

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:
Завідувач кафедри
Сергій СТИРЕНКО**

(підпис)

“ _ ” _____ 2021 р.

**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою “Інженерія програмного
забезпечення комп’ютерних систем” спеціальності 121 “Програмна
інженерія”**

на тему: Конструктор односторінкових сайтів

Виконала: студентка _____ 4-го курсу, групи ІПЗ-72
(шифр групи)

Трачук Вікторія Сергіївна
(прізвище, ім’я, по батькові) _____
(підпис)

Керівник _____ ас. Калюжний О. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____
(підпис)

Консультант(нормоконтроль) проф. к. ОТ, д.т.н., Сімоненко В. П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) _____
(підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____
(підпис)

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студентка _____
(підпис)

Київ – 2021 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальність 121 “Програмна інженерія”

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТИРЕНКО

(підпис)

“ _ ” _____ 2021 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студентки

Трачук Вікторії Сергіївни

1 Тема проєкту Конструктор односторінкових сайтів
керівник проєкту ас. Калюжний О. О.,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 11 травня 2021 року № 1139-с

2 Термін здачі студентки закінченого проєкту 30 травня 2021 р.

3 Вихідні дані до проєкту технічна документація. теоретичні дані

4 Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) огляд існуючих рішень та принципів створення сайту користувачем, аналіз технологій для реалізації додатку, додаток для конструктора односторінкових сайтів.

5 Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) функціональна схема системи, схема загального алгоритму, діаграма класів системи.

6 Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7 Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	<i>10.12.2021-15.12.2021</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2021-15.03.2021</i>	
3.	<i>Розробка архітектури та загальної структури додатку</i>	<i>15.03.2021-25.03.2021</i>	
4.	<i>Розробка структур окремих модулів додатку</i>	<i>25.03.2021-5.04.2021</i>	
5.	<i>Програмна реалізація додатку</i>	<i>5.04.2021-15.04.2021</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2021-20.05.2021</i>	
7.	<i>Захист програмного продукту</i>	<i>23.04.2021</i>	
8.	<i>Передзахист</i>	<i>21.05.2021</i>	
9.	<i>Захист</i>	<i>15.06.2021</i>	

Студентка-дипломник_____.
(підпис)

Керівник проєкту_____.
(підпис)

АНОТАЦІЯ

В бакалаврському дипломному проекті розроблено конструктор для односторінкових сайтів. За останні роки, з поширенням інтернету у повсякденному житті, достатньо сильно розвинулись засоби представлення інформації та виросла необхідність у конкурентній представленості своєї роботи, або свого дизайну сайту.

Запропоноване програмне забезпечення вирішує основні проблеми аналогічного даному в цій сфері – швидкість розробки для користувача, простий порог входу, перспектива можливості просування SEO. Додаток має широку сферу застосування для людей, які продвинуто володіють основними функціями програмного забезпечення, та для людей які лише базово з ними знайомі.

Програмне забезпечення було написане на мові JavaScript, за допомогою фреймворка Vue.js, з використанням ряду швидкодіючих бібліотек. До них ввійшли Firebase, Vuex, Vue Router, Vue Cli, та інші.

ABSTRACT

A constructor for one-pages sites has been developed in the bachelor's diploma project. For the rest of the rock, with the widening of the Internet in the widespread life, there has been a lot of progress in the presentation of information and the need for a competitive presentation of their work, or design of the site.

The proposed software solves the main problems of similar in this area - the speed of the distribution for the clerk, a simple entry threshold, the prospect of SEO. The application has a wide scope of storage for people, who are advanced by the main functions of software security, and for people who are deprived of their basic knowledge.

The software was written in JavaScript, using Vue.js framework, and a number of fast libraries. These include Firebase, Vuex, Vue Router, Vue Cli, and others.

**Опис альбому
до дипломного проєкту**

на тему: «Конструктор односторінкових сайтів»

Київ – 2021 року

[illegible]

Технічне завдання до дипломного проєкту

на тему: «Конструктор односторінкових сайтів»

Київ – 2021 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ.....	2
2. ПРИЧИНИ ДЛЯ РОЗРОБКИ.....	2
3. ЦІЛЬ ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4. ДЖЕРЕЛА РОЗРОБКИ.....	2
5. ТЕХНІЧНІ ВИМОГИ.....	3
• ВИМОГИ ДО ПРОДУКТУ, ЩО РОЗРОБЛЯЄТЬСЯ.....	3
• ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	3
6. ЕТАПИ РОЗРОБКИ.....	3

					ІАЛЦ.467800.002 ТЗ					
Зм.	Арк.	№ докум.	Підпис	Дата	Конструктордносторінкових сайтів Технічне завдання					
Розробив	Трачук В.С.									
Перевірів	Калюжний О. О.									
Реценз.										
Н. Контр.	Сімоненко В.П.									
Затв.	Стіренко С.Г.				Літ.		Аркуш		Аркушів	
							1		3	
					«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІПз-72					

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Додаток для односторінкових сайтів (Конструктор) може використовуватися у цілях реалізації будь-якої ідеї проекту користувача. Це може бути сайт-візитка, онлайн магазин, представлення власного бізнесу, або просто відтворення UX дизайну користувачем у інтернеті.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проекту професійної програми “Інженерія програмного забезпечення комп’ютерних систем” спеціальності 121 “Програмна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення додатку для можливості реалізації сайту користувача, який буде включати у себе комбінацію вирішень недоліків подібних проектів на ринку.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

					<i>ІАЛЦ.467800.003 ПЗ</i>	Арк
Зм.	Арк.	№ докум.	Підп.	Дата		2

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до розробленого продукту

- Багатий вибір можливості відтворення ідеї або дизайну користувача.
- Висока швидкість.
- Проста подача для користувачей, які не добре знайомі із програмним забезпеченням.
- Кроссплатформність.

5.2 Вимоги до програмного забезпечення

Підтримується будь-якою операційною системою для десктопного застосування, може бути застосована у будь-якому браузері. Пізніше буде розроблена мобільна версія.

6. ЕТАПИ РОЗРОБКИ

Етап розробки	Дата
6.1 Вивчення необхідної літератури	21.02.2021
6.2 Складання і узгодження технічного завдання	08.03.2021
6.3 Написання вступної частини та огляд рішень	22.03.2021
6.4 Розробка архітектури додатку	04.04.2021
6.5 Написання програмної частини	11.04.2021
6.6 Тестування та виправлення помилок	04.05.2021
6.7 Оформлення документації дипломного проєкту	15.05.2021

**Пояснювальна записка
до дипломного проєкту**

на тему: «Конструктор односторінкових сайтів»

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. Огляд існуючих рішень.....	4
1.1 Перелік прикладів.....	4
1.2 Розбір прикладів, недоліки та переваги.....	4
ВИСНОВОК ДО ПЕРШОГО РОЗДІЛУ.....	15
РОЗДІЛ 2. ОПИС МЕТОДИК НЕОБХІДНИХ ДЛЯ ВИРІШЕННЯ.....	16
2.1 Перелік основних технологій та бібліотек.....	16
2.2 Детальний опис кожної технології.....	16
ВИСНОВОК ДО ДРУГОГО РОЗДІЛУ.....	33
РОЗДІЛ 3. ОПИС ПРОЕКТУ.....	34
3.1 Структура проекту.....	34
3.2 Опис модулів та класів додатку.....	42
ВИСНОВОК ДО ТРЕТЬОГО РОЗДІЛУ.....	52
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	54
ДОДАТОК 1 Функціональна схема.....	57
ДОДАТОК 2 Діаграма класів.....	59
ДОДАТОК 3 Схема алгоритму.....	61
ДОДАТОК 4 Лістинг програми.....	63

					<i>ІАЛЦ.467800.003 ПЗ</i>		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Трачук В.С.			Конструктор для односторінкових сайтів Пояснювальна записка	Літ.	Арк.
Перевірів		Калюжний О. О.					1
Реценз.							73
Н. Контр.		Сімоненко В.П.				«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІПЗ-72	
Затв.		Стіренко С.Г.					

ВСТУП

У наш час дуже важливою частиною представлення будь-якої інформації стала наша веб-візитка, або наш власний веб інтерфейс. Це стосується як усіх обширних тем, таких як медицини, навчання, законодавства, так і просто маленького бізнес стартапу. Всю інформацію зараз ми отримуємо з інтернету. Тому дуже важливий крок побудови саме цього інтефейсу — його правильність, доступність, відповідний дизайн та при цьому простота. Під “простотою” я маю на увазі те, щоб інтерфейс був зрозумілий та комфортний як для молодого покоління, так і для люде й преклонного віку.

Метою дослідження є створення програми для можливості побудувати будь-якому користувачу свій інтерфейс, із власним дизайном та вимогами.

Об’єктом дослідження є моделі побудови веб-інтерфейсу.

Предметом дослідження є дослідження методів побудови веб інтерфейсів.

Завданням даної роботи є створення програми, яка буде корисною щодо представлення будь-якої інформації, для людей з невисоким рівнем технічної підготовки (або зовсім без неї).

Для успішного виконання даної дипломної роботи необхідно:

1. Опрацювати наукову літературу із обраної теми;
2. Провести огляд існуючих алгоритмів створення сайтів;
3. Розробити програму із інтерфейсом для користувача.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Зараз на ринку існує багато подібних рішень. Ми розглянемо декілька прикладів, які є лідерами у всьому світі, та декілька популярних прикладів у нашій країні. Розглянемо, які проблеми можуть виникати у користувача при взаємодії з ними.

1.1 Перелік прикладів

-Wix.com

-networksolutions.com

-site123.com

-tilda.com

-squarespace.com

1.2. Розбір прикладів, недоліки та переваги

- Wix.com

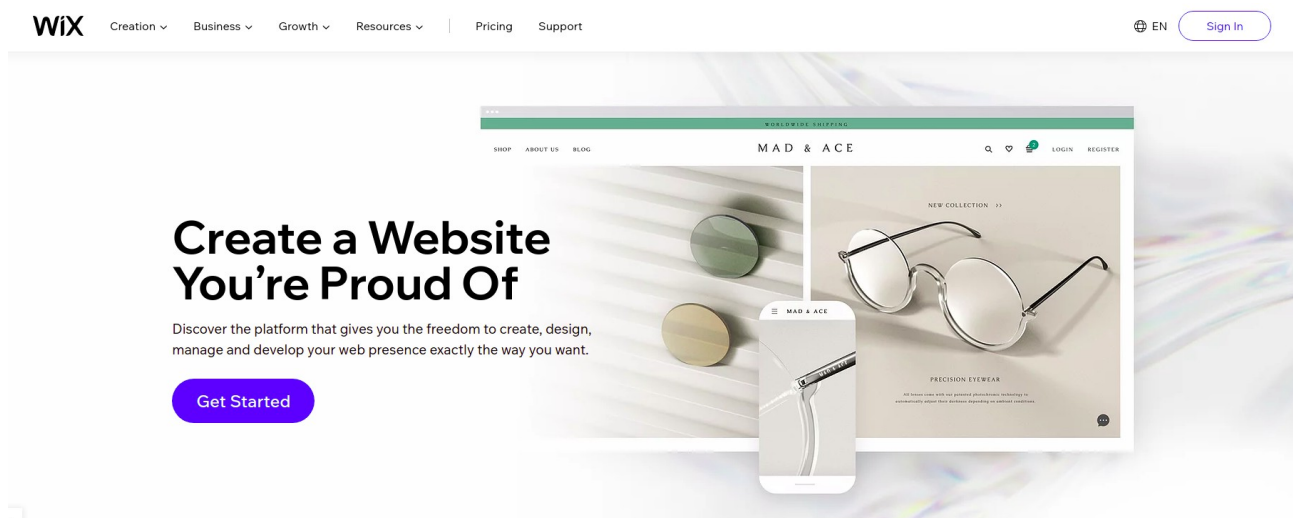


Рис. 1.1 – Знімок екрана веб-інтерфейсу Wix.com.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		4

Wix.com був заснований у 2008 році у США. Це платформа для тих, хто хоче швидко і без особливих зусиль (знань) створити сучасний сайт, який може бути: сайтом візиткою, галереєю, портфоліо і т.д. Звичайно, це плюс, але є і оборотна сторона - зробити щось більш серйозне, як наприклад, аналітика, калькулятор, або інші додаткові функції неможливо. Є можливість оплатити додаткові небезкоштовні функції або сервіси, які частково зможуть вирішити ці питання, але треба пам'ятати, що їх потрібно оплачувати на постійній основі. [1]

Розглянемо більш конкретно функціонал Wix.com.

Недоліки:

1) Неможливість SEO просування.

Важливий момент для користувача, який хоче представити, наприклад, себе або свій бізнес чи магазин — це можливість вийти на рівень із конкурентами. Без правильно налаштованого SEO це буде неможливо.

1) Обмежений функціонал. Як і було описано раніше, за допомогою вікс добре реалізовувати нескладну ідею, як сайт візитка або портфоліо. З чимось більш обширним та аналітичним виникнуть проблеми.

2) Фінансова залежність. При виборі сайту конструктору треба розуміти, які ціни можуть вас влаштовувати. Є деякі дослідження, у яких підводиться до того, що вигідніше буде оплатити один раз розробку повноцінного сайту, без сторонніх рішень, та оплачувати після тільки домен та хостинг.

3) Правообладання вашим сайтом. Якщо у вас виникне необхідність перенести свою розробку сайту з вікс на будь-які інші рішення, це буде неможливо. Все, що було розроблено на вікс може залишатися лише там. Перенести можливо тільки домен.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		5

- networksolutions.com

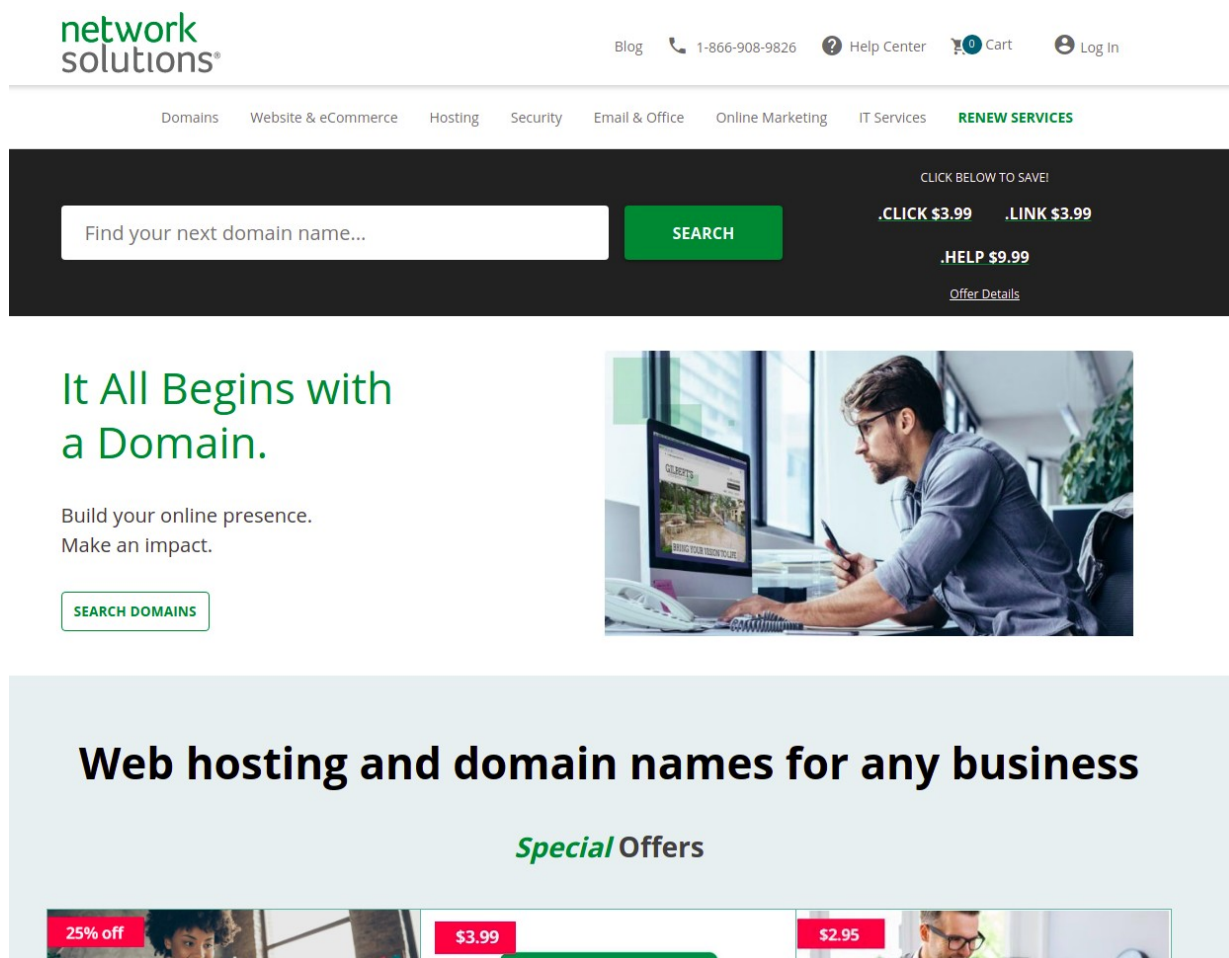


Рис. 1.2 – Знімок екрана веб-інтерфейсу networksolutions.com.

