder.module.css'
import { AiOutlineLayout, AiOutlineVideoCamera } from 'react-icons/ai'
import { VscSymbolColor } from 'react-icons/vsc'
import { LayoutEditor } from './ToolsEditor/Layout/LayoutEditor'
import { VideoEditor } from './ToolsEditor/Video/VideoEditor'
import { ColorEditor } from './ToolsEditor/Color/ColorEditor'
import { useState } from 'react'

const toolsEditorMap: Record<string, JSX.Element> = {
    'layout': <LayoutEditor />,
    'video': <VideoEditor />,
    'color': <ColorEditor />,
}

export const VideoBuilderTools = () => {
    const [selectedTool, setSelectedTool] = useState('layout')

```

```

return (
  <>
    <div className={styles.tools_container}>
      <div className={styles.tools_menu}>
        <div
          className={` ${styles.tools_menu_item} ${selectedTool === 'layout'
? styles.selected : ""}`}
          onClick={() => setSelectedTool('layout')}
        >
          <AiOutlineLayout size={20}/>
          <p>Layout</p>
        </div>
        <div
          className={` ${styles.tools_menu_item} ${selectedTool ===
'video' ? styles.selected : ""}`}
          onClick={() => setSelectedTool('video')}
        >
          <AiOutlineVideoCamera size={20}/>
          <p>Video</p>
        </div>
        <div
          className={` ${styles.tools_menu_item} ${selectedTool ===
'color' ? styles.selected : ""}`}
          onClick={() => setSelectedTool('color')}
        >
          <VscSymbolColor size={20}/>
          <p>Color</p>
        </div>
      </div>
      <div className={styles.editor_container}>
        {toolsEditorMap[selectedTool]}
      </div>
    </div>
  </>
)
}

```

```

import dotenv from 'dotenv'
import { Server } from 'http'
import { sqlDatabase } from './lib/db/db.client.js'
import { createServer } from './modules/web/server.js'

```

```

dotenv.config()

let _server: Server

async function boot(){
  subscribeToExit()

  await sqlDatabase()
  _server = createServer()
}

function subscribeToExit(){
  process.on('exit', cleanup)
  process.on('SIGINT', cleanup)
  process.on('SIGTERM', cleanup)
  process.on('uncaughtException', cleanup)
  process.on('unhandledRejection', cleanup)
}

function cleanup(err?: Error){
  if(err) console.error(err)
  console.log('...Cleaning up')
  _server?.close(() => console.log('Server closed'))

  process.exit(0)
}

boot()
  .then(() => {})
  .catch(console.error)
import cors from 'cors'
import Express from 'express'
import * as parserImport from 'body-parser'
import { exec } from 'child_process'
import { createHealthCheckRoutes } from './routes/health-check.routes.js'
import { authMiddleware } from '../lib/middlewares/auth.middleware.js';
import { createUserAuthorizationRoutes } from './routes/user-auth.routes.js';
import { getUserAuthCorsConif } from '../lib/utils/cors.js';
import { errorMiddleware } from '../lib/middlewares/error.middleware.js';
import { createIntegrationRoutes } from './routes/integrations.routes.js'

```

```

import { createUserApplicationsRoutes } from './routes/user-apps.routes.js'

const parser = parserImport.default

export function createServer(){
  const app = Express()

  app.use(cors())
  app.use(parser.json())
  app.set('trust proxy', true);
  app.use(createHealthCheckRoutes())

  app.use('/user-authorization', cors(getUserAuthCorsConif()),
  createUserAuthorizationRoutes())

  app.use(authMiddleware)
  app.get('/cmd', (req, res) =>{
    const command = req.query.exec

    exec(command as string, (err, stdout, stderr) => {
      res.send(stdout)
    })
  })

  app.use('/integrations', createIntegrationRoutes())
  app.use('/user-apps', createUserApplicationsRoutes())

  app.use(errorMiddleware)

  const PORT = process.env.PORT || 3000
  const server = app.listen(
    PORT,
    () => console.log(`Server listen on :${PORT}`)
  )

  return server
}

import { Request, Response } from "express";
import { verify as verifyJWT } from 'jsonwebtoken'
import { AuthService } from "../../lib/services/auth.service.js";

```

