Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

_____факультет інформаційних технологій_____
(факультет)

Кафедра_____інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
Кваліфікаційної роботи ступеня
_____бакалавра_____
(назва освітньо-кваліфікаційного рівня)

Студента___ Рудь Володимира Володимировича_____
(ПІБ)
академічної групи_________
(шифр)
спеціальності __126 «Інформаційні системи та технології»_____
(код і назва спеціальності)
за освітньо-професійною програмою_____
_____«Інформаційні системи та технології»_____
(офіційна назва)
на тему____«Розробка аналітичного веб-сайту аналізу виїзду мешканців України за кордон»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
Кваліфікаційної роботи	Гнатушенко В.В.			
розділів:				

Рецензент			

Нормоконтроль	Гнатушенко В.В.		

Дніпро
2021

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
___інформаційних систем та технологій_____
(повна назва)

_____
(підпис) (прізвище, ініціали)

«_____»_____20__ року

ЗАВДАННЯ

на дипломний проект (роботу)

_____бакалавра_____
(бакалавра, спеціаліста, магістра)

студенту __Рудь В.В._____ **академічної групи**_____
(прізвище та ініціали) (шифр)
спеціальності___126 «Інформаційні системи та технології»_____
за освітньою-професійною програмою_____
_____«Інформаційні системи та технології»_____
на тему____«Розробка аналітичного веб-сайту аналізу виїзду мешканців України за кордон»_____

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз стану області рішення задачі.	25.04.2021
Розділ 2	Проектні рішення по завданню.	16.06.2021

Завдання видано _____ ___В.В. Гнатушенко_____
(підпис) (прізвище,ініціали)

Дата видачі завдання: ___

Дата подання до екзаменаційної комісії _____

Завдання прийняв до виконання _____ __Рудь В.В._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85 с., 27 рис., 2 додатки, 29 джерел.

Об’єкт розробки: аналітичний веб-сайт аналізу виїзду мешканців України за кордон.

Мета роботи: створення аналітичного веб-сайту – у вигляді SPA на мові програмування javascript(з використанням фреймворку vue.js).

У вступі наведена інформація про стан проблеми, вказана її причина, наведена тема дипломної роботи та її мета.

В першому розділі наведені основні відомості про рівень розвитку інформаційних систем та технологій у сфері веб-розробки – описані основні актуальні технології для створення веб-сайтів, веб-додатків вказане їх призначення та способи їх використання.

У другому роділі описані проектні рішення по завданню розробки веб-сайту у вигляді SPA з використанням frontend фреймворку VueJs. Були описані етапи розробки сайту та проаналізовані екосистема фреймворку VueJs, JSON-формат, який є основним форматом для сховища та отримання данних за допомогою AJAX-запитів за допомогою javascript-бібліотеки axios, яка є аналогом Fetch Api. А також наведене обгрунтування використання текстового редактору VsCode, розроблено дизайн сайту у вигляді окремих компонентів, а також було завантажено проект на хостинг.

Практичне значення кваліфікаційної роботи полягає у встановленні кількості українців, які виїжджають за кордон, глибше і детальніше ознайомитись з динамікою зміни кількості населення України через міграцію мешканців

Розроблене технологічне рішення може бути використане в будь-яких державних установах для аналізу причин та кількості міграції українців та доступне для пересічних громадян для ознайомлення динаміки міграції населення України.

ВЕБ-САЙТ, HTML, CSS, ПРЕПРОЦЕСОРИ, JAVASCRIPT, AJAX, DOM API, VUEJS, SPA, JSON, AXIOS, РОЗРОБКА, АНАЛІЗ, КОМПОНЕНТИ, СТРУКТУРА, ЗАПИТИ.

ABSTRACT

Explanatory note: 85 pages, 27 figures, 2 appendices, 29 sources.

Object of development: an analytical website for the analysis of the departure of Ukrainians abroad.

Purpose: creation of an analytical website - in the form of a SPA in the javascript programming language (using the vue.js framework).

The introduction provides information about the state of the problem, an analysis of known analogues, substantiates its relevance and the purpose of this thesis.

The first section provides basic information about the level of development of information systems and technologies in the field of web development - describes the main current technologies for creating websites, web applications, their purpose and ways to use them.

The second section describes the design solutions for the task of developing a website in the form of SPA using the frontend framework VueJs.

The stages of site development were described and the ecosystem of the VueJs framework was analyzed, the JSON format, which is the main format for storing and retrieving data using AJAX requests using the javascript library axios, which is analogous to Fetch Api. And also the substantiation of use of the text editor VsCode is resulted, the design of a site in the form of separate components is developed, and also the project on hosting was loaded.

The practical significance of the qualification work is to establish the number of Ukrainians who go abroad, to get deeper and more detailed information about the dynamics of change in the population of Ukraine due to migration of residents

The developed technological solution can be used in any government agencies to analyze the causes and number of migration of Ukrainians and is available to ordinary citizens to learn about the dynamics of migration of the population of Ukraine.

WEBSITE, HTML, CSS, PREPROCESSORS, JAVASCRIPT, AJAX, DOM API, VUEJS, SPA, JSON, AXIOS, DEVELOPMENT, ANALYSIS, COMPONENTS, STRUCTURE, REQUESTS.