Насамперед слід зрозуміти, що цей сервіс не підніме ваш бізнес на якісно новий рівень в найближчому майбутньому. За допомогою Network Solution ви зможете опублікувати свій сайт в мережі і заявити про свою присутність. Тут, звичайно, є деякі дуже корисні функції (наприклад, SEO-просування і інтеграція з соціальними мережами), але сервісу явно не вистачає комфортного для користувача дизайну і сумісності з мобільними пристроями. [2]

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛЦ.467800.003 ПЗ

Арк.

6

По частині сервісу, його послуги доступні за ціною, і в цьому сенсі він навіть не поступається своїм конкурентам. Редактор цього сервісу є одним з найбільш простих в роботі, з хорошим меню і однозначними інструкціями, що стосуються процесу створення веб-сайту.

У Network Solutions понад 200 шаблонів найрізноманітнішої тематики, але дуже мало з них виконані якісно. Як плюс, всі шаблони сервісу легко можна доопрацювати, щоб вони виглядали більш привабливо, що займе трохи додаткового часу.

Користувач може налаштовувати аудіо, відео та інші мультимедійні-елементи. Ви не зможете кардинально їх змінити, але приблизно вони все ж впишуться в ваш шаблон.

Network Solutions має багато функцій, але вони всі односторонні. Наприклад, у є SEO-функції і можливість інтеграції з соціальними мережами, але вони дуже мізерні, при цьому, при бажанні їх можна розширити. Є інтеграція оплати через PayPal, можливість перевіряти статистику по відвідувачах, а також додавати власні домени та адреси електронної пошти в свій інтерфейс адміністратора. Найбільший недолік тут в тому, що створений вами веб-сайт не буде відразу ж сумісний з мобільними пристроями. Для цього вам потрібно буде створити окремі HTML- і CSS-файли, але ж далеко не кожен користувач знає, як це зробити.

Навчитися працювати з редактором Network Solution дуже просто, але в ньому немає всіх інструментів, які вам можуть знадобитися, щоб створити повноцінний веб-сайт. В меню редактора дуже просто орієнтуватися, оскільки його структура інтуїтивно зрозуміла.

Додавати елементи на свою сторінку дуже просто завдяки функції drag-and-drop. Всі основні властивості цих елементів змінюються в інтуїтивно зрозумілій редакторі, але іноді досить важко вписати дизайн елемента в загальний дизайн вашого веб-сайту.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		7

- site123.com



Главная

Премиум-Планы

О Нас

Логин

Начните здесь

Создать Сайт Бесплатно

Готовые стили и шаблоны для вашего удобства. Никакого веб-дизайна или программирования. Конструктор сайтов SITE123 - 1, 2, 3 и готово!

НАЧНИТЕ ЗДЕСЬ

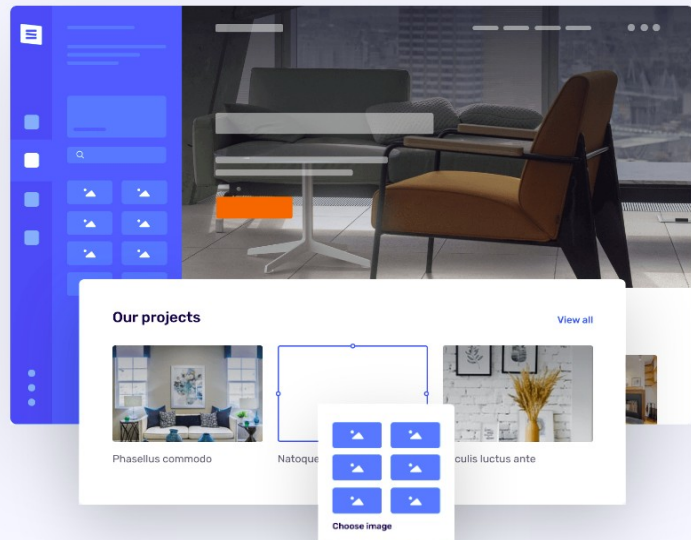


Рис. 1.3 – Знімок екрана веб-інтерфейсу site123.com.

Плюси:

- Гідний функціонал з налаштування дизайну, висока швидкість відкриття сайтів.
- Є редактор мобільної версії сайту.
- Є безкоштовний тариф.
- При зупиненні оплати, не видаляють сайт як мінімум рік.
- Є бібліотека безкоштовних фото.

Мінуси:

- Погано підтягнуті метатеги: Title, Description, Keywords та атрибути для картинок Alt і Title.
- Не дуже добрий тариф без реклами.

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛЦ.467800.003 ПЗ

Арк.

8

Всі зміни, які ви вносите в інтерфейс сайту, зберігаються автоматично. Щоб зробити їх загальнодоступними, досить натиснути на кнопку «Опублікувати». Міняти дизайн і наповнення можна скільки завгодно. Головне не забувати публікувати нову версію сайту після чергового внесення змін.

Ще одна зручна річ - перемикання типу сайту. Для редагування відкривається шаблон односторінкового сайту, але можна переключитися на багатосторінковий сайт зі збереженням контенту. Блоки перетворюються в сторінки, посилання на які виявляться закріпленими в меню.

Ще одна відмінна можливість Site123 - створення мультимовного сайту. У розділі «Параметри» із поточним регіон і мову. Ви можете змінити його або додати новий. Правда, для цього потрібно перейти на один з платних тарифів. Кількість доступних мов залежить від того, який пакет послуг ви придбали. На максимальному тарифі є створення 5 версій сайту для різних регіонів. [3]

- **tilda.com**

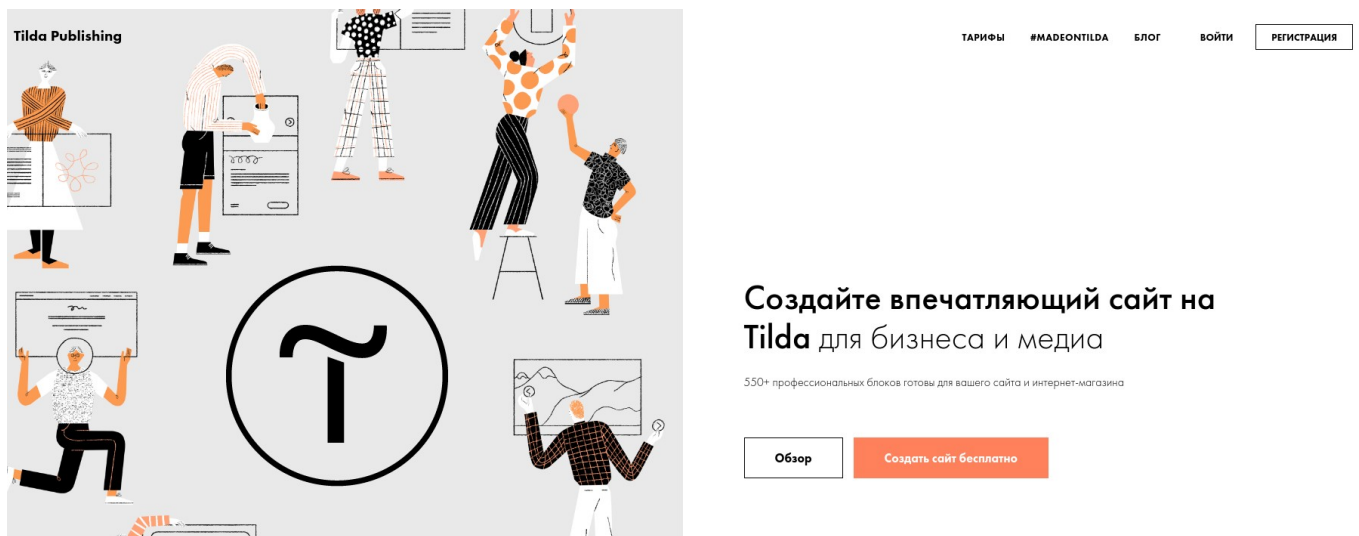


Рис. 1.4 – Знімок екрана веб-інтерфейсу tilda.com.

					ІАЛЦ.467800.003 ПЗ		Арк.
Зм.	Арк.	№ докум.	Підп.	Дата			9

Tilda - це конструктор, призначений для створення Лендінгов і простих веб ресурсів, в тому числі навіть нескладних інтернет магазинів. Tilda ділить ринок з такими не менше відомими конструкторами, як WIX, LPGenerator та іншими обширними сервісами. Ці рішення також мають багатий функціонал і можуть запропонувати багато плюсів.

Переваги:

1. Можливість розробити недороге рішення.

Все залежить від побажань користувача по дизайну.

2. Швидкість розробки.

Розробник на Tilda - це програміст, верстальник і дизайнер в одній особі. Проект не буде поетапно проходити ці три рішення, що дозволяє розробити сайт на Тільда лише за тиждень.

3. Можливість створити унікальний дизайн.

Платформа пропонує багатий вибір шаблонів, але якщо є необхідність інтегрувати власний дизайн з нуля, можна створити нульовий блок при розробці. Це надає багато можливостей. Наприклад, можна підключати власні унікальні шрифти.

4. Можливість самостійно управляти контентом.

Додавайте нові товари або статті в блог самостійно. Навчитися цьому ви зможете за пару годин, створивши свої перші 2-3 сторінки — вже буде зрозуміло принцип роботи з редактором Тільди.

5. Адаптивність.

Платформа добре адаптується до всіх пристроїв і виглядає привабливо як на робочому столі, так і на смартфоні. Що ж стосується zero блоків - вони адаптуються фахівцем, і в кінцевому підсумку буде отримано зручний інтерфейс.

6. Техпідтримка.

Будь-які питання, складності і збої швидко вирішуються техпідтримкою.

7. Підключення сервісів для комунікації з клієнтами.

Є можливість підключити різні варіанти комунікації — онлайн-чат, сервіси для e-mail розсилок, або форми для зворотних дзвінків.

8. Миттєві повідомлення в месенджер.

При розробці сайту можна обрати різні методи зв'язку з клієнтом, який хоче, наприклад, залишити замовлення.

9. Інтеграція сервісів оплати для онлайн-платежів.

Недоліки:

1. Необхідність платити за підписку на сервіс. Є мінусом, якщо наприклад користувач не планує вносити зміни на сайт і просто хоче користуватися візитною карткою.

2. Невеликий обсяг сховища хостингу.

Є забезпечення всього 1 гб на один ресурс, з чого пізніше можуть виникнути проблеми.

3. Обмеження для інтернет-магазинів. [4]

- Squarespace.com

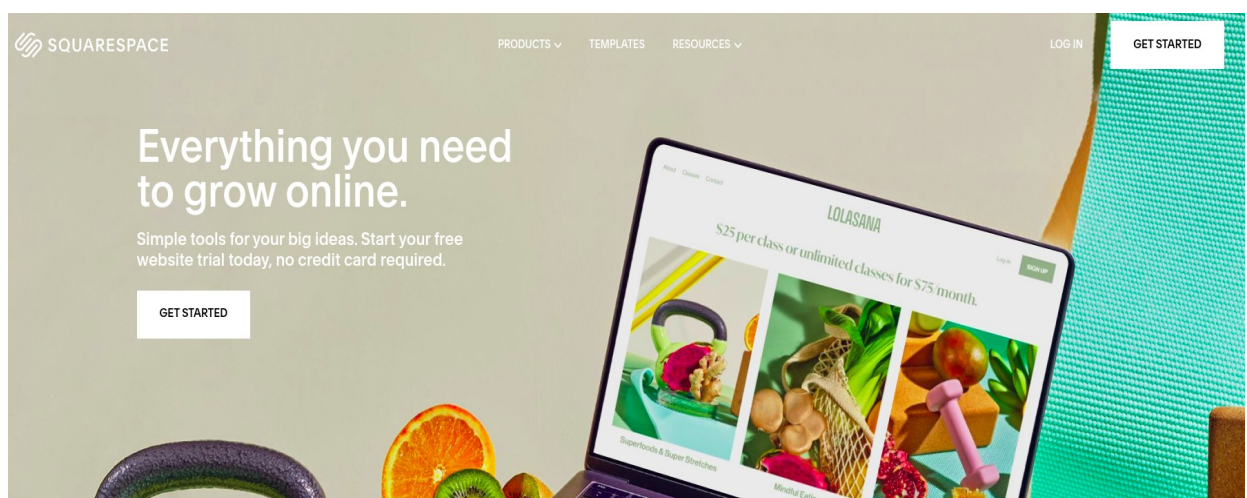


Рис. 1.5 – Знімок екрана веб-інтерфейсу Squarespace.com.

					ІАЛЦ.467800.003 ПЗ	Арк. 11
Зм.	Арк.	№ докум.	Підп.	Дата		

Спочатку Squarespace був тільки на основі SaaS системи управління контентом (CMS), яка запропонувала зручний для розробника конструктор сайтів, платформу для блогів і хостинг. Тепер Squarespace провів деякий час в сфері електронної комерції, щоб зробити його досвідченим суперником з точки зору платформи для електронної комерції. Його платформа називається Комерційна нерухомість, і він пропонує гарні шаблони, простий спосіб відкрити магазин і деякі надійні функціонально.

Squarespace і це комерційне рішення - гідні варіанти для людей, які хочуть почати працювати онлайн. Ви можете використовувати портфоліо Squarespace, щоб знайти місце для себе на сторінках результатів пошуку. Додатково, Squarespace має безліч функцій електронної комерції, які допоможуть вам продавати.

Однак Squarespace НЕ буде ідеальним інструментом для всіх. Наприклад, на відміну від конкурентів, Squarespace ціноутворення не входить до варіантами довгострокового безкоштовного плану. Ви отримуйте тільки 2-тижневий безкоштовний пробний період. [5]

Плюси:

- Він призначений для креативних власників малого бізнесу, Squarespace відмінно підходить для розробки приголомшливих сайтів. Ви можете створити ідеальну домашню сторінку, використовуючи теми, які виглядають приголомшливо. Більш того, вам не потрібно знання HTML або CSS, щоб стрибнути прямо всередину.
- Сильна надійністю і продуктивністю в порівнянні з багатьма іншими розробниками сайтів. Ви можете підтримувати працездатність вашого сайту протягом тривалого періоду часу.
- Безкоштовне доменне ім'я з покупкою: Squarespace пропонує безкоштовне доменне ім'я при покупці річного плану. Поки ви платите за послуги щорічно,

нове зареєстроване доменне ім'я нічого не буде коштувати. Однак, якщо ваш сайт вже розміщений в іншому місці, Squarespace буде стягувати з вас невелику плату за його передачу.