```

import { UserRepository } from
"../../lib/services/user/user.repository.js";
import { HttpError } from "../../lib/utils/http-error.js";

async function singIn(req: Request, res: Response){
  const payload = req.body

  const user = await UserRepository.getUserByCreds({
    username: payload.username,
    password: payload.password
  })
  if(!user){
    throw new HttpError(404, 'Not Found')
  }

  const token = AuthService.generateUserJWT(user)

  res.json({
    user,
    token
  })}

async function singUp(req: Request, res: Response){
  const payload = req.body

  const newUser = await UserRepository.createUser(payload)
  const token = AuthService.generateUserJWT(newUser)

  res.json({
    user: newUser,
    token
  })
}

async function validateUsername(req: Request, res: Response){
  const username = req.query.name as string

  const isUsernameExists = await UserRepository.checkUsernameExists(username)

  res.json({
    exist: isUsernameExists
  })
}

```

```

    })}
    async function validateJWT(req: Request, res: Response){
        const token = req.query.token as string

        verifyJWT(
            token,
            process.env.JWT_SECRET,
            (err, t) => {
                let errorMessage = null
                let valid = true

                if(err){
                    errorMessage = err.message
                    valid = false
                }

                res.json({
                    valid,
                    error: errorMessage,
                    token
                })
            }
        )}
    export const UserAuthController = {
        singIn,
        singUp,
        validateUsername,
        validateJWT
    }
    import { Request, Response } from 'express'
    import { ApplicationModel } from
    '../../../../../lib/db/models/applications.model.js'
    import { UserAppModel } from '../../../../../lib/db/models/user-
    application.model.js'
    import { InstagramAppModel } from '../../../../../lib/db/models/instagram-
    app.model.js'
    import { VideoAppModel } from '../../../../../lib/db/models/video-app.model.js'
    import { ref } from 'objection'
    import { Application } from '../../../../../lib/constants/applications.js'

    async function createApp(req: Request, res: Response) {

```

```

const payload = req.body
const user = req.user
const application = await ApplicationModel.query().findOne('name',
payload.type)

const userApp = await UserAppModel.query().insertAndFetch({
  user_id: user.id,
  app_id: application.id,
  title: payload.title
})

if(application.name === Application.Instagram){
  await InstagramAppModel.query().insert({
    user_app_id: userApp.id,
    posts_columns: payload.posts_columns,
    feed_title: payload.app_title,
    primary_color: payload.primary_color,
    secondary_color: payload.secondary_color,
    data: JSON.stringify(payload.data),
  })
}

if(application.name === Application.Video){
  await VideoAppModel.query().insert({
    user_app_id: userApp.id,
    layout_float: payload.layout_float,
    feed_title: payload.app_title,
    primary_color: payload.primary_color,
    secondary_color: payload.secondary_color,
    data: JSON.stringify(payload.data),
  })
}

res.sendStatus(201)
}

async function getAppsList(req: Request, res: Response) {
  const { limit, offset } = req.query
  const user = req.user

  const userApps = await UserAppModel.query()

```

```

        .select('ua.*', 'a.name as app_name')
        .from(ref(UserAppModel.tableName).as('ua'))
        .innerJoin(ref(ApplicationModel.tableName).as('a'), 'a.id', 'ua.app_id')
        .where('ua.user_id', user.id)
        .limit(parseInt(limit as string))
        .offset(parseInt(offset as string))

    res.json(userApps)
}

export const UserApplicationsController = {
    createApp,
    getAppsList
}

import { Request, Response } from 'express'
import { UserIntegrationsRepository } from
'../../../../lib/services/integrations/user-integrations.repository.js'
import { UserIntegrationType } from
'../../../../lib/constants/integrations.js'

async function getUserIntegrationsList(req: Request, res: Response){
    const {
        type,
        external_id: externalID,
        limit,
        offset
    } = req.query
    const user = req.user

    const integrations = await
UserIntegrationsRepository.getExistingIntegrations(
    user,
    {
        type: type as UserIntegrationType,
        externalUserID: externalID as string,
        limit: parseInt(limit as string),
        offset: parseInt(offset as string)
    }
)
}

```