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1 12
1.1 Основні технології для створення Web-сайту 12
1.1.1 HTML 12
1.1.2 CSS 14
1.1.2.1 Препроцесори 18
1.1.3 Javascript 21
1.1.3.1 Перевагий недоліки 21
1.1.3.2 Chart.Js 23
1.1.4 DOM API 25
1.1.4.1 Стандарти Dom 26
1.2 Поняття SPA-додаток 27
1.2.1 Тлумачення 27
1.2.2 Історія 29
1.2.3 AJAX 29
1.3 Vue.Js 31
1.3.1 Огляд 31
1.3.1.1 Концепції Vue.js 32
1.3.1.2 Екосистема фреймворку 38
1.3.1.3 Основні поняття 44
1.3.2 Порівняння з відомими аналогами 47
1.3.2.1 React 47
1.3.2.2. Angular 54
РОЗДІЛ 2 56
2.1. Вибір і обгрунтування рішення поставленої задачі 56
2.2. Вибір і обгрунтування використання програмних засобів 56
2.3. Розробка сайту 59
2.3.1 Створення структури сайту 59

2.3.2. Створення дизайну сайта 60
2.3.3. Поступове впровадження Vue.JS 61
2.3.4. Організаці зберігання даних та їх отримання 63
2.3.5. Фіналізація проекту 64
ВИСНОВКИ 66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 67
ДОДАТОК А 73
ВИХІДНИЙ КОД САЙТА 73
ДОДАТОК Б 70
ДИЗАЙН САЙТА 70

ВСТУП

Проблема спаду населення охоплює Україну довгі роки. На момент розпаду СРСР в 1991 році у республіці проживало 51,6 млн осіб. На грудень 2001 року, згідно з даними перепису, в Україні проживали 48 мільйонів 457 тисяч осіб і з кожним роком ця цифра стає все менше.

З січня по листопад мінулолго 2020 року чисельність населення країни скоротилася на 272 тис. чоловік, а смертність в два рази перевищила народжуваність. Про це свідчать дані Держстату України. Для порівняння за цей же період 2019 року зменшення становило 230 тис. чоловік. Зниження чисельності населення фіксується в 23 з 24 областях. Виняток - Київська область, куди на заробітки з'їжджаються мешканці інших регіонів.

Станом на 1 січня 2021 року згідно з даними Державної служби статистики України, оціночна чисельність наявного населення України на 1 січня 2021 року (без урахування Криму і Севастополя) склала 41 588 354 осіб, що на 314,1 тис. Осіб менше аналогічного показника попереднього року .

Експерт зазначають, що реальний спад набагато вищє офіційніх цифр. Так міністр кабінету міністрів України Дмитро Дубілет заявив, що В кінці 2019 року в Україні мешкали 37 мільйонів 290 тисяч людей (20,01 млн жінок і 17,28 млн чоловіків). Такий результат електронного тестового перепису населення. Тобто за 19 років населення країни (без урахування Криму і непідконтрольних уряду районів Донбасу) скоротилося на 11 мільйонів осіб.

Перепис не охоплював мешканців анексованого Криму, а також непідконтрольних Києву регіонів Донецької і Луганської областей. При підрахунку враховувалися дані трьох мобільних операторів (Kyivstar, Lifecell, Vodafone) зв'язку України, Державної служби статистики(Держстат) та електронних реєстрів, а також державного реєстру фізичних осіб.

Підтвердити або спростувати ці відомості міг би Всеукраїнський перепис населення, але він останній і єдиний раз проводився 2001 року.

Експерти вважають, що основними причинами скорочення населення в країні в першу чергу вважаються міграція - з країни виїжджає більше людей, ніж повертається. Так, згідно з Всесвітнього конгресу Українців в світі українська діаспора налічує більше 20 мільйонів людей, більшість з них є громадянами України. А також негативне співвідношення між народжуваністю і смертністю. Ще однією причиною є "окуповані території".

За отриманими даними про населення України можна припустити, що головною причиною демографічної кризи в Україні є трудова міграція або ж міграція з подальшим ПМЖ, тому слід розглянути причини трудової міграції з України.

Ключові фактори, які оцінювали респонденти - це: рівень доходів, поточна робота, системи охорони здоров'я і освіти, соціальне життя і дозвілля, безпеку, дотримання прав, екологія і загальний досвід життя в Україні.

Отже, згідно з опитуваннями причини виїзду Українців за кордон наступні:

- Низька зарабітна плата. Більшість українців(51%) вважають свій рівень доходів недостатнім для нормального життя.
- Деградація медецини. Так, 58% опитаних не задоволені станом системи охорони здоров'я.
- Освіта. Задоволених системою освіти виявилося майже 52%, при цьому 48% відзначили, що повністю або частково не задоволені нею.
- Стан безпеки. 72% респондентів не задоволені станом безпеки в країні і відчувають себе невпевнено щодо власного майбутнього.
- Правове поле. 62% негативно оцінюють правове поле і регулювання в країні, зокрема в контексті відсутності дискримінації та рівня демократичних свобод.

Через демографічну кризу не відома точна кількість населення, тому важче приймати рішення державного значення щодо регулювання цього питання.

Незважаючи на це, все ж можна встановити кількість українців, які виїжджають за кордон, зокрема за допомогою аналітичного веб-сайту, до

доступу якого, потрібен лише браузер, будь-який електроний пристрій (смартфон, планшет, комп'ютер та ін.) та підключення до мережі Інтернет.

Таким чином, мета дипломного проекту – створення аналітичного веб-сайту аналізу виїзду мешканців України за кордон.