- Безкоштовний SSL: Якщо ви турбуєтесь про безпеку свого бізнесу, то ви в безпеці з Squarespace. Кожен раз, коли ви реєструєте доменне ім'я, ви отримуєте безкоштовну конфіденційність WHOIS і сертифікат SSL, вбудований в вашу покупку. Це значно полегшує уми вашої цільової аудиторії, коли вони взаємодіють з вами.

- Тонни можливостей для продажу в Інтернеті: Якщо вам потрібна інтеграція з соціальними мережами або фантастичні системи онлайн-платежів, Squarespace гарантує, що ви зможете легко монетизувати свій веб-сайт. Ви навіть можете створювати і управляти інвентарем на своєму бекенда. Squarespace не обмежує кількість предметів, які ви можете продати.

- Повний довідковий центр: Можливо, у Squarespace не буде підтримки по телефону, але ви отримаєте цілодобову підтримку по електронній пошті, чат в реальному часі і велику бібліотеку допомоги, щоб почати роботу.

Мінуси:

- Ціноутворення Squarespace високе: Коли ви порівнюєте Squarespace У планах деяких конкурентів на ринку вони можуть здатися дуже високими. Наприклад, у Wix і Weebly є безкоштовні плани, і ви не можете отримати недорогий персональний план у Squarespace.

- На думку деяких оглядачів, час завантаження Squarespace може бути досить повільним. Це реальна проблема, коли вам потрібно, щоб люди швидко переміщалися по корзині покупок, особливо на мобільних пристроях.

- Говорячи про переміщення людей по корзині, у Squarespace не так багато варіантів оплати. Якщо ви блогер, який бажає заробити гроші, або власник сайту електронної комерції, то вам потрібно дотримуватися тільки Stripe.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		13

- Може бути важко орієнтуватися: Створення сайту має бути максимально простим для початківців. Незважаючи на те, що Squarespace докладас великі зусилля для того, щоб ви не були занадто перевантажені, ви, ймовірно, почнете відчувати себе трохи збентеженим процесом як користувач squarespace.
- Обмежувальні шаблони: Хоча Squarespace подбає про те, щоб ваш сайт виглядав чудово з відмінними шаблонами квадратного простору, це може здатися трохи обмежує для побудови з використанням блоків контенту. Ви не можете повністю контролювати, куди йде ваш контент. Це призначено, щоб перешкоджати тому, щоб новачки зробили помилки, але це може привести до обмеженою креативності та більшої кривої навчання для деяких людей.

Висновки до 1 розділу

На мою думку, для побудови такого роду проектів найважливіше — це розробити найбільш зрозумілий для користувача інтерфейс. Треба пам'ятати, що він може не мати часу та настрою розбиратися у всіх необхідних питаннях для розробки сайту. Йому потрібна сама реалізація своєї ідеї, багато прикладів дизайну та відмальовки інформації, щоб було з чого обирати, та не мати дуже схожий інтерфейс, як у його конкурента. Також, важливе значення має SEO. При побудові сайту клієнт, як правило, буде зацікавлений у його просуванні. Якщо проаналізувати усі розглянуті приклади, можна виділити ще деякі проблеми, рішення яких я представляю у своєму проекті.

Я намагаюся приділяти більше уваги таким моментам — простий для входу інтерфейс з багатою функціональністю та вибором дизайнів, можливість просування сайту для користувача, швидкість розробки і оновлення контенту, можливість користувачу експортувати та інтегрувати сайт. Також, важлива, на мій погляд, мобільна версія додатку. Це гарний плюс, коли користувач може у будь-який момент відкрити свій дизайн сайту з телефона та підкоректувати щось, або додати якусь важливу інформацію.

РОЗДІЛ 2. ОПИС МЕТОДИК НЕОБХІДНИХ ДЛЯ ВИРІШЕННЯ

2.1. Перелік основних технологій та бібліотек

Основна частина проекту буде описана за допомогою таких технологій: як основу буде використано фреймворк для веб-розробки — Vue.js. Дизайн та стилі прописано у декількох файлів, блоксхема проекту і основні слайди — реалізовані на Figma, перенесено до веб браузеру — за допомогою HTML та CSS. Логіку описано на JavaScript. Базу даних буде розгорнуто за допомогою Firebase.

2.2. Детальний опис кожної технології

Vue.js Фреймворк

Це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. Також, Vue повністю підходить і для створення складних односторінкових додатків. [6]

Одне із найкращих переваг, яке надає Vue — це принцип реактивності, який дуже ускорює процес розробки. Фреймворк Vue.js налаштований таким чином, щоб автоматично додавати можливість реактивності для будь-якого з створюваних в моделі даних властивостей: data, computed property, props і т.п. Важливо позначити, що саме браузер не працює на пряму з фреймворком. Так само, як наприклад, з JavaScript ECMA Script 6+. Щоб розробити будь-який не простий додаток нам потрібні бібліотеки, які треба уставлювати окремо за допомогою **node package manager (npm)**.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		16

npm складається з двох частин: CLI (інтерфейс командного рядка) - засіб для розміщення і скачування пакетів, та онлайн-репозиторії, що містять JS пакети. Структуру сховища npmjs.com можна уявити, як центр виконання замовлень, який отримує товари (npm-пакети) від продавців (автори пакетів) і поширює ці товари серед покупців (користувачі пакетів). У центрі виконання замовлень (npmjs.com) в якості персональних менеджерів для кожного покупця працює армія вомбатов (npm CLI). [7]

Тепер, коли ми розібралися із npm, за його допомогою ми можемо встановити програму, яка перекомпілює наш ісходний код з усіма залежностями у звичайний JavaScript, який напряду працює з браузером. Для цього можна використовувати наприклад gulp, webpack. У нашому випадку я буду користуватися сервісом Vue cli.

Vue CLI

Він забезпечує безперебійну роботу різних інструментів збірки, встановлює розумні значення за замовчуванням. У той же час, залишається сучасною функціональністю конфігурації кожного інструменту без необхідності вилучення конфігурації в окремий файл. Гарна перевага Vue cli — можна дуже швидко розгорнути локальний сервер для розробки, який у реальному часі перезбирає ісходний код для браузера. За допомогою цього ми можемо користуватися ще одною важливою частиною vue — компонентним підходом. [8]

Компоненти - це екземпляри Vue зі своїм ім'ям, які ми можемо використовувати необхідну кількість разів. Компоненти беруть ті ж опції що і new Vue, такі як data, computed, watch, methods, хукі життєвого циклу. Єдиними винятками будуть кілька специфічних для кореневого примірника опцій, наприклад el.

					ІАЛЦ.467800.003 ПЗ	Арк. 17
Зм.	Арк.	№ докум.	Підп.	Дата		



Рис. 1.6 – Схема компонентів фреймворку Vue.js.

Чому рекомендовано користуватися компонентами у розробці Vue додатка — це ізначально допомагає оптимізувати код, робити його доступнішим для розуміння, та звідси ускорює усі етапи побудови логіки. Найважливіше — ізначально робити такі компоненти, які зможуть бути використані у різних частинах проекту, та для різних потреб. Наприклад, як кнопки. Одна і та ж сама кнопка може бути використана будь-де. Але вона може підтягувати різні параметри, коректувати дизайн та відображення відповідно до них. Для цього у компонентах використовують такі поняття, як props, slots. [3]

Бібліотека Vue Router

Vue.js має повноцінну систему маршрутизації, яка дозволяє зіставляти запити до додатка з певними компонентами. Кожен маршрут визначає властивість path, яке представляє шлях запиту, і властивість component - компонент, який буде обробляти запит з цього шляху. [10]

Модуль Vue Router не використовується дефолтно у Vue.js додатках, його треба інтегрувати окремо. Після інтеграції, треба прописати роуту маршруту — тобто список сторінок, із відповідними компонентами, які будуть використовуватися у поточному проекті. З'являється можливість владеності

компонентів, тобто дочерних. Кожен елемент маршрутів має свій шлях, можна указати ім'я та деякі інші властивості. Також ми можемо використовувати спеціальні функції переводу або редіректу на інші сторінку всередині будь-якого компонента чи модуля.

Також маршрутизатор має свої хуки життєвого циклу: `beforeRouteEnter`, `beforeRouteUpdate`, `beforeRouteLeave` і тд. Завдяки їм можна реалізовувати перевірки при оновленні сторінки, або виконувати функції при переході між маршрутами.

За замовчуванням `vue-router` працює в режимі хеша - він використовує хеш URL для симуляції повного URL-адреси, що дозволяє уникнути перезавантаження сторінки при зміні URL. Але можна обійтися без хеша, використовуючи режим (mode) `history`, який працює з API `history.pushState` для досягнення тієї ж мети.

JavaScript

Це мова програмування, який дає можливість реалізовувати складну поведінку веб-сторінки. Кожен раз, коли ви бачите веб-сторінку, вона не тільки відображає статичну вміст, а й робить більше - своєчасно відображає оновлення контенту, виводить інтерактивні карти, і тд. [11]

Кожна веб-сторінка, яка завантажується в браузер, має свій власний об'єкт `document`. Інтерфейс документа служить точкою входу для отримання вмісту веб-сторінки (всього DOM - дерева, включаючи такі елементи як `<body>` і `<table>` (en-US)), а також забезпечує функціональність, яка є глобальною для документа, наприклад, для отримання URL-адреси сторінки або створення нових елементів в документі).

Об'єкт document може бути отриманий з різних API: Найчастіше використовується прямий доступ до об'єкту document зі сценаріїв scripts які завантажуються документом. (Цей же об'єкт доступний як window.document.) Або через властивість contentDocument об'єкта iframe.

Як відповідь responseXML об'єкта XMLHttpRequest. Доступ до документа може бути отриманий з елемента або вузла через властивість ownerDocument.

Залежно від виду документа (тобто HTML або XML) у об'єкта document можуть бути доступні різні API. Всі об'єкти документів реалізують інтерфейс Document (і отже Node і EventTarget інтерфейси). Таким чином основні властивості і методи, описані на цій сторінці, доступні для всіх видів документів. В сучасних браузерах деякі документи (тобто ті, які містять контент text / html) також реагують HTMLDocument інтерфейс.

ECMA Script 6

JavaScript створювався як скриптова мова для Netscape. Після чого він був відправлений в ECMA International для стандартизації (ECMA - це асоціація, діяльність якої присвячена стандартизації інформаційних і комунікаційних технологій). Це призвело до появи нового мовного стандарту, відомого як ECMAScript. Подальші версії JavaScript вже були засновані на стандарті ECMAScript. Простіше кажучи, ECMAScript - стандарт, а JavaScript - найпопулярніша реалізація цього стандарту.

ES - це просто скорочення для ECMAScript. Кожне видання ECMAScript отримує аббревіатуру ES з подальшим його номером. Всього існує 8 версій ECMAScript. ES1 була випущена в червні 1997 року, ES2 - в червні 1998 року, ES3 - в грудні 1999 року, а версія ES4 - так і не була прийнята. Не будемо заглиблюватися в ці версії, так як вони морально застаріли, а розглянемо тільки останні чотири.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		20

ES5 був випущений в грудні 2009 року, через 10 років після виходу третього видання. [12] Серед змін можна відзначити:

- підтримку суворого режиму (strict mode);

ES - це просто скорочення для ECMAScript. Кожне видання ECMAScript отримує аббревіатуру ES з подальшим його номером. Всього існує 8 версій ECMAScript. ES1 була випущена в червні 1997 року, ES2 - в червні 1998 року, ES3 - в грудні 1999 року, а версія ES4 - так і не була прийнята. Не будемо заглиблюватися в ці версії, так як вони морально застаріли, а розглянемо тільки останні чотири.

Останньою глобальною новою версією була версія ECMAScript6. Розглянемо, основні моменти, що до нього пішли:

- аксесор getters і setters;
- можливість використовувати зарезервовані слова в якості ключів властивостей і ставити коми в кінці масиву;
- багаторядкові рядкові літерали;
- нову функціональність в стандартній бібліотеці;
- підтримку JSON
- новий тип даних symbol
- асинхронність: обгортка функції - async, await
- багато нових методів роботи з масивами та об'єктами: reduce, replace, map, filter, sort і тд.
- цикли перебору даних: forEach
- стрілочні функції (без контексту)
- нові шалони строчних елементів
- класи

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		21

- в шаблонних рядках можна оголошувати рядки за допомогою ``` (зворотних лапок). Шаблонні рядки можуть бути багаторядковими, також можуть інтерполювати;
- `let` і `const` - альтернативи `var` для оголошення змінних. Додана «тимчасова мертва зона»;
- ітератор і протокол ітерації тепер визначають спосіб перебору будь-якого об'єкта, а не тільки масивів. `Symbol` використовується для присвоєння ітератора до будь-якого об'єкта;
- додані функції-генератори. Вони використовують `yield` для створення послідовності елементів. Функції-генератори можуть використовувати `yield *` для делегування в іншу функцію генератора, крім цього вони можуть повертати об'єкт генератора, який реалізує обидва протоколи;
- додані Проміс. [13]

Інтерфейс `Promise` являє собою обгортку для значення, невідомого на момент його створення. Він дозволяє обробляти результати асинхронних операцій так, як якщо б вони були синхронними: замість кінцевого результату асинхронного методу повертається свого роду обіцянку (дослівний переклад слова "промис") отримати результат в певний момент в майбутньому.

`Promise` може перебувати в трьох станах:

- очікування (`pending`): початковий стан, не виконаний і не відхилений.
- виконано (`fulfilled`): операція завершена успішно.
- відхилено (`rejected`): операція завершена з помилкою.

При створенні промис знаходиться в очікуванні (`pending`), а потім може стати виконаним (`fulfilled`), повернувши отриманий результат (значення), або

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		22

відхиленням (rejected), повернувши причину відмови. У будь-якому з цих випадків викликається обробник, прикріплений до Проміс методом then. (Якщо в момент призначення обробника проміс вже виконаний або відхилений, обробник все одно буде викликаний)[14].

Об'єкт Promise створюється за допомогою ключового слова new і свого конструктора. Конструктор Promise приймає в якості аргументу функцію, звану "виконавець" (executor function). Ця функція повинна приймати дві функції-колбека в якості параметрів. Перший з них (resolve) викликається, коли асинхронна операція завершилася успішно і повернула результат свого виконання у вигляді значення. Другий колбек (reject) викликається, коли операція не вдалася, і повертає значення, яке вказує на причину невдачі, найчастіше об'єкт помилки.

Розглянемо особливості функцій у JavaScript.

В JavaScript у кожній виконуваній функції, блоку коду і скрипта є пов'язаний з ними внутрішній (прихований) об'єкт, званий лексичним оточенням LexicalEnvironment, який складається з двох частин:

- Environment Record - об'єкт, в якому як властивості зберігаються всі локальні змінні (а також деяка інша інформація, така як значення this).
- Посилання на зовнішнє лексичне оточення - тобто те, яке відповідає коду зовні (зовні від поточних фігурних дужок).

"Змінна" - це просто властивість спеціального внутрішнього об'єкта: Environment Record. «Отримати або змінити змінну», означає, «отримати або змінити властивість цього об'єкта».

Розглянемо, як глобальне лексичне оточення змінюється в процесі виконання коду:

1. На початку скрипта лексичне оточення пусте.
2. З'являється визначення змінної let phrase. У неї немає присвоєного значення,

тому присвоюється undefined.

3. Перемінної phrase присвоюється значення.

4. Мінлива phrase змінює значення.

При запуску функції для неї автоматично створюється нове лексичне оточення, для зберігання локальних змінних і параметрів виклику.

Коли код хоче отримати доступ до змінної - спочатку відбувається пошук у внутрішньому лексичному оточенні, потім в зовнішньому, потім в наступному і так далі, до глобального. Якщо змінна не була знайдена, це буде помилкою в strict mode. Без strict mode, для забезпечення сумісності, присвоювання неіснуючої змінної створює нову глобальну змінну з таким ім'ям. [15]

Функція отримує поточне значення зовнішніх змінних, тобто, їх останнє значення

Функція отримує поточне значення зовнішніх змінних, тобто, їх останнє значення. Старі значення змінних ніде не зберігаються. Коли функція хоче отримати доступ до змінної, вона бере її поточне значення зі свого або зовнішнього лексичного оточення.