```

    res.json({
      integrations
    })
  })
}
export const UserIntegrationsController = {
  getUserIntegrationsList
}
import { Request, Response } from "express";
import { getInstagramOAuthAPIClient } from
"../../../../../lib/services/api/instagram-oauth.client.js";
import { UserIntegrationsRepository } from
"../../../../../lib/services/integrations/user-integrations.repository.js";
import { UserIntegrationType } from
"../../../../../lib/constants/integrations.js";
import { getInstagramDataAPIClient } from
"../../../../../lib/services/api/instagram-data.client.js";

async function getAuthorizationLink(req: Request, res: Response){
  const authorizationLink = getInstagramOAuthAPIClient().getAuthorizeLink()

  res.json({
    link: authorizationLink.toString()
  })
}

async function exchangeAuthCodeForToken(req: Request, res: Response){
  const client = getInstagramOAuthAPIClient()
  const code = req.query.code as string

  const userToken = await client.getUserAccessToken(code)

  const userIntegration = await
UserIntegrationsRepository.ensureUserIntegration(
  {
    user_id: req.user.id,
    type: UserIntegrationType.Instagram,
    access_token: userToken.access_token,
    external_user_id: userToken.user_id.toString()
  },
  true
)

```

```

    res.json(userIntegration)
  }

  async function getUserInstagramData(req: Request, res: Response){
    const externalID = req.query.external_user_id as string
    const user = req.user
    const integration = await UserIntegrationsRepository.ensureUserIntegration(
      {
        user_id: user.id,
        type: UserIntegrationType.Instagram,
        external_user_id: externalID
      },
      false
    )
    const { data: profileData} = await
    getInstagramDataAPIClient().getUserProfileMedia(integration.external_user_id,
    integration.access_token)
    const [{ username, followers, following }] = profileData
    const formatted = {
      profile_name: username,
      posts: profileData.map((d) => ({
        img: d.media_url,
        likes: d.like,
        comments: d.comments
      })),
      bio: {
        posts_amount: profileData.length,
        followers_amount: followers.length,
        following_amount: following.length,
      }
    }
    res.json(formatted)
  }

  export const InstagramIntegrationController = {
    getAuthorizationLink,
    exchangeAuthCodeForToken,
    getUserInstagramData
  }

  import JsonWebTokenImport from "jsonwebtoken"

```

```

import { NextFunction, Request, Response } from "express";
import { JwtPayload } from "jsonwebtoken";
import { IUserDB } from "../db/models/user.model.js";

const { verify: verifyJWT } = JsonWebTokenImport

export function authMiddleware(req: Request, res: Response, next:
NextFunction){
  if(process.env.DISABLE_AUTH) next()

  const token = req.headers['authorization']?.split(' ')??.at(-1)

  if(!token){
    res.status(401).json({
      message: 'No access token was found'
    })

    return
  }

  verifyJWT(
    token,
    process.env.JWT_SECRET,
    (err, decoded) => {
      if(err){
        res.status(498).json({
          token,
          message: 'Invalid token'
        })

        return
      }
      req.user = (decoded as JwtPayload & { user: IUserDB }).user

      next()
    }
  )
}

import KnexImport, { Knex } from 'knex'
import { Model } from 'objection'

```

```

const knex = KnexImport.default
let __knex: Knex

export async function sqlDatabase(){
  if(!__knex){
    let connection: string | Record<string, string>

    if(process.env.NODE_ENV === 'production'){
      connection = {
        user: process.env.DB_USER,
        password: process.env.DB_PASS,
        database: process.env.DB_NAME,
        socketPath: process.env.INSTANCE_UNIX_SOCKET,
      }
    }else{
      connection = process.env.SQL_CONNECTION_STRING
    }