Сайт розроблений у вигляді SPA(Single Paged Application) з використанням мови сценаріїв Javascript, яка орієнтована на Web-розробку для надання інтерактивності веб-сторінкам та популярного фреймворку VueJs.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1 Основні технології для створення Web-сайту

1.1.1 HTML

Мова розмітки гіпертексту або HTML (Hyper Text Markup Language) – самий базовий будівельний блок в Web. Під гіпертекстом("hypertext") розуміються посилання, які з'єднують веб-сторінки один з одним або в межах одного веб-сайту, або між веб-сайтами. Посилання є фундаментальним аспектом Вебу. Завантажуючи контент в Інтернет і пов'язуючи його зі сторінками, створеними іншими людьми, ви стаєте активним учасником Всесвітньої павутини.

Текстові документи, що містять розмітку на мові HTML (такі документи зазвичай мають розширення .html або .htm), обробляються спеціальними додатками, які відображають документ в його форматованому вигляді. Такі додатки, називають «браузерами» або «інтернет-оглядачами», зазвичай надають користувачеві зручний інтерфейс для запиту веб-сторінок, їх перегляду (і виведення на інші зовнішні пристрої) і, при необхідності, відправки введених користувачем даних на сервер. Найбільш популярними на сьогоднішній день браузерами є Google Chrome, Mozilla Firefox, Opera, Edje і Safari.

HTML - це стандартна мова розмітки для документів, призначених для відображення в веб-браузері і визначає зміст і структуру веб-контенту. Цьому можуть допомогти такі технології, як каскадні таблиці стилів (CSS) і мови сценаріїв, такі як JavaScript. [1]

Веб-браузери [2] [3] отримують HTML-документи з веб-сервера або з локального сховища і перетворюють їх в мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично і спочатку включає підказки для зовнішнього вигляду документа.

Елементи HTML - це будівельні блоки HTML-сторінок. За допомогою конструкцій HTML зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані в сторінку. HTML надає засоби для створення структурованих документів(рис. 1.1) шляхом позначення структурної семантики.

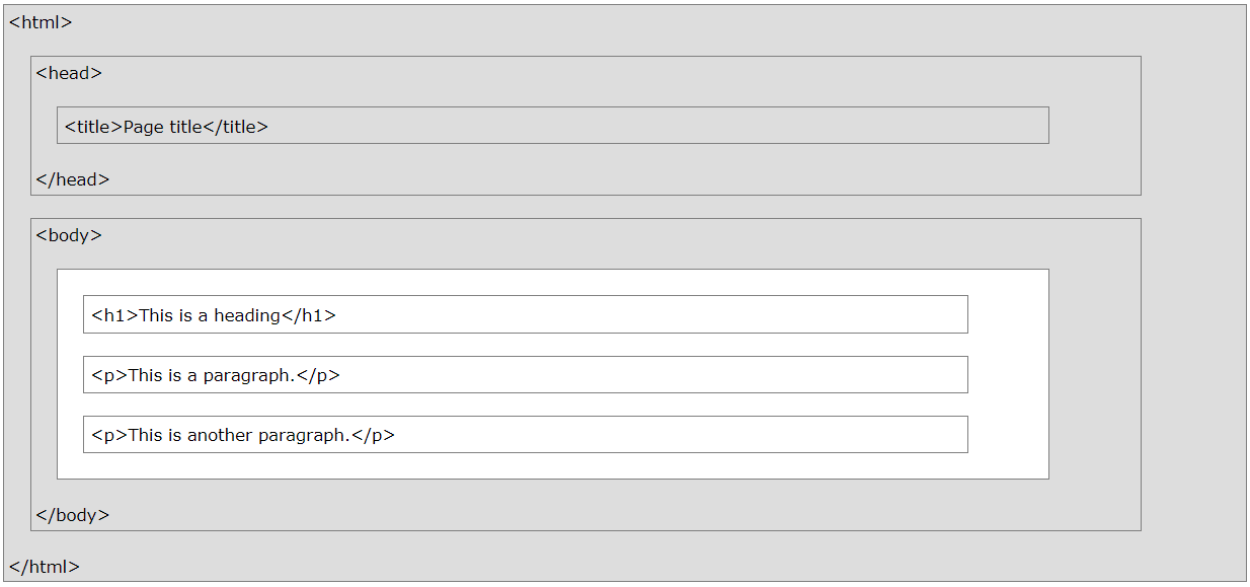

Рис. 1.1. Структура HTML-сторінки

Елементи HTML виділяються тегами, записаними з використанням кутових дужок. Такі теги, як <img /> і <input />, безпосередньо вводять контент на сторінку. Інші теги, такі як <p>, оточують і надають інформацію про текст документа і можуть включати інші теги в якості піделементів. Браузери не відображають HTML-теги, але використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати програми, написані на мові сценаріїв, такому як JavaScript, який впливає на поведінку і вміст веб-сторінок. Включення CSS визначає зовнішній вигляд і макет контенту. Консорціум World Wide Web (W3C), колишній розробник HTML і в даний час працює в стандарті CSS, з 1997 року заохочує використання CSS замість явного презентаційного HTML.[4] [5]

1.1.2 CSS

Каскадні таблиці стилів або CSS (Cascading Style Sheets) - це мова таблиць стилів, використовувана для опису представлення документа, написаного на мові розмітки, такому як HTML. CSS - це наріжна технологія всесвітньої павутини, поряд з HTML і JavaScript, вона описує те, як повинні бути відображені html-елементи

CSS розроблена для поділу оформлення та вмісту, включаючи макет, кольори і шрифти. Це поділ покращує доступність контенту, забезпечує більшу гнучкість і контроль в специфікації характеристик уявлення, дозволяє декільком веб-сторінкам спільно використовувати форматування, вказавши відповідний CSS в окремому файлі .css, що знижує складність і повторення структурного контенту, а також дозволяє файл .css, який необхідно кешувати, щоб підвищити швидкість завантаження сторінки між сторінками, які спільно використовують файл, і його форматування.