Firestore

В разі збереження інформації щодо авторизованих користувачів та контенту, який кожен користувач буде розробляти під себе, будемо використовувати інтерфейс Firestore. Firestore надає в режимі реального часу базу даних та бекенд як службу. Ця служба надає розробникам застосунків API, який дозволяє синхронізувати дані застосунків між клієнтами та зберігати їх у хмарі Firestore. Компанія також надає клієнтські бібліотеки, які дозволяють інтеграцію із застосунками Android, iOS, JavaScript/Node.js, Java, Objective-C, Swift. База даних також доступна через REST API та прив'язки до декількох сценаріїв JavaScript, таких як Vue, AngularJS, React, Ember.js та Backbone.js. REST API використовує протокол подій із сервером, який є інтерфейсом для створення

					ІАЛЦ.467100.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		24

HTTP-з'єднань для отримання push-повідомлень від сервера. Розробники, які використовують Realtime Database, можуть захищати свої дані за допомогою правил безпеки, що застосовуються на сервері. Cloud Firestore, яка є наступною генерацією Firebase Realtime Database, була випущена у бета-версії.

Firebase Hosting — це статичний та динамічний веб-хостинг, який було запущено 13 травня 2014 року. Він підтримує хостинг статичних файлів, таких як CSS, HTML, JavaScript та інші файли, а також динамічну підтримку Node.js через Cloud Functions. Служба передає файли через мережу доставки контенту (CDN) за допомогою протоколу HTTPS та шифрування SSL. Firebase підтримує Fastly, CDN, щоб забезпечити підтримку CDN Firebase Hosting. Компанія стверджує, що хостинг Firebase виріс із запитів клієнтів; розробники використовували Firebase для своєї бази даних в режимі реального часу, але вони потребували місця для розміщення їхнього вмісту. [16]

Firebase також надає багато інших можливостей, але ми не будемо використовувати їх у проєкті.

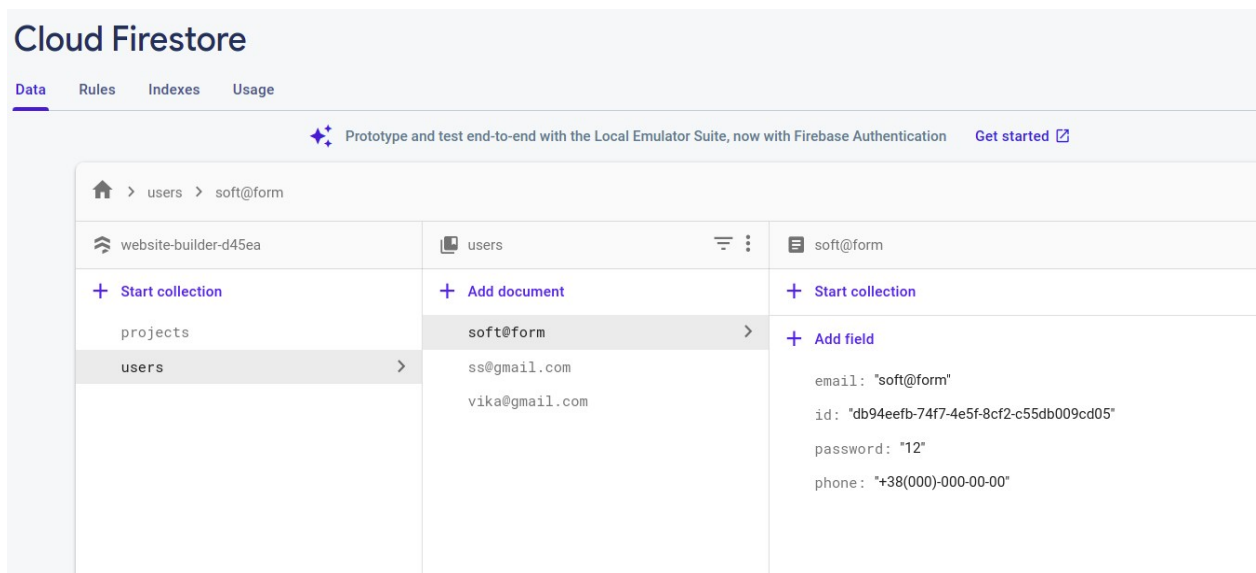


Рис. 1.7 – Знімок екрана веб-інтерфейсу Firebase.

Порівняння фреймворку Vue з іншими

- Vue та React

У React коли стан компонента змінюється, воно запускає повторну отрисовку всього поддерева компонента, починаючи з себе. Щоб уникнути непотрібної повторної відтворення дочірніх компонентів, вам потрібно або використовувати `PureComponent`, або реалізовувати `shouldComponentUpdate` всюди де це можливо. Вам також може знадобитися використовувати незмінні (immutable) структури даних, щоб зробити зміни вашого стану більш зручними до оптимізації.

У Vue залежності компонента автоматично відслідковуються під час відтворення, тому система точно знає, які компоненти дійсно необхідно повторно малювати при зміні стану. Кожен компонент можна розглядати як має `shouldComponentUpdate`, автоматично реалізований для вас, без обмежень для вкладених компонентів.

В цілому, це усуває необхідність цілого класу оптимізацій продуктивності розробника, що дозволяє більше зосередитися на побудові самого проекту.

У Vue у нас також є `render`-функції і навіть підтримка JSX, тому що іноді вам потрібні ці можливості. Проте, за замовчуванням ми пропонуємо шаблони як більш просту альтернативу. Будь-валідний HTML також буде дійсним додатком в міру його масштабування.

У React абсолютно все - це JavaScript. Також React повністю реалізує компоненти за допомогою JSX.шаблоном Vue, і це призводить до виникнення кількох переваг:

Для багатьох розробників, які працюють з HTML, шаблони просто більш природні для читання і написання. Саме ця перевага може бути кілька суб'єктивним, але якщо це робить розробника більш продуктивним, то перевага очевидна.

- HTML-шаблони полегшують поступову міграцію існуючих додатків для використання можливостей реактивності Vue.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		26

- Це також полегшує дизайнерам і менш досвідченим розробникам розбиратися і вносити доопрацювання в поточну кодову базу.
- Ви можете навіть використовувати препроцесори, такі як Pug (раніше відомий як Jade), щоб створювати ваші шаблони у Vue.

Якщо ви шанувальник підходу CSS-in-JS - багато популярних бібліотеки підтримують Vue (наприклад, styled-components-vue і vue-emotion). Головною відмінністю між React і Vue тут буде те, що за замовчуванням стилізація Vue виконується через знайомі теги style в однофайлових компонентах.

В якості ще одного важливого відмінності між React і Vue можна згадати той факт, що всі додаткові бібліотеки Vue, включаючи бібліотеки для керування станом додатки і для роутінга (серед інших завдань), офіційно підтримуються в актуальному відповідно до ядром бібліотеки. React, навпаки, вважає за краще віддати ці питання на відкуп спільноті, тим самим створюючи більш фрагментовану екосистему. Втім, як уже зауважувалося раніше, в силу популярності React, його екосистема значно ширший, ніж у Vue.

Також плюси розробки на Vue CLI були розглянуті раніше.

- Vue та Angular

Vue значно простіше AngularJS, як в сенсі API, так і в сенсі архітектури. AngularJS має жорстке думка щодо структури вашого застосування, в той час як Vue проявляє гнучкість і є більш модульним рішенням. Хоча це і робить Vue придатним для більшої різноманітності проектів, ми розуміємо і те, що коли рішення вже прийняті за тебе, можна відразу почати програмувати, і в цьому є свої переваги.

AngularJS використовує двунаправлене зв'язування даних між областями видимості, в той час як Vue концентрується на однонаправленому потоці даних між компонентами. Це дозволяє полегшити розуміння потоку даних в

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		27

нетривіальних додатках.

Vue чіткіше поділяє директиви і компоненти. Директиви призначені тільки для інкапсуляції низькорівневих маніпуляцій з DOM, в той час як компоненти являють собою повноцінні автономні об'єкти, зі своєю власною логікою даних та подання. У AngularJS директиви роблять всю роботу, а компоненти всього лише певний тип директив.

Vue продуктивніше AngularJS. Крім того, через відсутність dirty-checking, оптимізувати Vue-додатки набагато-набагато простіше. AngularJS сповільнюється при збільшенні кількості спостерігачів, оскільки кожен раз при зміні чого-небудь в області видимості всі ці спостерігачі повинні бути перезапущені. Крім того, цикл може повторитися кілька разів перед стабілізацією, оскільки реакція спостерігачів може спровокувати таке оновлення. Користувачам AngularJS нерідко доводиться вдаватися до вельми езотеричних технік для обходу цих труднощів, а в деяких випадках оптимізація і зовсім стає неможливою.

Для Angular потрібно використовувати TypeScript, оскільки майже вся його документація та навчальні ресурси засновані на TypeScript. TypeScript має свої очевидні переваги - перевірка статичних типів може бути дуже корисна для великих додатків, і може додати продуктивності розробникам, які працюють на Java і C #. Однак не всі хочуть використовувати TypeScript.

Часто для невеликих додатків введення системи типів може призвести до більшого збільшення накладних витрат ніж збільшення продуктивності розробки. У таких випадках вам краще скористатися Vue, так як використовувати Angular без TypeScript може бути складним. Нарешті, хоча Vue і не так глибоко інтегрований з TypeScript як Angular, але надає офіційні декларації типів і офіційний декоратор для тих, хто хоче використовувати TypeScript з Vue. [17]

					ІАЛЦ.467800.003 ПЗ	Арк. 28
Зм.	Арк.	№ докум.	Підп.	Дата		

Бібліотека i18n

Інтернаціоналізація і локалізація - це засоби адаптації комп'ютерного програмного забезпечення до інших мов, регіональним відмінностям і технічним вимогам конкретного ринку.

Інтернаціоналізація - це процес розробки програмних додатків, які потенційно можуть адаптуватися до різних мов і регіонах без інженерних змін. Локалізація - це процес адаптації інтернаціоналізувати програмного забезпечення для певного регіону або мови шляхом додавання локальних компонентів і перекладеного тексту. [18]

Локалізація (яка потенційно виконується кілька разів для різних мов) використовує інфраструктуру або гнучкість, що забезпечується інтернаціоналізацією.

Саме цю функціональність реалізовує бібліотека i18n.

В процесі інтегрування модуля бібліотеки, потрібно зробити такі моменти:

- Указати активний регіон локалізації проекту.
- Обрати резервний мовний стандарт локалізації проекту.
- Обрати каталог, в якому зберігається інформація про локалізацію проекту. [19]
- Прописати json файли для підтягування значень активної мови проекту.

Бібліотека надає швидкий та комфортний інтерфейс для прописань необхідної кількості мов проекту, динамічного підтягування даних по назві з json файлів. Є можливість змінювати активну мову всього інтерфейсу лише однією функцією. При імпортуванні модуля у основний js файл проекту, бібліотекою створюється локальне сховище із перемінними, та функцією для змін. Можна зберігати лише значення активної мови у сховищі Vuex, Local Storage або у базі даних. Тоді при рендері інтерфейсу буде автоматично підтягуватися це значення, та будуть відображатися правильні дані.

Бібліотека Vuelidate

Валідація форм - одна з найважливіших завдань на сайті. У процесі розробки доводиться затверджувати дані на присутність, на збіг по паттерну, створювати асинхронні валідації, застосовувати валідації тільки після зняття фокуса з поля або перед відправкою форми.

Навіть у простому прикладі користування формою на флеймворку Vue виникають такі проблеми:

- Браузери показують лише одну помилку за один раз. Користувачеві доведеться намагатися відправити форму кілька разів, щоб побачити всі свої помилки.
- Помилки відображаються користувачеві тільки після спроби відправки форми. Щоб змінити цю поведінку доведеться написати ще тонну коду: викликати у кожного елемента за подією blur функцію `reportValidity()`.
- Для стилізації полів введення існують тільки псевдокласи: `valid` і `invalid`, немає можливості зловити стан, коли дані в поле введення невірні, але ще не взаємодіяв з ним.
- Кожен браузер відображає помилки валідації по-своєму, вони можуть виглядати непривабливо в вашому дизайні.
- Валідацію API у Vue.js незручно використовувати: доводиться зберігати елементи в `$refs`.

Для вирішення цих та інших проблем розробляють бібліотеки валідацій.

Існує дві найпопулярніші бібліотеки для Vue.js флеймворку. Я розгляну їх різницю, і обосную, чому для проекту було обрано останню.

1. Vee-validate.

Це бібліотека для валідацій, що з'явилася за часів Vue.js першої версії, має велике співтовариство і використовується у великій кількості проектів.

Вага бібліотеки - 31 КБ (minified + GZIP), що в півтора рази більше самого Vue.js.

					ІАЛЦ.467800.003 ПЗ	Арк. 30
Зм.	Арк.	№ докум.	Підп.	Дата		

Це пов'язано це з тим, що бібліотека містить в собі відразу багато функціоналу:

- 35 вбудованих валідаторів
- Переклад помилок на англійську мову
- Директиви для валідацій
- Компоненти для валідацій
- Підтримує 41 мову для помилок

Особливість бібліотеки в тому, що вона пропонує два варіанти валідації: із допомогою директиви та із допомогою компонентів.

Підхід за допомогою директиви дуже простий: треба додати на поле введення директиву v-validate, якій передасте список валідаторів. Стан валідації і рядки помилок потім можна отримати з полів fields і errors всередині компонента.

Підхід за допомогою компонентів: надається можливість обрати один із двох компонентів. ValidationProvider - компонент, в якому ви описуєте валідації і в який обертаєте поле введення. ValidationObserver - компонент, який об'єднує прапори валідацій декількох ValidationProvider'ов.

Мінуси vee-validate:

Перша і найбільша проблема — це її розмір. Бібліотека не підтримує tree-shaking і вибіркове додавання валідаторів. В вашому модулі завжди буде присутні 2 компонента і директива валідацій, переклад на англійську мову і 35 валідаторів.

Друга проблема - через підходу, заснованого на підписці на події, можуть виникати проблеми при інтеграції з іншими бібліотеками, які теж змінюють поведінку полів введення.

Третя проблема - більш суб'єктивна - переклади помилок загальні і некрасиві, не відображають реальної суті.

2. Vuelidate

Бібліотека Vuelidate з'явилася у відповідь на проблеми з підходами, які містить в собі бібліотека vee-validate. Vuelidate не має ні обробників подій для полів, ні переказів для помилок валідації. Вона вимагає від вас тільки одного - описати валідації в об'єкті `validations`. Потім вона сама створить `computed`-поле `$ v` з прапорами валідації полів і функціями для зміни цих прапорів.

Бібліотека Vuelidate з'явилася у відповідь на проблеми з підходами, які містить в собі бібліотека vee-validate. Vuelidate не має ні обробників подій для полів, ні переказів для помилок валідації. Вона вимагає лише одного - описати валідації в об'єкті `validations`. Потім вона сама створить `computed`-поле `$ v` з прапорами валідації полів і функціями для зміни цих прапорів.

Завдяки простому підходу до валідації, Vuelidate важить всього 3.4 КБ (minified + GZIP), майже в 10 разів менше vee-validate.

Кастомні валідатори можна швидко і легко описувати за допомогою функцій. Якщо необхідно, щоб параметри вашого валідатора потрапили в об'єкт `$ params`, потрібно використовувати спеціальний хелпер `withParams`.

Поля з декількома помилками займають дуже багато місця і виглядають монструозність — але вирішується це створенням окремого компонента для відображення валідацій.

Також бібліотека реалізовує роботу із асинхронними валідаціями, та валідаціями колекцій.

З проблем можна виділити хіба що порівняну невелику кількість валідаторів з коробки, часто доводиться писати свої функції для валідації. [20]

У висновку, я обрала другу бібліотеку Vuelidate через модельний підхід к описанню валідацій та менший розмір моділів, також більше можливостей для вводу своїх можливостей.