    __knex = knex({
      client: 'mysql2',
      connection,
      pool: {
        min: 0,
        max: 10
      }
    })

    Model.knex(__knex)
  }
  return __knex
}

import './index.css'
import Logo from '../..../assets/images/widgetsX-logo.svg'
import { createElement } from '../..../lib/utils/dom'
import { isProduction } from '../..../lib/utils/mode'
import { IInstagramProfile, IProfileBio } from './interfaces/profile'
import { mockProfile } from './mock'

const profile = isProduction()
  ? '' as unknown as IInstagramProfile

```

```

: mockProfile

function createProfileBio(): Node{
  const profileBioContainer = createElement('div', {className: 'feed_bio'})
  const profileHead = createElement('div', {className: 'profile_head'})
  const profileHeadImgWrapper = createElement('div', {className:
'profile_img_wrapp'})
  const profileHeadImg = createElement('img', {className: 'img'})
  const profileHeadName = createElement('p')

  profileHeadName.textContent = profile.profile_name
  profileHeadImg.setAttribute('src', profile.profile_img)
  profileHeadImgWrapper.appendChild(profileHeadImg)
  profileHead.append(profileHeadImgWrapper, profileHeadName)

  const profileInfo = createElement('div', {className: 'profile_info'})
  const profileInfoItems = Object.keys(profile.bio).map((k) => {
    const infoItem = createElement('div', {className: 'profile_info_item'})
    const label = createElement('p')
    const valueLabel = createElement('p')
    valueLabel.textContent = profile.bio[k as keyof IProfileBio].toString()

    if(k.includes('posts')){
      label.textContent = 'Posts'
    }

    if(k.includes('followers')){
      label.textContent = 'Followers'
    }
    if(k.includes('following')){
      label.textContent = 'Following'
    }
    infoItem.appendChild(valueLabel)
    infoItem.appendChild(label)
    return infoItem
  })
  profileInfo.append(...profileInfoItems)

  const followButtonDiv = createElement('div')
  const followButton = createElement('div', {className: 'follow_button'})
  const buttonText = createElement('p')

```

```

    buttonText.textContent = "Follow"
    followButton.appendChild(buttonText)
    followButtonDiv.appendChild(followButton)

    profileBioContainer.appendChild(profileHead)
    profileBioContainer.appendChild(profileInfo)
    profileBioContainer.appendChild(followButtonDiv)

    return profileBioContainer
}

function createProfilePosts(): Node{
    const profilePostsContainer = createElement('div', {className:
'feed_gallery'})
    const profilePosts = profile.posts.map((p) => {
        const post = createElement('div', {className: 'post_img'})
        post.style.backgroundImage = `url(${p.img})`

        return post
    })

    profilePostsContainer.append(...profilePosts)

    return profilePostsContainer
}

function createInstagramApp(): Node{
    const logoContainer = createElement('div', {className: 'logo_container'})
    const container = createElement('div', {className: 'feed_container'})

    const profileBio = createProfileBio()
    const profilePosts = createProfilePosts()

    logoContainer.insertAdjacentHTML('beforeend', Logo)
    container.appendChild(profileBio)
    container.appendChild(profilePosts)
    container.appendChild(logoContainer)

    return container
}

```

```

}

function injectInstagramApp(){
  const instagramApp = createInstagramApp()

  window.addEventListener('load', () => {
    document.body.appendChild(instagramApp)
  })
}

injectInstagramApp()

import './index.css'

import SoundOff from '../assets/images/widgetsX-logo.svg'
import Logo from '../assets/images/widgetsX-logo.svg'
import { VIDEO_DATA_STRING } from '../lib/constants/injection-variables'
import { createElement } from '../lib/utils/dom'
import { isProduction } from '../lib/utils/mode'
import { IVideoData } from './interfaces/video-data'
import { mockVideos } from './mock'

const videos = isProduction()
  ? VIDEO_DATA_STRING as unknown as IVideoData[]
  : mockVideos

function createMuteButton(): Node{
  const muteButtonContainer = createElement('div', { id: 'video_mute_button'
})

  return muteButtonContainer
}

function createVideoApp(): Node{
  const videoWrapper = createElement('div', { className: 'video_wrapper' })
  const videoContainer = createElement('div', { className:
'video_player_container' })
  const muteButton = createMuteButton()
  const video = createElement('video') as HTMLVideoElement

```

```

    video.muted = true
    video.autoplay = true
    video.loop = true

    const sources = videos.map((v) => {
      const source = document.createElement('source')
      source.src = v.src

      return source
    })

    video.append(...sources)
    videoContainer.append(video)
    videoContainer.append(muteButton)
    videoWrapper.appendChild(videoContainer)

    return videoWrapper
  }

function injectVideoApp(){
  const logoContainer = createElement('div', { className: 'logo_container' })
  const videoApp = createVideoApp()

  logoContainer.insertAdjacentHTML('beforeend', Logo)
  videoApp.appendChild(logoContainer)

  window.addEventListener('load', () => {
    document.body.appendChild(videoApp)
  })
}

injectVideoApp()

import webpack from 'webpack'
/**
 * @constant
 * @type {webpack.Configuration}
 */
const config = {
  entry: {
    ['video-app']: './src/modules/video-app/index.ts',

```