Поділ форматування і вмісту також уможливлює уявлення однієї і тієї ж сторінки розмітки в різних стилях для різних методів рендеринга, таких як на екрані, у друку, голосом (через мовний браузер або програму читання з екрану) і на основі шрифту Брайля. В CSS також є правила для альтернативного форматування, якщо доступ до контенту здійснюється на мобільному пристрої.

Каскадування імен відбувається з зазначеної схеми пріоритету, щоб визначити, яке правило стилю застосовується, якщо конкретного елементу відповідає кілька правил.[6]

Правило CSS(рис. 1.2) складається з селектора і блоку оголошення. Селектор вказує на HTML-елемент, який потрібно стилізувати.

Блок оголошень містить одне або кілька оголошень, між якими ставиться крапка з комою. Кожне оголошення включає ім'я властивості CSS і значення, розділені двокрапкою. Кілька оголошень CSS розділяються крапкою з комою, а блоки оголошень полягають у фігурні дужки.

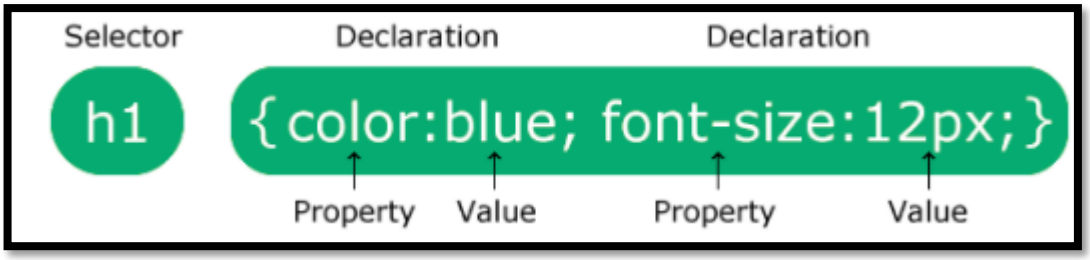

Рис. 1.2. CSS синтаксис

Правила CSS можуть розташовуватися як в самому веб-документі, зовнішній вигляд якого вони описують, так і в зовнішніх файлах, що мають розширення .css. Формат CSS - це текстовий файл, в якому міститься перелік правил CSS і коментарів до них.

Стилі CSS можуть впроваджені в HTML-документ чотирма способами:

- External(Зовнішній) CSS – Зовнішні стилі визначаються в элементі <link> всередині розділу <head> HTML-сторінки:

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    .....
  </body>
</html>
```

Таблиця стилів повинна бути збережена з розширенням .css. Зовнішній файл .css не повинен містити жодних тегів HTML. Ось як виглядає файл "style.css":

```
body {
 background-color: lightblue;
```

```
}

h1 {
  margin-left: 20px;
}
```

- Internal(внутрішній) CSS – Внутрішня таблиця стилів може вживатися, якщо одна-єдина сторінка HTML має унікальний стиль. Внутрішні стилі визначаються в елементі <style> всередині розділу <head> HTML-сторінки:

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <style>
      body {
        color: red;
      }
    </style>
  </head>
  <body>
    .....
  </body>
</html>
```

- Inline(рядковий) CSS - Рядковий стиль може використовуватися для застосування унікального стилю для окремого елемента. Рядкові стилі в атрибуті "style" відповідного елемента:

```
<!DOCTYPE>
<html>
  <head>
    .....
```

```
  </head>
  <body>
    <p style="font-size: 20px; color: green;">
      .....
    </p>
  </body>
</html>
```

- Інструкцією @import – Може використовуватися коли файл розміщується окремо від батьківського документа, його підключають за допомогою інструкції @import в елементі <style>:

```
    <!DOCTYPE html>
<html>
  <head>
    .....
    <style media="all">
      @import url(style.css);
    </style>
  </head>
</html>
```

Даний спосіб не рекомендується використовувати, оскільки може збільшити час завантаження сторінки. Коли сторінка завантажується за допомогою команди @import, вона повинна прочитати і імпортувати таблицю стилів, зазначену в правилі, а потім застосувати її до поточної сторінці. Це не займе багато часу, і навіть на найбільших веб-сайтах, що використовують команду @import, помітної різниці в часі завантаження не відчувається. Однак, для ботів, які використовують час завантаження для обчислення вашого рейтингу в пошуку, це може зрушити вас вниз у видачі пошукових систем.

1.1.2.1 Препроцесори

CSS препроцесор [7] - це надбудова над CSS, яка додає раніше недоступні можливості для CSS, за допомогою нових синтаксичних конструкцій. Основне завдання препроцесора - це надання зручних синтаксичних конструкцій для розробника, щоб спростити, і тим самим, прискорити розробку та підтримку стилів в проектах.

CSS препроцесори перетворять код, написаний з використанням препроцесорної мови, в чистий і валідний CSS-код.