Висновки до 2 розділу

Vue.js дуже комфортний для розробки фреймворк, який допоможе нам правильно побудувати структуру проекту, оптимізувати код, не втрачати швидкість при компілюванні до зрозумілого користувачу інтерфейсу. Також він добре функціонує з дуже багатою кількістю сторонніх бібліотек, у усіма є свої грамотно написані документації, тому просто і швидко включити їх у свій проект.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		33

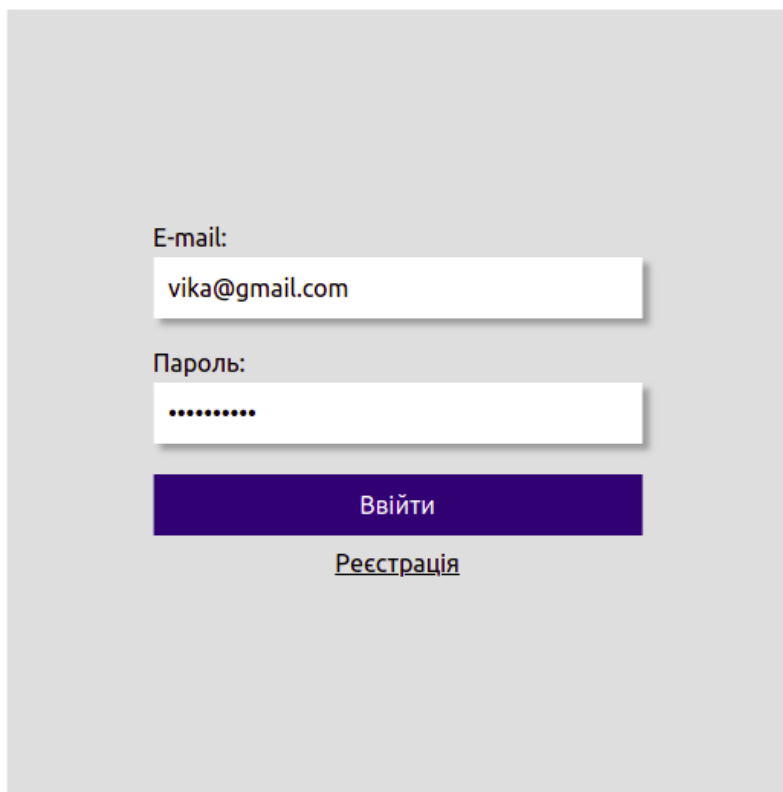
РОЗДІЛ 3 ОПИС ПРОЕКТУ

Створений проект побудовано за допомогою бібліотеки Vue cli та фреймворку Vue.js. Vue cli у свою чергу має під копитом збірник webpack, який переводить ESMA Script 6, та інші сторонні бібліотеки, у стандартний JavaScript файл. Vue компоненти, в яких таких приступі template, script та style також переводяться за допомогою збірника у дефолтні темплейти HTML, зі стилями CSS, до яких підключаються скрипти. Vue cli реалізовує комфортний та швидкий інтерфейс розробки даного проекту у реальному часі, отмальований на локальному сервері.

3.1 Структура проекту

Розглянемо поступово кожний блок проекту з його функціональністю.

Сторінка авторизації.



The screenshot shows a login form on a light gray background. It includes an 'E-mail:' label, a text input field containing 'vika@gmail.com', a 'Пароль:' label, a password input field with masked characters, a blue 'Ввійти' button, and a 'Реєстрація' link below it.

Рис. 3.1 – Знімок екрана сторінки авторизації веб-інтерфейсу.

Перша сторінка, на яку при заході на додаток потрапляє користувач — це сторінка авторизації. Він не може побачити свій контент, якщо не авторизується.

Для успішного отримання інформації потрібно лише два поля — логін або email, та пароль. Ці дані раніше мали бути введені та збережені на сторінці реєстрації, яку ми розглянемо пізніше. Якщо користувач не має полей для входу, або перший раз на сайту, він може натиснути на кнопку “Реєстрація”, та відбудеться перехід на відповідну сторінку.

При вводі обох полей, користувач нажимає на кнопку “Ввійти” та відбувається наступне:

1. За допомогою бібліотеки Vuelidate буде перевірено наявність та формат даних.
2. При неуспішній проверці — буде виведено на екран необхідне повідомлення та авторизація не пройде далі. При успішній валідації — буде виконано необхідний екшен у сховище Vuex з передачею введених даних.
4. У функції екшену setUserInfo буде створена опція з поля “users” бази даних Firebase, та відправлен get запит на базу даних у промісі.
5. Проміс може відповісти або помилку серверу, тоді авторизація не відбудеться і буде виведено текст помилки, або повернути інформацію щодо опції користувача з введеним логіном.
6. Якщо була успішно отримана інформація — можливий варіант, якщо не існує користувача з введеним логіном, або існує, але паролі не співпадають. В даних випадках авторизація буде прервана та буде виведено відповідний опис помилки. В іншому випадку — інформація буде збережена до сховища Vuex.
7. Відбудеться переклад до стартової сторінки (/home).

Сторінка реєстрації.

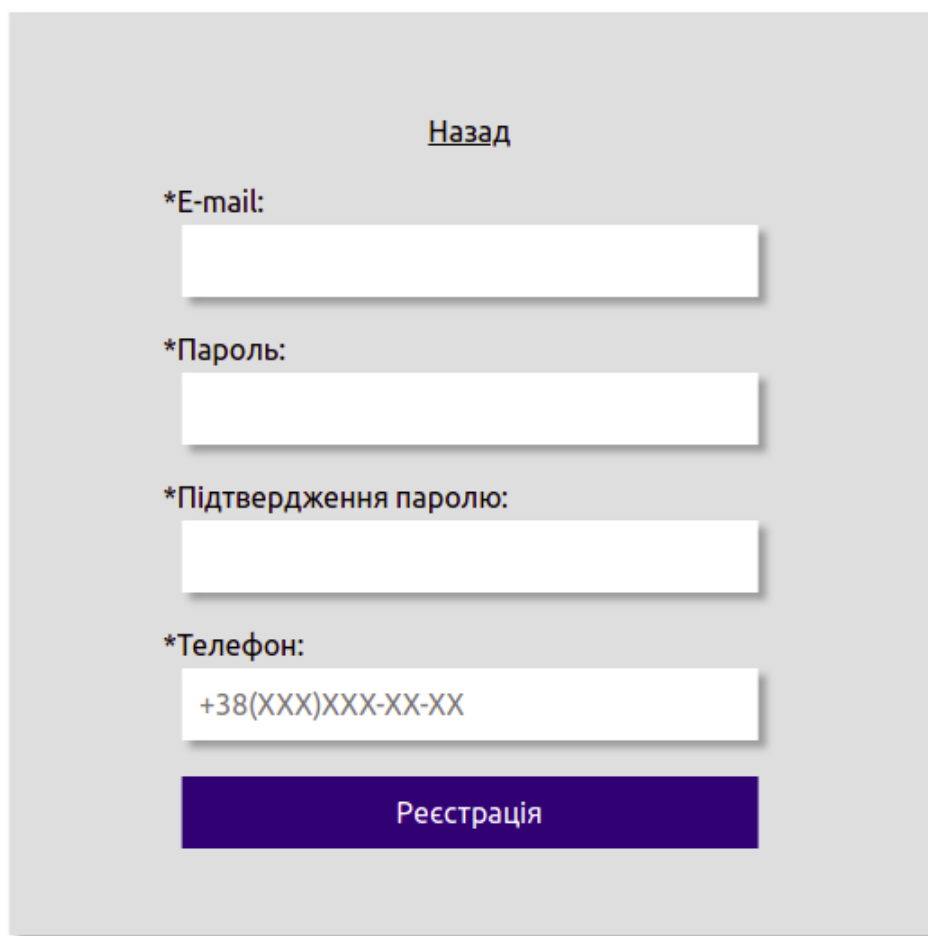


Рис. 3.2 – Знімок екрана сторінки реєстрації веб-інтерфейсу.

Мета цього блоку — авторизувати нового користувача. Чотири основних поля — email для логіну, телефон, пароль та підтвердження паролю. При кліку на кнопку “Назад” користувач може повернутися на сторінку авторизації. При кліку на кнопку “Реєстрація” відбудеться валідація або перевірка всіх присутніх полей за допомогою бібліотеки Vuelidate — сторонньої, окремо підключеної бібліотеки. Також окремо перевіряється повна ідентичність паролів. При помилках наявності полей або несхожості паролей — користувач отримає повідомлення, з текстом опису помилки.

При успішній перевірці відбувається наступне:

1. Створюється новий об'єкт з введеними полями користувача: email, password, phone, та полем id, яке генерується кожний раз нове за допомогою модуля uuid.
2. В базу даних Firebase в опцію “users” за допомогою set запросу зберігається раніше сформований об'єкт.
3. При успішному зберіганні даних у Firebase, дані користувача також зберігаються у сховище Vuex (відбувається мутація setUserInfo) та зберігається лише email у Local Storage (сховище браузеру).
4. Аналогічно, збору даних користувача по логіну і паролю з бази даних, викликається функція updateUserInfo, мета якої — підтягнути якої підтягнути усі проекти, які раніше створював користувач. У Firebase це опція “projects”. Кожен користувач у базі даних зберігається із власним id, який був сгенерований під час успішної реєстрації. Цей ідентифікатор передається у функцію updateUserInfo.
4. Відбувається перехід до стартової сторінки конкретного користувача.

Стартова сторінка користувача.

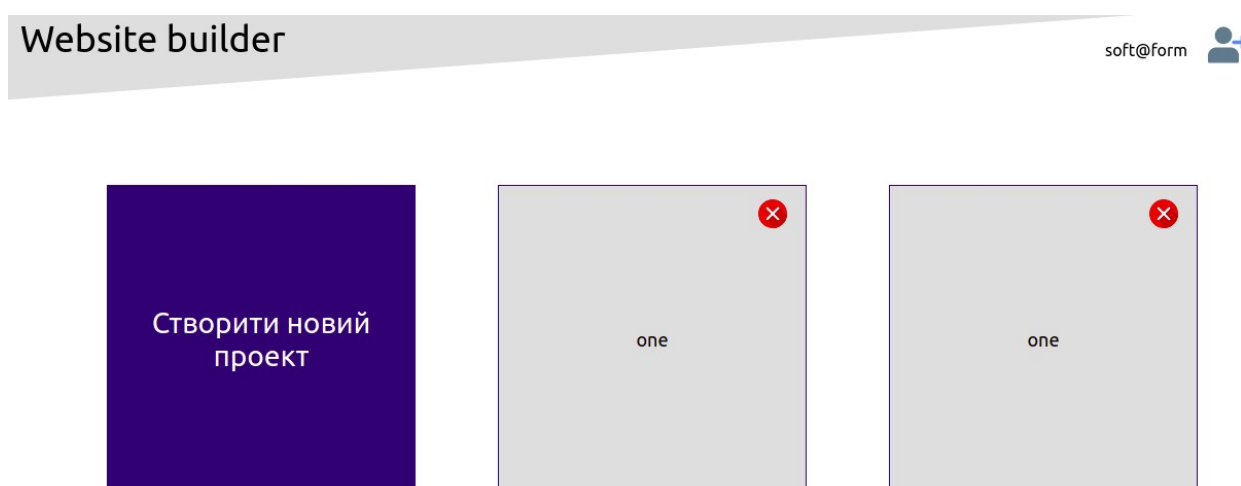


Рис. 3.3 – Знімок екрана сторінки реєстрації веб-інтерфейсу.

При переході на сторінку користувач буде бачити або список своїх раніше созданих проектів, або лише один блок — “Створити новий проект”.

У верхній частині (шапці) з’являється, окрім логотипу, інформація логіну користувача та кнопка, яка несе в собі функціонал виходу із інтерфейсу. При кліку по неї очищується всі дані поточного користувача і відбувається перехід назад на сторінку авторизації. Такий вигляд шапки зберігається на всіх останніх сторінках (тільки для авторизованого користувача).

Щодо основного блоку сторінки — користувач може обрати один із списку своїх раніше зафіксованих проектів, або створити новий. Також він може видалити свої проекти.

Видалення відбувається функцією `deleteUserProject` по ідентифікатору (`id`) обраного проекту. У базу даних з опцією “`projects`” посилається запит `delete`, та при успішній відповіді проміса, у який огорнений цей запит, в інтерфейсі буде динамічно перемальовані проекти користувача.

Якщо користувач обирає “Створити новий проект” відбувається перехід до сторінки редагування проектів (`/draft`).

Якщо користувач обрав один із раніше існуючих проектів, перехід буде здійснено до тієї ж сторінки, але замість пустого контенту він має побачити раніше створену сторінку сайту.

Усховище `VueX` зберігається ідентифікатор активного проекту. Якщо був створений новий проект — `id` проекту буде сгенерований при першому додаванні будь-якого контенту.

Сторінка редагування.

При переході на сторінку редагування, створюючи новий проект, або обираючи вже раніше існуючий, у основний блок відображення контенту буде не підтягнута або підтягнута відповідно існуюча HTML розмітка, яка була раніше збережена у базі даних.

Права частина (найбільша) відображує змінний контент.

Ліва частина — це меню, за допомогою якого користувач може змінювати різні атрибути та властивості обраного елемента.

Розглянемо основний функціонал на прикладі створення простого блоку.

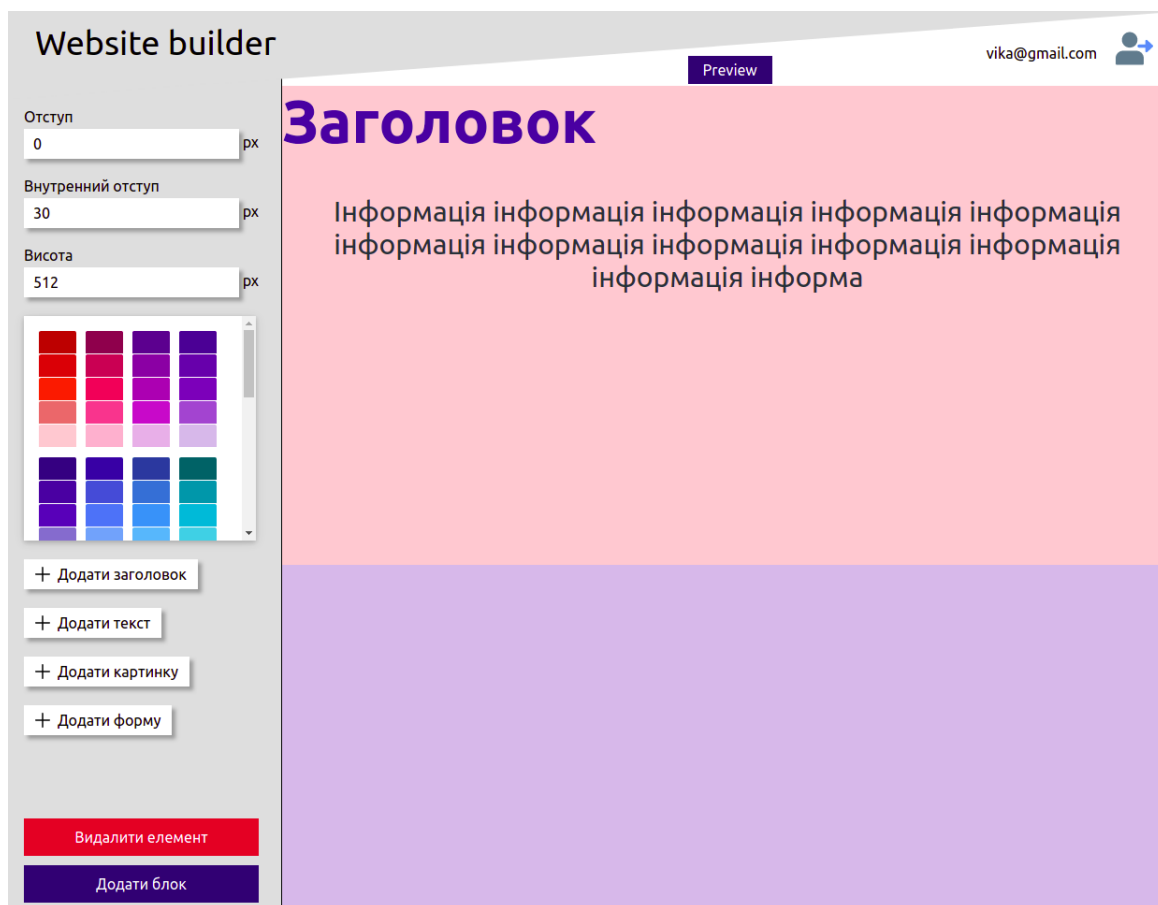


Рис. 3.4 – Знімок екрана сторінки редагування веб-інтерфейсу.