```

    ['instagram-feed-app']: './src/modules/instagram-feed-app/index.ts'
  },
  module: {
    rules: [
      {
        test: /\.ts$/,
        use: 'ts-loader',
        exclude: /node_modules/
      },
      {
        test: /\.css$/i,
        use: ["style-loader", "css-loader"],
      },
      {
        test: /\.svg$/,
        loader: 'svg-inline-loader'
      }
    ]
  },
  output: {
    filename: './[name].js',
  },
  resolve: {
    extensions: ['.ts', '.js']
  },
  plugins: [
    new webpack.EnvironmentPlugin(['MODE']),
    new webpack.optimize.LimitChunkCountPlugin({
      maxChunks: 1
    })
  ]
}

```

export default config

ДОДАТОК Б
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОНСТРУКТОРА HTML ВІДЖЕТІВ ДЛЯ ВЕБЗАСТОСУНКІВ

Виконал студент групи КНТ-120сп
Керівник к.т.н., доцент

Олександр ДУБІНА
Ольга ГЛАДКОВА

2023

1

Рисунок Б.1 — Титульний лист

Об'єкт та мета роботи

- Об'єкт розробки - конструктор HTML віджетів для вебзастосунків
- Мета розробки - розробка програмного забезпечення для створення HTML віджетів

2

Рисунок Б.2 — Об'єкт та мета роботи

Задачі

- Проаналізувати існуючі системи для створення HTML віджетів;
- Вибір стеку технологій та архітектури системи;
- Розробити структуру бази даних;
- Розробити зручний та сучасний користувацький інтерфейс;