За допомогою препроцесорів ви можете писати код, який націлений на:

- Читабельність для людини
- Структурованість і логічність
- Продуктивність

На сьогоднішній день можна виділити три популярних препроцесора:

- Less
- Sass (SCSS)
- Stylus

Less – Найпопулярніший на сьогоднішній день препроцесор. Заснований в 2009 році Алексіс Сельер (Alexis Sellier) і написаний на JavaScript Має всі базові можливості препроцесорів і навіть більше, але не має умовних конструкцій і циклів в звичному для нас розумінні. Основним плюсом є його простота, практично стандартний для CSS синтаксис і можливість розширення функціоналу за рахунок системи плагінів. Приклад синтаксису:

```
block {
  margin: 10px 5px;
  padding: 2px;
}
p {
  .block;
```

```
  border: 1px solid #EEE;
}
ul, ol {
  .block;
  color: #333;
  text-transform: uppercase;
}
```

Sass(SCSS) – Найпотужніший з CSS-препроцесорів. Має досить велику спільноту розробників. Заснований в 2007 році як модуль для HAML і написаний на Ruby (є порт на C ++). Має куди більший асортимент можливостей в порівнянні з Less. Можливості самого препроцесора розширюються за рахунок багатофункціональної бібліотеки Compass, яка дозволяє вийти за рамки CSS і працювати, наприклад, з спрайтами в автоматичному режимі.

Має два синтаксиса:

- Sass (Syntactically Awesome Style Sheets) - спрощений синтаксис CSS, який заснований на ідентації. Вважається застарілим. Приклад синтакксису:

```
$ Color: gray = my-font ($ color)

  font-family: Arial, Helvetica, sans-serif
  font-size: 16px
  color: $ color

body
  background: $ color
  margin: 0
  + My-font (white)
```

- SCSS (Sassy CSS) - заснований на стандартному для CSS синтаксисі.Приклад синтаксису:

```
$ Color: gray;
@mixin my-font ($ color) {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 16px;
  color: $ color;
}
body {
  background: $ color;
  margin: 0;
  @include my-font (white);
}
```

Stylus – Наймолодший, але в той же час найперспективніший CSS-препроцесор. Заснований в 2010 році відомої TJ Holowaychuk. Кажуть, це найзручніший і розширюваний препроцесор, а ще він гнучкіше Sass. Написаний на JavaScript. Підтримує безліч варіантів синтаксису від подібного CSS до спрощеного (відсутні:,;, {} і деякі дужки). Приклад синтаксису:

```
/* style.styl */
h1 {
  color: #0982C1;
}

/* omit brackets */
h1
  color: #0982C1;

/* omit colons and semi-colons */
h1
  color #0982C1
```

1.1.3 Javascript

JavaScript (JS) - це легка, що інтерпретується або оперативно скомпільована мова програмування з функціями першого класу. Хоча вона найбільш відома як мова сценаріїв для веб-сторінок, вона також використовується в багатьох середовищах, не пов'язаних з браузером, наприклад, Node.js, Apache CouchDB і Adobe Acrobat.

JavaScript це заснована на прототипах, мультіпарадігмена, однопотокова, з динамічною типізацією мова, що підтримує об'єктно-орієнтоване, імперативне і декларативне (наприклад, функціональне програмування) стилі.

Стандарти для JavaScript - це специфікація мови ECMAScript (ECMA-262), яка відповідає міжнародній редакції ISO / IEC 16262.

Поряд з HTML і CSS, JavaScript є однією з основних технологій Всесвітньої павутини. Понад 97% веб-сайтів використовують її на стороні клієнта для визначення поведінки веб-сторінок, часто з використанням сторонніх бібліотек або фреймворків. Всі основні веб-браузери мають спеціальний механізм JavaScript для виконання коду на пристрої користувача.

Як мову з безліччю парадигм, JavaScript підтримує керовані подіями, функціональні і імперативні стилі програмування. Вона має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних і об'єктною моделлю документа (DOM).

Основні архітектурні риси JS: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

Не плутайте JavaScript з мовою програмування Java. І «Java», і «JavaScript» є товарними знаками або зареєстрованими товарними знаками Oracle в США та інших країнах. Однак ці дві мови програмування мають дуже різні синтаксис, семантику і використання. [8] [9]

1.1.3.1 Перевагий недоліки

Переваги:

- Незамінність для веб-розробки. Підтримка скриптів усіма популярними браузерами; повна інтеграція з верстkою сторінок (HTML + CSS) і серверної частиною (backend).
- Швидкість роботи і продуктивність. Javascript дозволяє частково обробляти веб-сторінки на комп'ютерах користувача без запитів до сервера. Це економить час і трафік, знижує навантаження на сервер.
- Потужна інфраструктура (екосистема). Перші 10 років цього не було. Потім, з його, кількість готових рішень в відкритому доступі зросла, і його стало можливо застосовувати у багатьох інших галузях, окрім веб-розробки.
- Простота і раціональність застосування. Просту завдання можна вирішити за 5 хвилин, не треба робити зайву роботу. Для складних завдань є варіанти вирішення, можна підібрати кращий, адаптувати.
- Зручність для користувача інтерфейсів. Заповнення форм, вибір дій, активація кнопок, перевірки введення, реагування на наведення / кліки миші і т.п Це дає приголомшливий рівень юзабіліті.
- Легкість освоєння.