При додаванні будь-якого елемента, викликається відповідна функція у екшенах Vueх. Створюється необхідний тег з властивостями, якщо вони потрібні, та додається у HTML розмітку, яка тот час відображується у основному блоці. Кожному новому елементу присвоюється новий сгенерований ідентифікатор у атрибут id, тазначається подія кліку. За допомогою цього id та події, ми можемо у будь-який час у основному блоці обирати усі елементи. У екшенах буде відлавлюватися функція зміни ідентифікатору. Та динамічно відносно його

та відносно типу елемента, буде відображатися необхідний компонент зміни контенту у боковому меню.

Наприклад, якщо ми додаємо заголовок, а потім звичайну текстову інформацію, і те, і те являється текстовими елементами, тому у меню буде підтягуватися один компонент, але з невеликими різницями стосовно типу елемента. У зміні самого блоку вже будуть інші атрибути на необхідності, тому це інший компонент. А такі атрибути, як наприклад, відступи та внутрішні відступи (margin, padding) можуть бути віднесені до усіх елементів.

У меню у нижній частині завжди закріплені дві основні кнопки екшенів — додати новий блок та видалити активний елемент. Видалення, так само, вносить зміни у основний блок, видаляючи його із розмітки.

При всіх змінах: додаванні нових елементів, зміни існуючих, видалення відбувається автоматичне зберігання активного проекту.

І, на відміну, від вже існуючих рішень зберігання проекту також відбувається навіть у офлайн режимі. Наприклад, якщо користувач змінює контент, і раптом відключилась мережа, він може продовжувати свою роботу — зміни будуть зберігатися як локально, так і у базі даних.

У верхній частині сторінки редагування, біля шапки, також фіксується кнопка прев'ю, функціонал якої полягає у тому, щоб користувач міг перейти на сторінку предпросмотру розробленого проекту (/preview). При переході не треба передавати даних, або збереження контенту, так як контент зберігається автоматично при кожному додаванні елемента або зміні його властивостей.

Також у шапку (header) буде додано перемикач для зміни активної локалізації або мови проекту. Користувач зможе працювати для себе на комфортній йому мові. На разі доступні лише дві основні мови, але у наступному їх буде більше.

Приклад зміни текстового елемента у меню:

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		40

Також у шапці з'являється кнопка для експорту створеного проекту. Це надає можливість інтегрувати проект зі власною метою, не прив'язуючись до системи розробки.

При перевірці — далі є варіант лише повернутися назад до зміни сайту, якщо треба щось доробити, або експортувати у власні файли.

3.2 Опис модулів та класів додатку

Проект включає у себе декілька основних частин: сховище Store, маршрутизатор router, папку views, яка включає усі сторінки, які будуть динамічно підтягуватися окремими маршрутами, папку з компонентами Vue (глобально зареєстровані та локально у окремих файлах), папку style з основними стилями для проекту, також допоміжні міксіни та плагіни, та файли, які об'єднують усі частини проекту. Файл package.json — описує усі властивості та залежності проекту до інших бібліотек, та основну інформацію щодо самого проекту. При розробці Vue cli сервіс надає деякі можливості — команда npm і встановлює пакети для усіх описаних залежностей в package.json, команда npm run serve запускає локальний сервер, npm run build — збирає усі файли та модулі ES6 та переводить їх у звичайний JavaScript.

Папка public включає у себе основний файл index.html, папку з іконками, які будуть використовуватися, та папку з файлами для конкретної інтерфейсної мови — українська, англійська і тд. Ці файли будуть динамічно підтягнуті бібліотекою i18n та застосовані у всіх словах або речень, які будуть прописані через об'єкт цієї бібліотеки.

Розглянемо детальніше кожну модульну частину.

Основний файл main.js

Мета цього файлу — імпортувати у себе усі сторонні бібліотеки та модулі.

```
import Vue from "vue";
import App from "./App.vue";
import i18n from "./plugins/i18n";
import upperFirst from "lodash.upperfirst";
import camelCase from "lodash.camelcase";
import Vuelidate from "vuelidate";
import router from "./router/index";
import store from "./store";
import { mask } from "vue-the-mask";
//import vuetify from '@plugins/vuetify';
```

Рис. 3.7 – Знімок екрана основні імпорти проекту.

Перше, що ми повинні імпортувати — це сам Vue для того, щоб розробити інстанс об'єкту Vue, і вже до нього підключати будь-які компоненти та модулі.

```
const requireComponent = require.context(
  "./components/app",
  false,
  /App[A-Z]\w+\.(vue|js)$/
);

requireComponent.keys().forEach(fileName => {
  const componentConfig = requireComponent(fileName);
  const componentName = upperFirst(
    camelCase(
      fileName
        .split("/")
        .pop()
        .replace(/\.?\.w+$/, "")
    )
  );
  Vue.component(componentName, componentConfig.default || componentConfig);
});
```

Рис. 3.8 – Знімок екрана глобальна реєстрація компонентів.

У main.js зробимо автоматичну обробку базових компонентів, які будуть підтягуватися глобально у всьому проекті.

Імпортуємо маршрутизатор (router) та сховище даних (store), структуру яких розглянемо пізніше, та деякі інші сторонні бібліотеки.

Тепер розглянемо реєстрування підключення проекту до бази даних Firebase.

Перед тим, як підключити Firebase до проекту, треба скористуватися клієнт-сервісом <https://console.firebase.google.com/>. Треба створити нову базу даних, якій при після створення буде присвоєно id. Це ідентифікатор нам знадобиться при інтегруванні бази даних до конкретного проекту.

З інтегруванням Firebase у додаток нам допоможе бібліотека vuefire.

Пропишемо дефолтний компонент створення інстансу vuefire, де вкажемо id бази даних (website-builder-d45ea), який був збережений раніше. Також пропишемо модуль із назвою .firebaseerc, де такою вкажемо ідентифікатор.

Після цього з'являється можливість експортувати прописаний інстанс бази даних із поточним id. Це необхідно для того, щоб імпортувати його у інших різних модулях та компонентах, що у свою чергу, надає можливість працювати за допомогою методів та промісів із даними Firebase.

У той же час, для зберігання нової інформації, та для її подальшого отримання, необхідно створити потрібні опції. У проекті будемо використовувати дві основні:

1. Users

До цієї опції бази даних будемо відносити усю інформацію користувачів: зберігання даних нового користувача, перевірка на існування і коректність даних при авторизації, отримання загальної інформації щодо користувача при успішній авторизації.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		44

2. Projects

Ця опція буде отримувати і віддавати інформацію щодо проектів поточного користувача, реалізовувати роботу з ними.

Методи, якими ми будемо користуватися:

- get

Використовується для отримання інформації.

- set

Виконується при зберіганні будь-яких даних.

- delete

Видалення елементів конкретної опції.

Реалізація бази даних проекту:

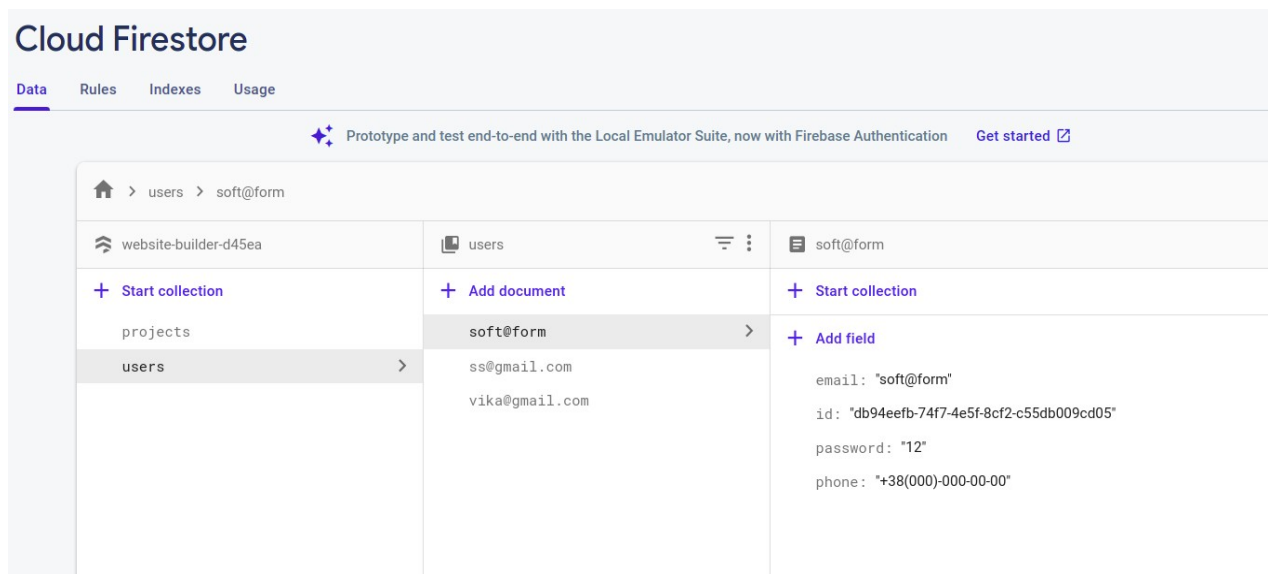


Рис. 3.9 – Знімок екрана основні опції бази даних.

Зареєструвавши у додатку та описавши базу даних на клієнті Firebase, ми можемо напряму із компонентів нашого інтерфейсу зберегати нові дані, або дізнаватися інформацію від вже існуючих.

Компонент App.vue

У цьому файлі ми імпортуємо основний файл scss зі стилями CSS, відмалюємо шапку (header) інтерфейсу, за допомогою глобальної реєстрації компонентів та додаємо template відмальовки vue-router.

Ключ, який буде динамічно підтягнуто маршрутизатором позначає сторінку, на якій буде знаходитися користувач. Для більш гармонійного сприйняття додаємо тег transition, який додає анімацію переключення між сторінками, за допомогою стилей.

Компонент App.vue по суті лише обгортка для усіх внутрішніх елементів додатку. Тому його метод, при рендері (наприклад, при оновленні будь-якої сторінки) будуть викликатися завжди.

Сам метод — це функція оновлення інформації користувача `updateUserInfo()`. У базу даних посилається запит у опцію “users” з ідентифікатором логіну поточного користувача. При його визові можливі два наступні варіанти:

1. Якщо не передається логін, та він не збережений у локальному сховищі браузера Local Storage, або активний логін присутній, та база даних при запиті відповідає помилку, автоматично відбувається перенаправлення до сторінки авторизації.
2. Якщо присутній логін та в запиті від бази даних повертається інформація авторизованого користувача, після нього відпрацьовується також функція `setUserProjects` для оновлення кількості та вмісту контенту створених користувачем раніше проектів.

Основний об’єкт Vue, який описується у файлі `main.js`, буде ссилатися на темплейт з `id #app` у базовому файлі `index.html` та буде відмалювати основний компонент `App.vue`.

Маршрутизатор vue-router

Спочатку імпортуємо інстанс з самої бібліотеки vue-router, та створюємо наш об'єкт, який буде використовуватися у проекті. В масиві routes прописуємо основні сторінки, які будуть необхідні для відмалювки. Кожна сторінка включає у себе назву роутера, сам шлях та силку на компонент Vue, який буде підтягуватися при переході. Експортуємо router, щоб можна було його імпортувати у файлі main.js та підключити до основного компонента Vue.

```
import Vue from 'vue';
import VueRouter from 'vue-router';

Vue.use(VueRouter);

function lazyLoadView(view) {
  return () => import(`@/views/${view}.vue`);
}

const routes = [
  {
    path: "/",
    name: "Login",
    component: lazyLoadView("Login")
  },
  {
    path: "/registration",
    name: "Registration",
    component: lazyLoadView("Registration")
  },
  {
    path: "/home",
    name: "Home",
    component: lazyLoadView("Home")
  },
  {
    path: "/draft",
    name: "Draft",
    component: lazyLoadView("Draft")
  },
  {
    path: "/preview",
    name: "Preview",
    component: lazyLoadView("Preview")
  }
];

const router = new VueRouter({
  mode: 'history',
  routes,
});

export default router;
```

Рис. 3.10 – Знімок екрана основні шляхи роутеру проекту.

Vue компоненти

Усі компоненти додатку поділяються на декілька груп:

- /views

Компоненти цієї директорії існують для відображення сторінок відповідно до назви елементу Vue Router:

1. Login.vue — компонент для сторінки авторизації. У темплейті включає у себе лише форму для вводу даних користувачем.

- /components/App

Компоненти цієї директорії реєструються глобально у основному файлі main.js. Не треба імпортувати окремо.

- /components/Draft

Включає у себе необхідні компоненти для роботи із сторінкою редагування контенту.

Розглянемо детальніше кожний файл.

1. Компонент Sidebar.vue

Базовий контейнер бокового меню сторінки редагування. Включає у себе фіксовану верхню частину — дефолтні поля для обозначення відступів (margin, padding), які можуть бути задані для будь-якого елементу під час редагування; фіксовану нижню частину — кнопки екшенів додавання нового блоку та видалення активного елементу; та змінну основну частину. Основна частина залежить від типу активного на редагуванні елементу.

2. Групи компонентів із назвою “Change” + тип активного елементу у верхньому реєстрі.

Ці компоненти автоматично підтягуються у родовому компоненті меню.

<component :is="setActiveComponent" />

Наприклад, якщо тип активного елементу — text (заголовок, текстова інформація і тд) назва компоненту буде — ChangeText.vue. Якщо тип — block, компонент — ChangeBlock.vue

3. Компонент MainContent.vue

Це основна частина сторінки редагування — відображення створеного контенту.

Компонент має лише один блок із ідентифікатором (id="main"). При першій отрисовці на хуці життєвого циклу mounted відбувається перевірка наявності контенту. Наприклад, якщо користувач вже працював раніше над поточним проектом, та вирішив продовжити, при переході на сторінку редагування, відбудеться запит щодо отримання проекту по id (який фіксується при виборі зі списку проектів), інформація зберігається у окремих змінних сховища Vuex. Звідси, у компоненті MainContent.vue метод getDraftContent відобразить отриману HTML розмітку.

- /components/

Усі останні компоненти, які не відносяться до Draft, використовуються на інших сторінках, та потребують пофайлового імпорту, та реєстрації у необхідних місцях.

Локалізація

Інтерфейс підтримує різні мови відображення контенту. Зберігання і зміна активної мови реалізується за допомогою бібліотеки i18n. Вона прописана у окремому модулі проекту i18n.js, який підтягує автоматично json файли із директорії /public/langs. У кожному окремому json файлі прописані ключ — назва ключу об'єкту, на яке буде ссилатися кожне необхідна позиція із тексту, та значення — переклад слова відносно активної мови.

Ця бібліотека надає можливість дуже легко винести будь-який перемикач у додаток, та одним виводом функції повністю переключити мову всього інтерфейсу.

На даний момент прописано лише дві мови інтерфейсу — українська та англійська.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		49

Сховище vuex

Так само, як і при реєстрації маршрутизатора, імпортуємо бібліотеку vuex та підключаємо основний об'єкт сховища до компонентів Vue. Однією з гарних характеристик vue-store є можливість поділити все сховище на необхідні модулі, в кожному з яких будуть свої змінні значення, мутації, гетери та екшени.