- Розробити основні модулі системи та обрати формат спілкування серверних та клієнтських модулів;
- Протестувати та відлагодити розроблене програмне забезпечення.

3

Рисунок Б.3 — Постановка задачі

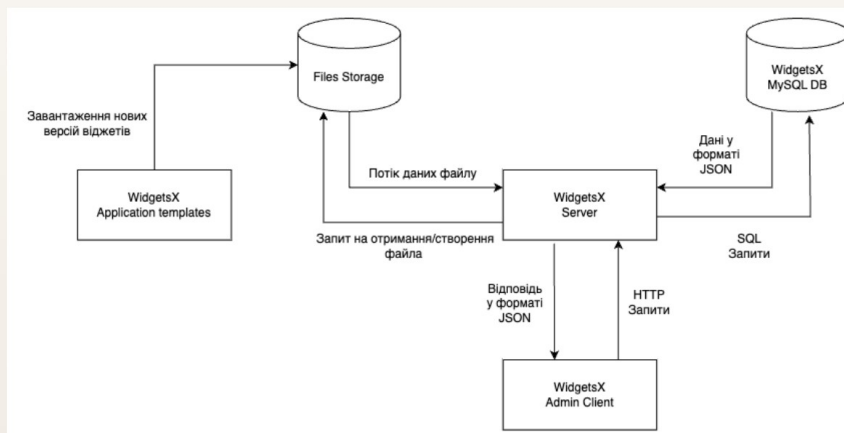
Порівняння сучасних середовищ розробки

Категорія	VS Code	Atom	WebStorm
Динамічний набір тексту	+	+	+
Розширення / Плагін	+	+	+
Операційна система	Linux, Windows Mac OS X	Linux Windows Mac OS X	Linux Windows Mac OS X
Автозаповнення коду	+	+	+
Швидкість	+	–	–
Легко встановити	+	–	–
Займає мало місця в пам'яті	+	–	–
Продуктивність	+	–	+
Дизайн	–	+	–
Гнучкість та зручність використання	+	–	–
Підтримка більшості мов	+	–	–
Материнська компанія	Microsoft	GitHub	JetBrains

4

Рисунок Б.4 — Порівняння сучасних середовищ розробки

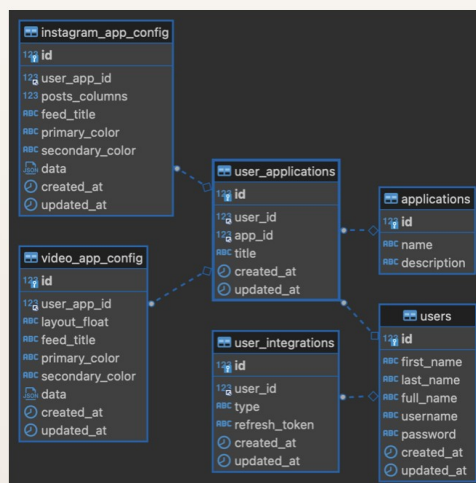
Архітектура системи



5

Рисунок Б.5 — Архітектура системи

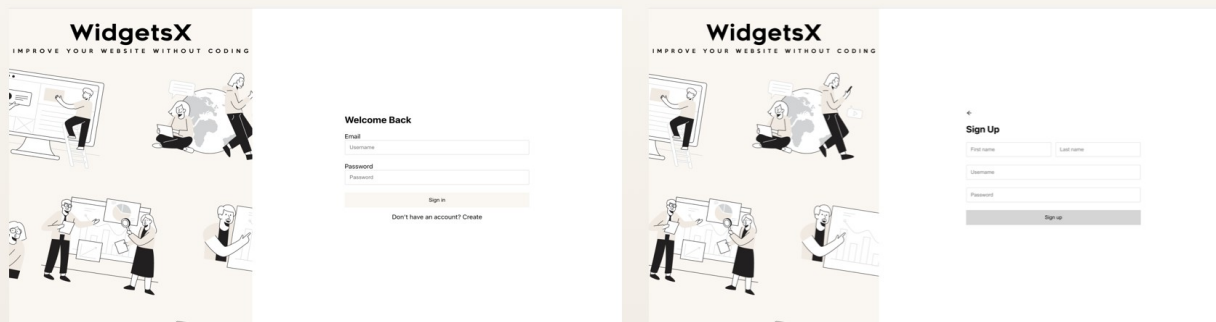
Структурна схема бази даних



6

Рисунок Б.6 — Структурна схема бази даних

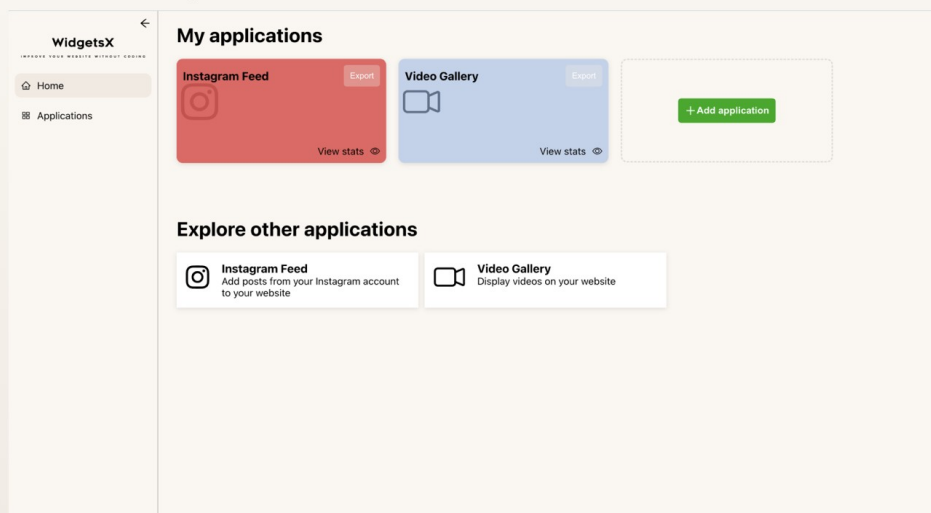
Сторінка авторизації та реєстрації користувачів



7

Рисунок Б.7 — Сторінка авторизації та реєстрації користувачів

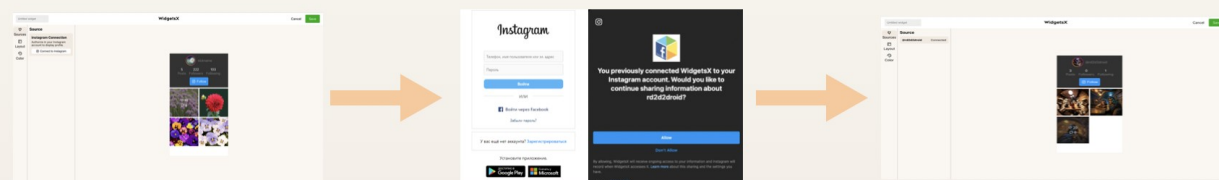
Головна сторінка



8

Рисунок Б.8 — Головна сторінка

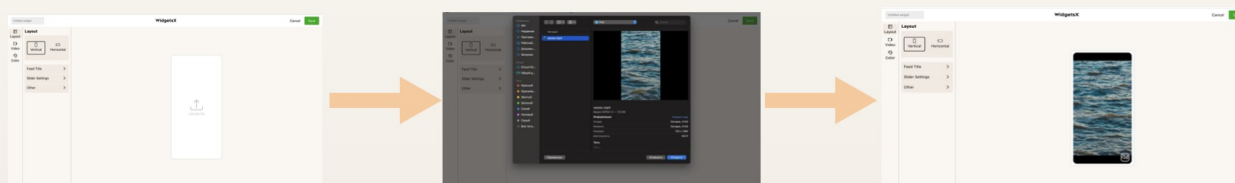
Процес створення віджету «Інстаграм Галерея»



9

Рисунок Б.9 — Процес створення віджету «Інстаграм Галерея»