Недоліки:

- Немає можливості читання та завантаження файлів. Це обмеження функціональності на стороні клієнта. Головна причина - міркування безпеки.
- Нестрога типізація і вільне трактування. Мова ігнорує явні нестиковки. Має місце різна інтерпретація даних. Немає можливості раннього виявлення помилок. Всі недоліки виявляються вже на етапі роботи.
- Нема підтримки віддаленого доступу. Тому мова не можна використовувати для мережевих додатків. За це Javascript навіть не вважають повноцінною мовою програмування. [10]

1.1.3.2 Chart.Js

Chart.js - це популярний інструмент, Який призначений для створення графіків і діаграм на основі HTML5 Canvas.

Дана бібліотека дозволяє без особливих зусиль створювати графіки та діаграми будь-якого типу, а також вибудовувати дані на діапазоні часу і логарифмічною шкалою. Також в неї вбудовані засоби роботи з анімацією, що дозволить ефектно змінювати графіки в залежності від нових даних, а також експериментувати з кольором.

Уявимо таблицю з чисельністю населення країн світу у вигляді стовпчастий діаграми. На осі у цифри, а на осі x назва країн. Для початку створимо елемент canvas з ідентифікатором popChart.

```
<canvas id="popChart" width="600" height="400"></canvas>
```

Атрибути width і height визначають розміри діаграми. Для того щоб графік був адаптивним ми повинні визначити ширину і висоту елемента canvas. Далі нам необхідно ініціалізувати клас Chart. Зробити це можна, вибравши елемент сторінки, ініціювавши бібліотеку через jQuery або 2d контекст елемента canvas.

```
var popCanvas = $ ( "# popChart");
var popCanvas = document.getElementById ( "popChart");
var popCanvas = document.getElementById ( "popChart"). getContext ( "2d");
```

Все що залишилося зробити - передати параметри в конструктор:

```
var barChart = new Chart(popCanvas, {
  type: 'bar',
  data: {
    labels: ["China", "India", "United States", "Indonesia", "Brazil", "Pakistan", "Nigeria", "Bangladesh", "Russia", "Japan"],
```

```
    datasets: [{
      label: 'Population',
      data: [1379302771, 1281935911, 326625791, 260580739, 207353391, 204924861, 190632261, 157826578, 142257519, 126451398],
      backgroundColor: [
        'rgba(255, 99, 132, 0.6)',
        'rgba(54, 162, 235, 0.6)',
        'rgba(255, 206, 86, 0.6)',
        'rgba(75, 192, 192, 0.6)',
        'rgba(153, 102, 255, 0.6)',
        'rgba(255, 159, 64, 0.6)',
        'rgba(255, 99, 132, 0.6)',
        'rgba(54, 162, 235, 0.6)',
        'rgba(255, 206, 86, 0.6)',
        'rgba(75, 192, 192, 0.6)',
        'rgba(153, 102, 255, 0.6)'
      ]
    }]
  }
});
```

Chart.js витягне інформацію з переданого об'єкта і намалює відповідний графік. У параметрі type необхідно вказати тип діаграми. Дане поле може приймати одне з наступних значень: line, bar, radar, polarArea, pie, doughnut або bubble.

Для кожного типу діаграм характерні особливі характеристики. В даному випадку вийде наступна діаграма(рис. 1.3):

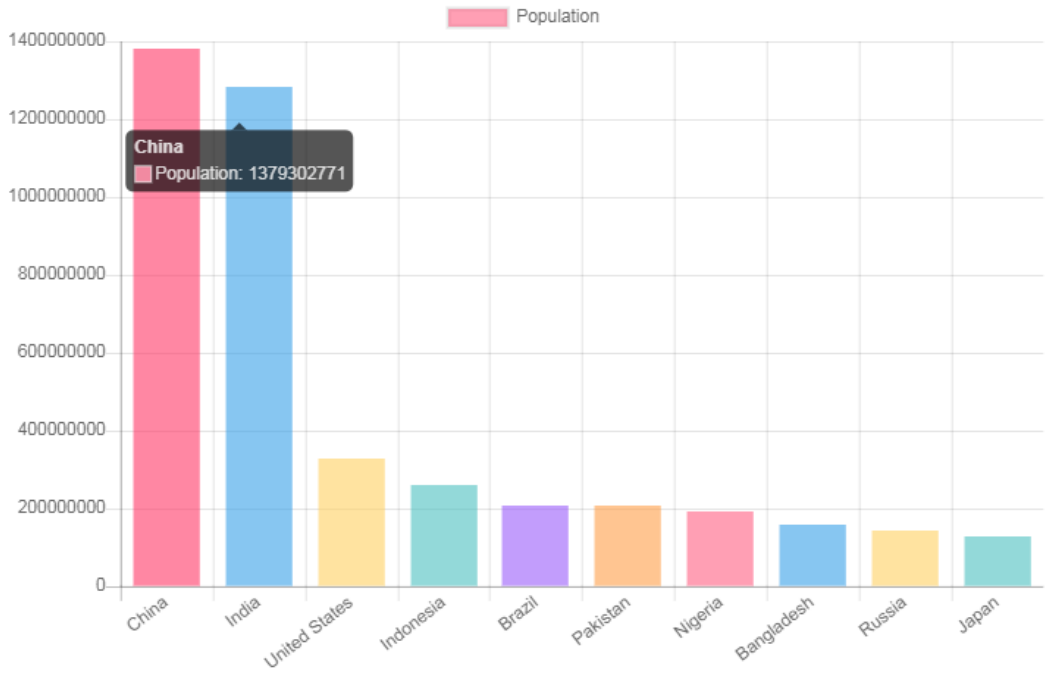

Рис. 1.3. Chart.js діаграма

Для того щоб побачити конкретні цифри досить підвести курсор миші до того чи іншого колонки. Можна помітити що розміри графіка не відповідаю виставленим раніше. Зате діаграма повністю адаптивна під різні пристрої.