Сховище проекту поділяється на основний файл index.js, який безпосередньо створює інстанс Vuex, та імпортує у себе два основних модуля:

1. draft.js

Цей модуль створений для реалізації всієї логіки редагування контенту поточним користувачем.

Включає у себе такі основні функції (actions):

- Сберегання поточного редагованого проекту у локальне сховище та у базу даних.
- Установка активних подій для отриманого існуючого раніше контенту.
- Додавання усіх можливих блоків, елементів, тегів і тд. (Створення елементів у document, установка необхідних стилей, атрибутів, індивідуального ідентифікатору, назначення події для подальшої можливості редагування).
- Зміна активного елементу, установка нового ідентифікатору та типу елементу у сховищі.
- Обновлення або установка будь-яких властивостей стилей елементу по назві властивості та ідентифікатору елементу.
- Зміна значення.
- Видалення елементу по ідентифікатору.

2. users.js

Цей модуль для реалізації функціоналу роботи з користувачами та базою даних.

Включає у себе такі основні функції (actions):

- Регістрація нового користувача.
- Авторизація існуючого користувача.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		50

- Отримання особистої інформації користувача.
- Отримання інформації щодо існуючих проектів поточного користувача.
- Редірект на інші сторінки відповідно до активних даних користувача.
- Сбери́гання нового або існуючого редагованого проекту користувача.
- Видалення проекту користувача.
- Вихід із інтерфейсу.

Стилі CSS

Стилі прописані у проекті за допомогою препроцесору SCSS у директорії `/src/assets/style`.

Препроцесор SCSS надає багато можливостей щодо швидкішої розробки та стилізації елементів. За його допомогою можна легко імпортувати будь-які файли у необхідні компоненти, або у інші модулі SCSS. Надається можливість користуватися змінними, міксінами, вкладенністю стилей. Набагато скорочує код.

Основний файл `main.scss` включає у себе основні стилі, які відносяться до спільних елементів проекту, та імпортує модуль `helpers.scss`.

Модуль `login.scss` імпортований пезпосередньо у компоненті `Login.vue`, та містить лише стилі цього компоненту.

Модуль `variables.scss` імпортований глобально у файлі конфігурації сборки `vue.config.js`.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		51

Висновки до 3 розділу

Було реалізовано основний функціонал для створення сайту користувачем. При авторизації створювалася опція з новим полем по ключу логіна в базі даних Firebase, також присвоювався новий uuid користувача. При авторизації порівнювалися дані, введені в форму зі збереженими опціями баз даних. При успішній авторизації користувач попадав на стартову сторінку, де мог обрати раніше створений проект, або створити новий. Додавання або зміна контенту відбувається за допомогою бокового сайдбару, який динамічно підтягує компоненти для кожного типу елемента окремо. Збереження контенту відбувається постійно у локальному форматі, та у базу даних, при наявності інтернету.

При розробці проекту, я намагалася підсумувати недоліки та слабкі сторони існуючих та розглянутих у другому розділі рішень.

Моменти, які було вирішено у моїй розробці:

- Офлайн зберігання даних.

Це надасть користувачу додатковий захист власних даних.

- Можливість користування інтерфейсом та редагування контенту з мобільних пристроїв.

- Підготовка до SEO просування.

Можливість змінювати теги, прописувати title, description атрибути.

- Можливість експортувати повний створений проект.

Надає можливість розробити та інтегрувати, при необхідності, у іншу систему.

- Простий і інтуїтивно зрозумілий інтерфейс для користувачей різного рівня підготовки.

- Швидкість розробки та оновлення контенту.

Завдяки розробки на бібліотеці Vue cli присутня висока швидкість користування.

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		52

Висновки

В разі виконаної роботи було проаналізовано багато подібних на ринку рішень. Складені переваги, недоліки до деяких розглянутих прикладів, зроблено висновки.

Якщо підводити ітог, для користувача завжди важливий вибір та багата функціональність. Не всі користувачі мають час на розробку власного дизайну та потім сайту, не всі бажають спілкуватися напрямку із дизайнерами та розробниками. Потрапляючи на додатки такого типу, користувач шукає найбільш вигідну пропозицію, як в плані фінансовому, так і в плані реалізації. Тому в мене була основна ціль — розробити багатофункціональний проект, із вирішенням основних проблем, які присутні на деяких прикладах. До них відносяться: проблеми SEO просування, відсутність мобільної версії, просто входу непідготовленого користувача, швидкість роботи самого інтерфейсу, інтеграція своїх рішень і стилей, можливість експортування створеного проекту.

Щодо розробки, було ознайомлено з флеймворком Vue.js та з його багатьма можливостями та сторонніми бібліотеками. Було зрозуміло, як створювати окремо дизайн та UX логіку спочатку, та відтворення її у процесі розробки. Також було опрацьовано можливості маршрутизатора та сховища Vue Router та Vuex, створення бази даних за допомогою Firebase, робота з нею, отримання та зберігання даних.

Щодо розробки — в основі проекту закладено роботу із DOM елементами сторінок за допомогою JavaScript, а реалізація повністю предствлена на Vue.js флеймворку. Було ознайомлено із великою кількістю сторонніх бібліотек, які легко інтегруються із флеймворком. Роботу із даними організовано на базі даних Firebase, яка інтегрується за допомогою бібліотеки Vuefire. Маршрутизація реалізовано бібліотекою Vue Router, а локальне зберігання даних — бібліотекою Vuex.

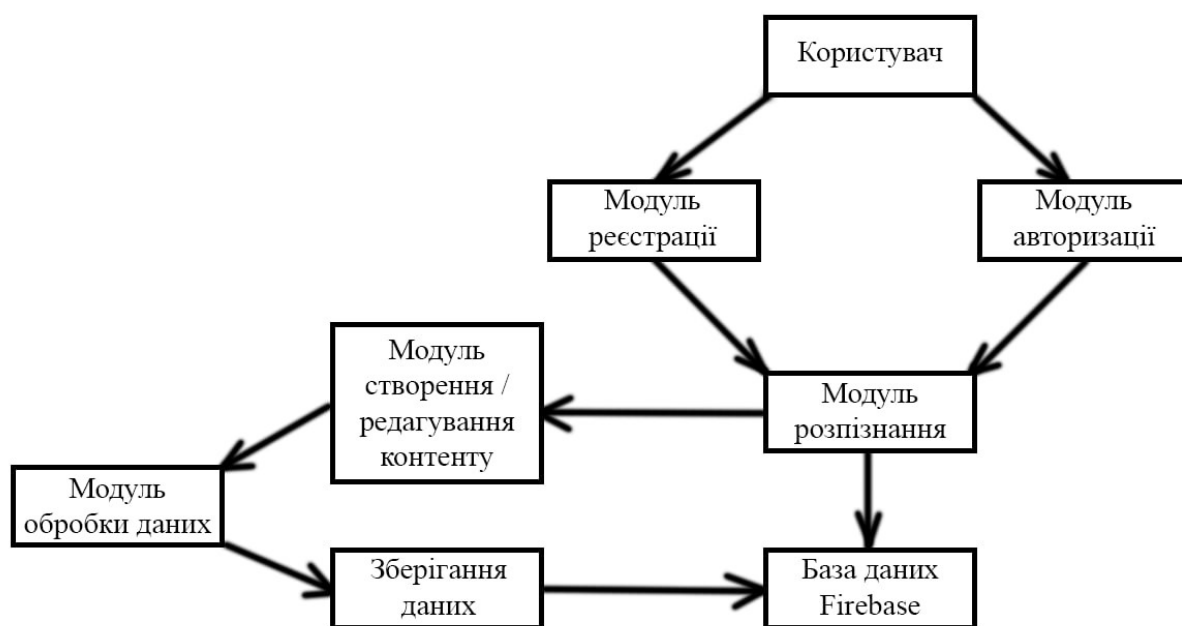
					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		53

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Wix отзывы пользователей. Недостатки и плюсы 2018-2021 [Электронный ресурс] – Режим доступа до ресурсу: <https://filandor.com/wix-otzivi-40/>.
2. Обзоры и мнения экспертов Network Solution [Электронный ресурс] – Режим доступа до ресурсу: <https://www.websiteplanet.com/ru/website-builders/network-solution/#overview>.
3. Обзор конструктора сайтов Site123 [Электронный ресурс]. - 2019. – Режим доступа до ресурсу: <https://konstruktorysajtov.com/site123>.
4. Преимущества и недостатки Tilda [Электронный ресурс] – Режим доступа до ресурсу: <https://artjoker.ua/ru/blog/10-preimushchestv-tildy-dlya-vashego-biznesa/>.
5. Squarespace обзор [Электронный ресурс]. - 2021. – Режим доступа до ресурсу: <https://ecommerce-platforms.com/ru/ecommerce-reviews/ultimate-squarespace-commerce-review>.
6. The Vue Instance [Электронный ресурс] – Режим доступа до ресурсу: <https://vuejs.org/>.
7. Что такое npm? [Электронный ресурс]. - 2020. – Режим доступа до ресурсу: <https://proglib.io/p/chto-takoe-npm-gayd-po-node-package-manager-dlya-nachinayushchih-2020-07-21>.
8. Vue cli введение [Электронный ресурс] – Режим доступа до ресурсу: <https://cli.vuejs.org/ru/guide/>.
9. Основы компонентів [Электронный ресурс] – Режим доступа до ресурсу: <https://ru.vuejs.org/v2/guide/components.html>.
10. Dynamic Route Matching [Электронный ресурс]. - 2020. - Режим доступа до ресурсу: <https://router.vuejs.org/guide/essentials/dynamic-matching.html>.
11. JavaScript [Электронный ресурс]. - Режим доступа до ресурсу: <https://developer.mozilla.org/ru/docs/Learn/JavaScript>.
12. Что такое ECMA Script? [Электронный ресурс]. - 2017. - Режим доступа до ресурсу: <https://tproger.ru/translations/wtf-is-ecmascript/>.
13. ES6 для начинающих [Электронный ресурс]. - Режим доступа до ресурсу: <https://monsterlessons.com/project/series/es6-dlya-nachinayushih>.
14. Promise [Электронный ресурс]. - Режим доступа до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Promise.
15. Замыкания [Электронный ресурс]. - Режим доступа до ресурсу: <https://learn.javascript.ru/closure>.

16. Хостинг Firebase [Электронный ресурс]. - Режим доступа до ресурсу: <https://firebase.google.com/docs/hosting>.
17. Vue vs React vs Angular: який фронтенд-фреймворк обрати? [Электронный ресурс]. - 2020. - Режим доступа до ресурсу: <https://proglib.io/p/vue-vs-react-vs-angular-kakoy-frontend-freymvork-vybrat-2020-08-0>.
18. Локализация программного обеспечения [Электронный ресурс]. - 2021. – Режим доступа до ресурсу: https://ru.wikipedia.org/wiki/Локализация_программного_обеспечения.
19. Локализация на основе компонентов [Электронный ресурс]. Режим доступа до ресурсу: <https://kazupon.github.io/vue-i18n/ru/guide/component.html>.
20. Валидация форм во Vue.js [Электронный ресурс]. - 2019. – Режим доступа до ресурсу: <https://habr.com/ru/post/444900/>.

Додаток 1
до дипломного проєкту
на тему: «Конструктор односторінкових сайтів»



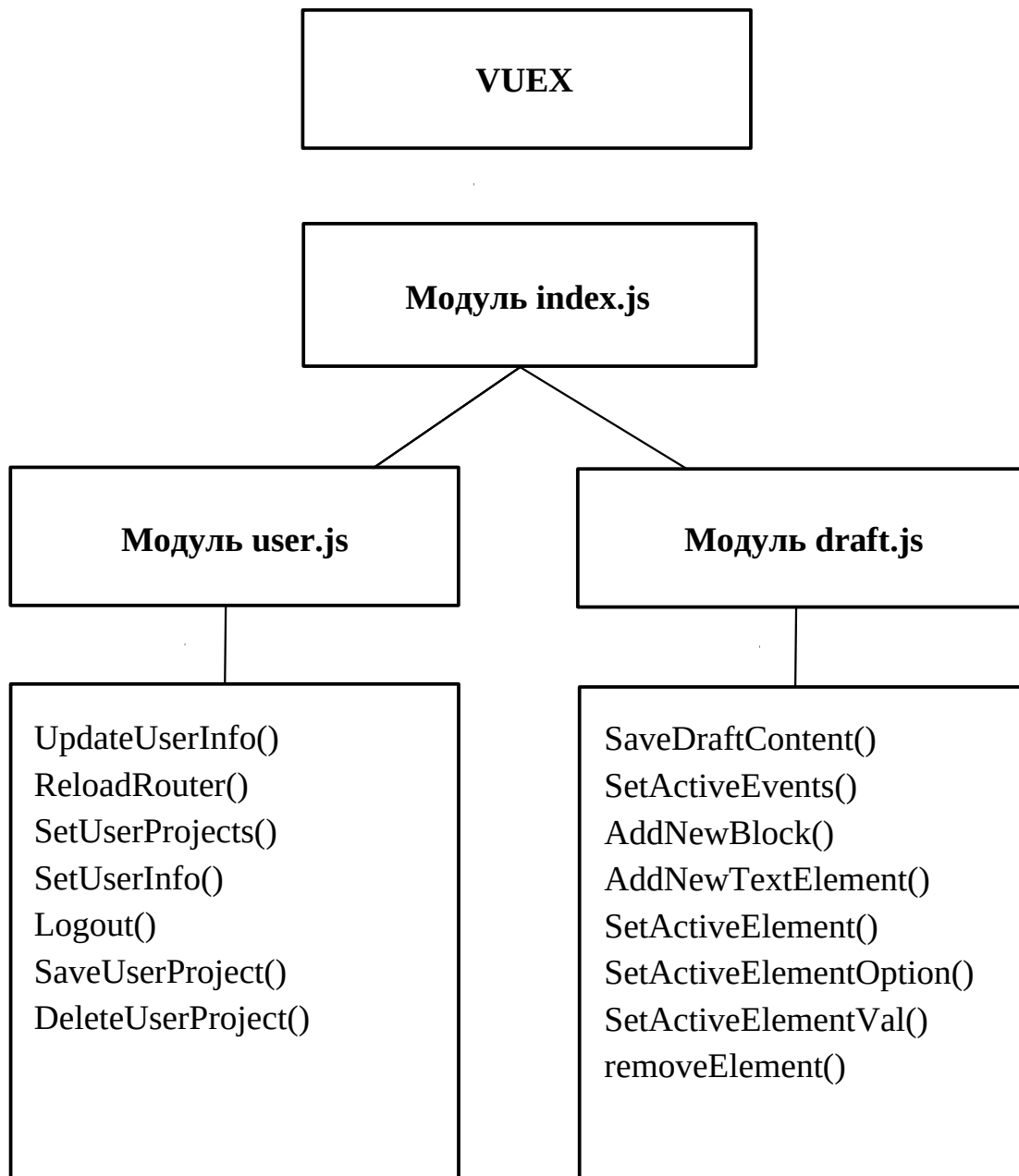
ІАЛЦ.467800.002 Д1

					ІАЛЦ.467800.002 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Трачук В.С.			Конструктордносторінкових сайтів Функціональна схема		
Перевірив		Калюжний О. О.					
Реценз.							
Н. Контр.		Сімоненко В.П.					
Затв.		Стіренко С.Г.					
					Літ.	Аркуш	Аркушів
						1	1
					«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІПЗ-72		