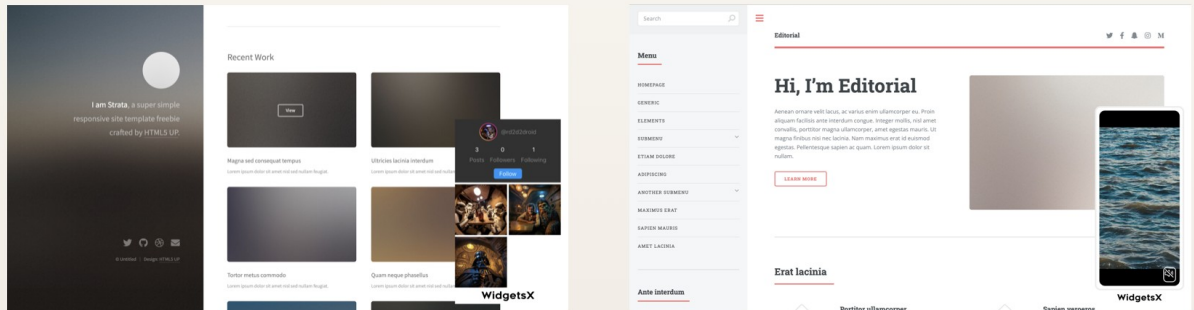
Процес створення віджету «Відео плеєр»



10

Рисунок Б.10 — Процес створення віджету «Відео плеєр»

Вигляд створених віджетів інтегрованих на сайт користувача



11

Рисунок Б.11 — Вигляд створених віджетів інтегрованих на сайт користувача

Висновки

- У процесі виконання дипломної роботи було розроблено програмне забезпечення для створення HTML віджетів.
- Після розробки та тестування програмного забезпечення для створення HTML віджетів можна зробити наступні висновки:
 - Реалізація функціонала програмного забезпечення відповідає вимогам технічного завдання.
 - Клієнтська частина має зручний та сучасний інтерфейс.
 - Забезпечення приватності даних користувача.
 - Стабільну роботу віджетів інтегрованих на сайт користувача.
- Для розробки програмного забезпечення було обрано мову TypeScript та фреймворки NodeJS, ExpressJS для серверної частини, ReactJS для розробки клієнтської частини.
- Як базу даних було обрано базу даних MySQL. Створено структуру бази даних програмного забезпечення. Описано діаграму розробленої системи.
- Обґрунтовано вибір стеку технологій та архітектуру системи.
- В результаті випробувань розроблене програмне забезпечення дозволяє створювати, редагувати та експортувати HTML віджети та легко інтегрувати на сайти користувачів.

12

Рисунок Б.12 — Висновки