Якщо необхідно щоб діаграма була одного і того розміру на всіх пристроях, то поставте значення false в поле responsive. [11]

1.1.4 DOM API

Об'єктна модель документа (DOM) - це кросплатформений і не залежний від мови інтерфейс (API), який представляє і взаємодіє з усіма HTML або XML документами. . DOM це модель документа завантажена в браузер і представляє документ як деревоподібну структуру(рис. 1.4), в якій кожна гілка дерева закінчується вузлом і кожен вузол є об'єктом , який представляє частину документа (наприклад, елемент документа, рядок тексту або коментар). Методи DOM забезпечують програмний доступ до дерева; з їх допомогою можна змінити структуру, стиль або зміст документа.

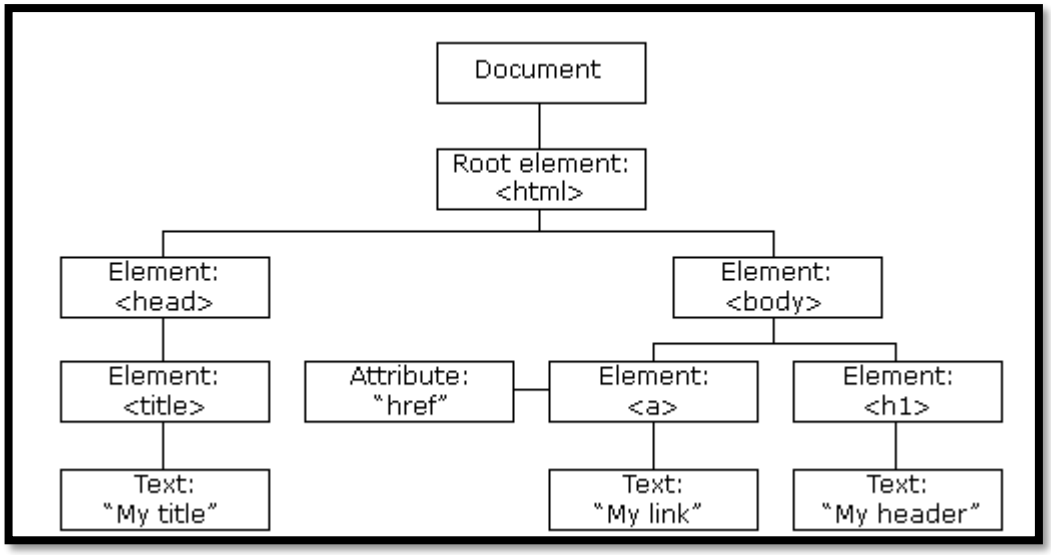

Рис. 1.4. Структура DOM

DOM це самий використовуваний API в Web тому, що він дає коду запущеному в браузері доступ і взаємодію з кожним вузлом в документі. Вузли можуть бути створені, переміщені та змінені. Обробники подій можуть бути додані до вузлів і спрацьовують при настанні цієї події.

DOM не був визначений спочатку - він прийшов коли браузери почали реалізовувати підтримку JavaScript. Цей успадкований DOM іноді називають DOM 0. Принциповою стандартизацією моделі DOM займався Консорціум World Wide Web (W3C), який в останній раз розробив рекомендацію в 2004 році. WHATWG взяла на себе розробку стандарту, опублікувавши його як живий документ. Починаючи з публікації DOM Level 4 в 2015 році, W3C тепер публікує нові рекомендації на основі знімків стандарту WHATWG. [9]

1.1.4.1 Стандарти Dom

- DOM Level 1 надає повну модель для всього документа HTML або XML, включаючи засоби для зміни будь-якої частині документа.
- DOM Level 2 був опублікований в кінці 2000 року. Він представив функцію getElementById, а також модель подій і підтримку просторів імен XML і CSS.

- В DOM Level 3, опублікованому в квітні 2004 р, додана підтримка XPath і обробки подій клавіатури, а також інтерфейс для серіалізації документів в форматі XML.
- DOM Level 4 був опублікований в 2015 році. Це знімок живого стандарту WHATWG. [13]

1.2 Поняття SPA-додаток

1.2.1 Тлумачення

Односторінковий додаток SPA (Single Page Application) – це веб-додаток або веб-сайт, який взаємодіє з користувачем, динамічно перезаписуючи поточну веб-сторінку новими даними з веб-сервера, замість методу за замовчуванням, коли веб-браузер завантажує цілі нові сторінки. Мета - більш швидкі переходи, які зроблять веб-сайт більш схожим на нативний додаток.

У SPA весь необхідний код HTML, JavaScript і CSS або витягується браузером при завантаженні однієї сторінки, або відповідні ресурси динамічно завантажуються і додаються на сторінку в міру необхідності, зазвичай у відповідь на дії користувача. Сторінка не потребує перезавантаження ні на якому етапі процесу за рахунок динамічного оновлення за домомогою AJAX і не передає управління іншій сторінці, хоча хеш розташування або API історії HTML5 можна використовувати для забезпечення сприйняття і навігації по окремим логічним сторінок в додатку.

Популярні приклади SPA – Gmail(рис. 1.5), Google Maps(рис. 1.6), Facebook(рис. 1.7).