Додаток 2

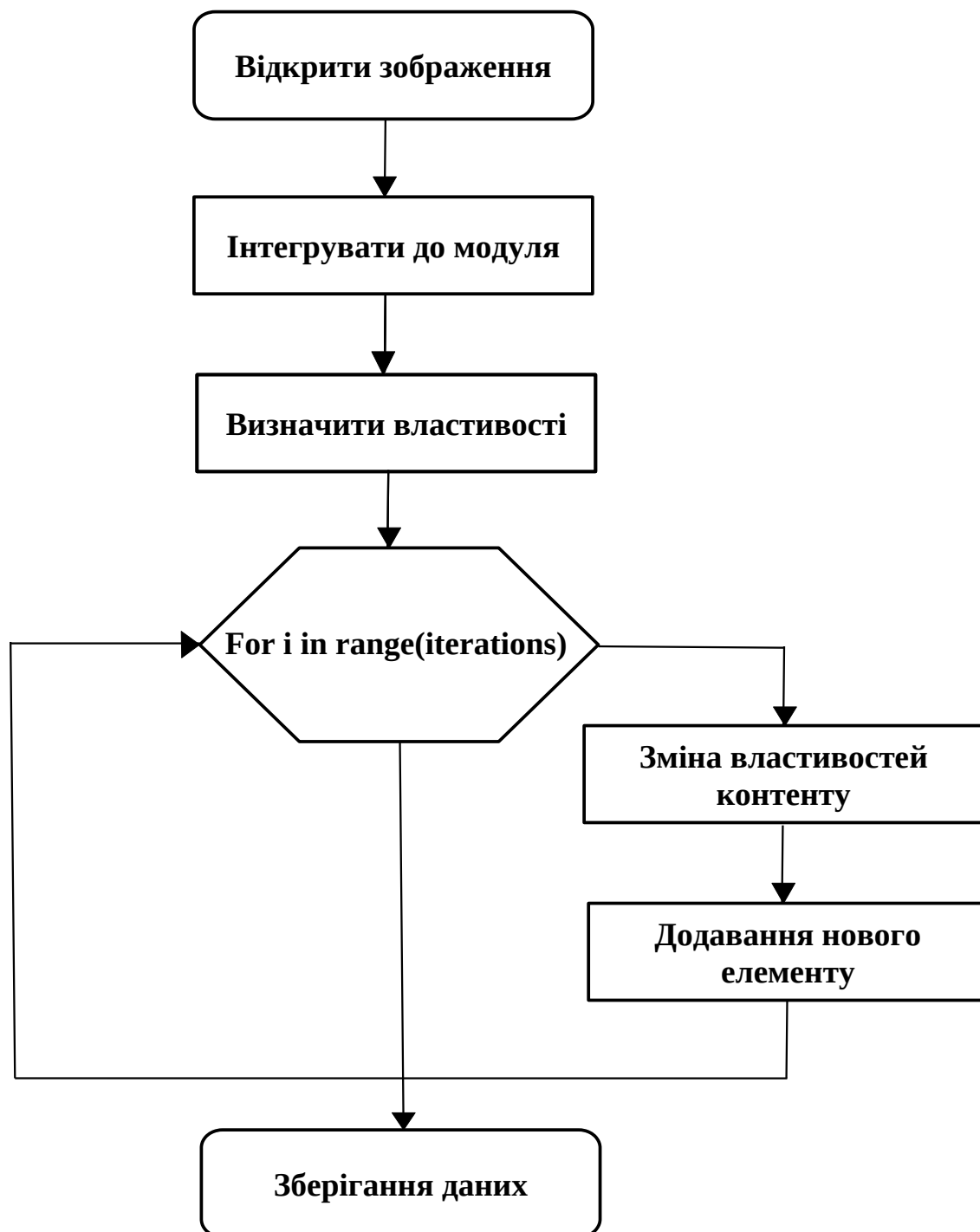
до дипломного проекту

на тему: «Конструктор односторінкових сайтів»



					ІАЛЦ.467800.002 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Трачук В.С.			Конструктор двосторінкових сайтів Діаграма класів	Літ.	Аркуш
Перевірив		Калюжний О. О.					1
Реценз.						«КПІ імені Ігоря Сікорського» ФІОТ, гр. ІПЗ-72	
Н. Контр.		Сімоненко В.П.					
Затв.		Стіренко С.Г.					

Додаток 3
до дипломного проєкту
на тему: «Конструктор односторінкових сайтів»



					ІАЛЦ.467800.002 ДЗ								
Зм.	Арк.	№ докум.	Підпис	Дата									
Розробив		Трачук В.С.			Конструктордносторінкових сайтів Схема алгоритму				Літ.	Аркуш	Аркушів		
Перевірив		Калюжний О. О.									1	1	
Реценз.									«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІПЗ-72				
Н. Контр.		Сімоненко В.П.											
Затв.		Стіренко С.Г.											

Додаток 4

до дипломного проекту

на тему: «Конструктор односторінкових сайтів»

main.js

```
import Vue from "vue";
import App from "./App.vue";
import i18n from "./plugins/i18n";
import upperFirst from "lodash.upperfirst";
import camelCase from "lodash.camelcase";
import Vuelidate from "vuelidate";
import router from "./router/index";
import store from "./store";
import { mask } from "vue-the-mask";
//import vuetify from '@plugins/vuetify';

import { firestorePlugin } from "vuefire";
Vue.use(firestorePlugin);
import firebase from "firebase/app";
import "firebase/firestore";

export const db = firebase
.initializeApp({
projectId: "website-builder-d45ea"
})
.firestore();

const { Timestamp, GeoPoint } = firebase.firestore;
export { Timestamp, GeoPoint };

Vue.use(Vuelidate);
Vue.directive("mask", mask);

const requireComponent = require.context(
"./components/app",
false,
/App[A-Z]\w+\.(vue|js)$/
);

requireComponent.keys().forEach(fileName => {
const componentConfig = requireComponent(fileName);
```

					ІАЛЦ.467800.002 Д4				
Зм.	Арк.	№ докум.	Підпис	Дата					
Розробив		Трачук В.С.			Конструктор дносторінкових сайтів Лістинг програми	Літ.	Аркуш	Аркушів	
Перевірив		Калюжний О. О.						1	1
Реценз.						«КПІ імені Ігоря Сікорського» ФІОТ, гр. ІПЗ-72			
Н. Контр.		Сімоненко В.П.							
Затв.		Стіренко С.Г.							

```

const componentName = upperFirst(
  camelCase(
    fileName
      .split("/")
      .pop()
      .replace(/\.?w+$/, "")
  )
);
Vue.component(componentName, componentConfig.default || componentConfig);
export const bus = new Vue();
window.$bus = bus;

```

```

new Vue({
  el: "#app",
  //vuetify,
  i18n,
  router,
  store,
  render: h => h(App)
});

```

app.vue

```

<template>
<transition name="fade">
<div id="app">
<app-header />
<transition name="fade" mode="out-in">
<router-view :key="$route.path" />
</transition>
</div>
</transition>
</template>

```

```

<script>
export default {
  name: "app",
  created() {
    this.$store.dispatch("updateUserInfo");
  }
};
</script>

```

					ІАЛІЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		64

```
<style lang="scss">
@import "~@/assets/style/main.scss";
</style>
```

router/router.js

```
import Vue from 'vue';
import VueRouter from 'vue-router';

Vue.use(VueRouter);

function lazyLoadView(view) {
return () => import(`@/views/${view}.vue`);
}

const routes = [
{
path: "/",
name: "Login",
component: lazyLoadView("Login")
},
{
path: "/registration",
name: "Registration",
component: lazyLoadView("Registration")
},
{
path: "/home",
name: "Home",
component: lazyLoadView("Home")
},
{
path: "/draft",
name: "Draft",
component: lazyLoadView("Draft")
},
{
path: "/preview",
name: "Preview",
component: lazyLoadView("Preview")
}
];
```

					ІАЛЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		65

```
const router = new VueRouter({
mode: 'history',
routes,
});
export default router;
```

store/index.js

```
import Vue from 'vue';
import Vuex from 'vuex';
import draft from './draft';
import user from './user'
```

```
Vue.use(Vuex);
```

```
export default new Vuex.Store({
modules: {
draft,
user
}
});
```

store/user.js

```
//import { bus } from "@main";
import router from "@router";
import { db } from "@main";
import { uuid } from "vue-uuid";
```

```
export default {
state: {
user_info: null,
user_projects: {},
active_project_id: null
},
getters: {
getUserInfo(state) {
return state.user_info;
},
```

```

getUserId(state) {
  if (state.user_info) return state.user_info.id;
  return null;
},
getUserProjects(state) {
  return state.user_projects;
},
getActiveProjectId(state) {
  return state.active_project_id;
}
},
mutations: {
  setUserInfo(state, payload) {
    state.user_info = payload;
  },
  setUserProjects(state, payload) {
    state.user_projects = payload;
  },
  setActiveProjectId(state, payload) {
    state.active_project_id = payload;
  }
},
actions: {
  reloadRouter({}) {
    setTimeout(() => {
      const active_router = router.currentRoute.name;
      if (active_router !== "Registration" && active_router !== "Login")
        return router.push({ name: "Login" });
    }, 500);
  },
  updateUserInfo({ commit, dispatch }) {
    let email = localStorage.getItem("email");
    if (!email) return dispatch("reloadRouter");
    const docRef = db.collection("users").doc(email);
    docRef
      .get()
      .then(doc => {
        const user = doc.data();
        if (!user) dispatch("reloadRouter");
        delete user.password;

        commit("setUserInfo", user);
        dispatch("setUserProjects");
      })
  }
}

```

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛЦ.467800.003 ПЗ

Арк.

67

```

.catch(error => {
dispatch("reloadRouter");
});
},
setUserProjects({ commit, getters }) {
const user_id = getters.getUserId;
if (!user_id) return;
const docRef = db.collection("projects").doc(user_id);
const projects = [];
docRef
.get()
.then(doc => {
const projects = doc.data();
if (!projects) return commit("setUserProjects", []);
commit("setUserProjects", projects);
})
.catch(error => {
console.log("error get", error);
});
},
setUserInfo({ commit, dispatch }, payload) {
const docRef = db.collection("users").doc(payload.email);
docRef
.get()
.then(doc => {
const user = doc.data();
if (!user) return alert("User not found!");
if (user.password !== payload.password)
return alert("Wrong password");
delete user.password;
localStorage.setItem("email", payload.email);
commit("setUserInfo", user);
dispatch("updateUserInfo");
router.push({ name: "Home" });
})
.catch(error => {
console.log("error get", error);
});
},
logout({ commit }) {
localStorage.removeItem("email");
commit("setUserInfo", null);
router.push({ name: "Login" });
},

```

```

saveUserProject({ getters }) {
  const content = getters.getDraftContent;
  const exist_projects = getters.getUserProjects;
  const id = getters.getActiveProjectId
  ? getters.getActiveProjectId
  : uuid.v4();
  exist_projects[id] = {
    name: "one",
    content
  };
  const ProjectService = db.collection("projects");
  ProjectService.doc(getters.getUserId)
    .set(exist_projects)
    .then(res => {})
    .catch(error => {
      alert("Error save project: ", error);
    });
}
};

```

store/draft.js

```

export default {
  state: {
    draft_content: null,
    active_draft_id: null,
    active_draft_type: null,
    elements_draft: []
  },
  getters: {
    getActiveDraftId(state) {
      return state.active_draft_id;
    },
    getActiveDraftType(state) {
      return state.active_draft_type;
    },
    getDraftContent(state) {
      return state.draft_content;
    }
  },
  mutations: {
    setActiveDraftId(state, id) {
      state.active_draft_id = id;
    },

```

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛІЦ.467800.003 ПЗ

Арк.

69

```

setActiveDraftType(state, payload) {
state.active_draft_type = payload;
},
saveDraftContent(state, payload) {
state.draft_content = payload;
}
},
actions: {
setActiveEvents({ dispatch }) {
const main = document.getElementById("main");
let active_elements_id = [];
let children = main.childNodes;
console.log("children", children);
children.forEach(child => {
active_elements_id.push(child.id);
document
.getElementById(child.id)
.addEventListener("click", function(e) {
e.stopPropagation();
dispatch("setActiveElement", e.target);
});
child.childNodes.forEach(el => {
active_elements_id.push(el.id);
document.getElementById(el.id).addEventListener("click", function(e) {
e.stopPropagation();
dispatch("setActiveElement", e.target);
});
});
});
},
addNewBlock({ state, dispatch }) {
const main = document.getElementById("main");
const new_el = document.createElement("div");
const new_id = Math.random()
.toString(36)
.substr(2, 9);
const default_styles =
"width: 100%; height: 512px; border-bottom: 1px solid";
new_el.setAttribute("id", new_id);
new_el.setAttribute("style", default_styles);
new_el.addEventListener("click", function(e) {
e.stopPropagation();
dispatch("setActiveElement", e.target);
});
});

```



```

main.appendChild(new_el);
//state.elements_draft.push(new_el);
},
addNewHeading({ state, dispatch }) {
const active_block = document.getElementById(state.active_draft_id);
const new_el = document.createElement("h1");
const new_id = Math.random()
.toString(36)
.substr(2, 9);
new_el.setAttribute("id", new_id);
new_el.innerHTML = "Heading";
new_el.addEventListener("click", function(e) {
e.stopPropagation();
return dispatch("setActiveElement", e.target);
});
active_block.appendChild(new_el);
},
setActiveElement({ commit }, target) {
let tag = target.tagName.toLocaleLowerCase();
let target_block_type = null;
if (tag === "div") target_block_type = "block";
if (tag === "h1") target_block_type = "text";
commit("setActiveDraftId", target.id);
commit("setActiveDraftType", target_block_type);
},
setActiveElementHeight({ state }, height) {
const active_el = document.getElementById(state.active_draft_id);
active_el.style.height = height + "px";
},
setActiveHeading({ state }, heading) {
const active_el = document.getElementById(state.active_draft_id);
active_el.innerHTML = heading;
},
setActiveElementFontSize({ state }, fontSize) {
const active_el = document.getElementById(state.active_draft_id);
active_el.style.fontSize = fontSize + "px";
},
setActiveElementColor({ state }, color) {
const active_el = document.getElementById(state.active_draft_id);
active_el.style.color = color;
}
}
}
}

```

mixins/focusMixin.js

```
export default {
  created() {
    window.addEventListener('keyup', this.keyEvents);
  },
  beforeDestroy() {
    window.removeEventListener('keyup', this.keyEvents);
  },
  mounted() {
    setTimeout(() => {
      this.$refs[this.focusedInput].focus();
    }, 100);
  },
  data() {
    return {
      focusedInput: 1,
    };
  },
  methods: {
    keyEvents(e) {
      e.stopPropagation();
      let refs = Object.keys(this.$refs);
      const refsLength = refs.length;
      if (e.key === 'ArrowDown' || (!e.shiftKey && e.key === 'Tab')) {
        e.preventDefault();
        if (this.focusedInput < refsLength) {
          this.focusedInput++;
        } else {
          this.focusedInput = 1;
        }
      }
      if (e.key === 'ArrowUp' || (e.shiftKey && e.key === 'Tab')) {
        e.preventDefault();
        if (this.focusedInput > 1) {
          this.focusedInput--;
        } else {
          this.focusedInput = refsLength;
        }
      }

      this.$refs[this.focusedInput].focus();
    },
  },
};
```

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛЦ.467800.003 ПЗ

Арк.

72

mixins/staticMixin.js

```
export default {
  methods: {
    setStrCapitalized(str) {
      if (!str) return;
      return str.charAt(0).toUpperCase() + str.slice(1);
    },
    replaceNotDigits(str) {
      return str.replace(/\D/g, "");
    },
    isNumber(e) {
      let charCode = e.keyCode;

      /*if (isNaN(e.target.value))
      e.target.value = e.target.value.slice(0, -1);*/

      if (
        e.shiftKey ||
        (charCode > 31 &&
          charCode !== 32 &&
          (charCode !== 44 || isDecimal) &&
          (charCode !== 45 || !allowMinus) &&
          (charCode !== 46 || isDecimal) &&
          (charCode < 48 || charCode > 57) &&
          (charCode < 96 || charCode > 105) &&
          charCode !== 108 &&
          charCode !== 110 &&
          charCode !== 190 &&
          charCode !== 191)
      ) {
        e.preventDefault();
        return false;
      } else {
        return true;
      }
    },
  },
}
```

					ІАЛІЦ.467800.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підп.	Дата		73