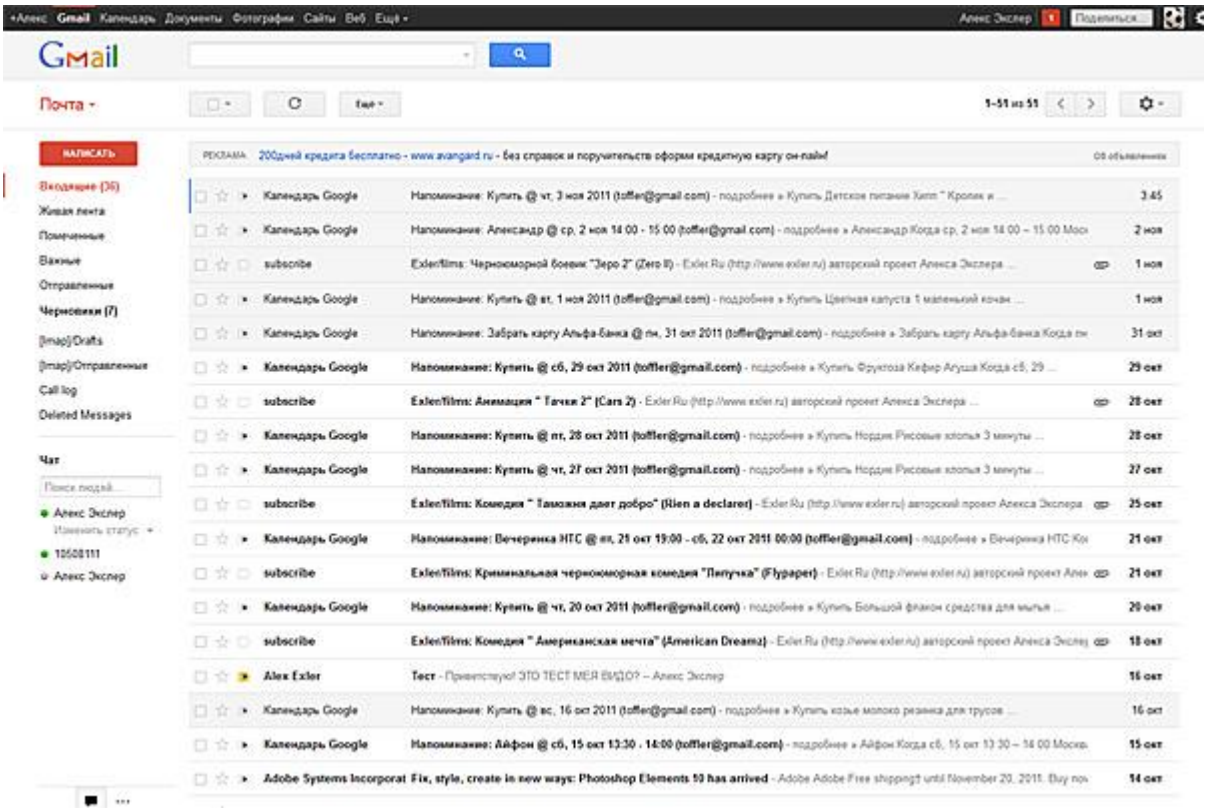

Рис. 1.5. Інтерфейс Gmail

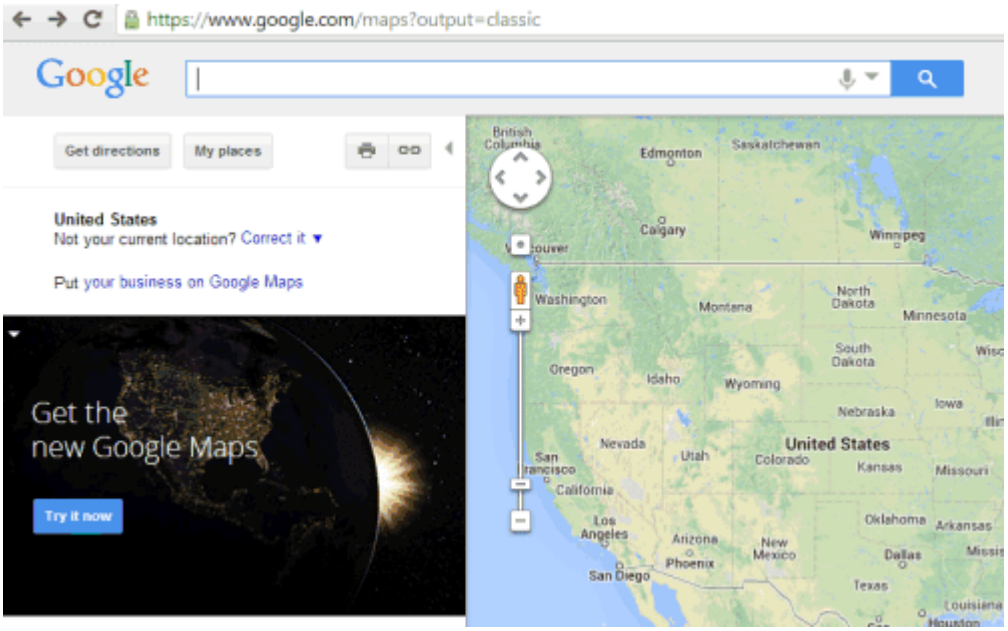

Рис. 1.6. Інтерфейс GoogleMaps

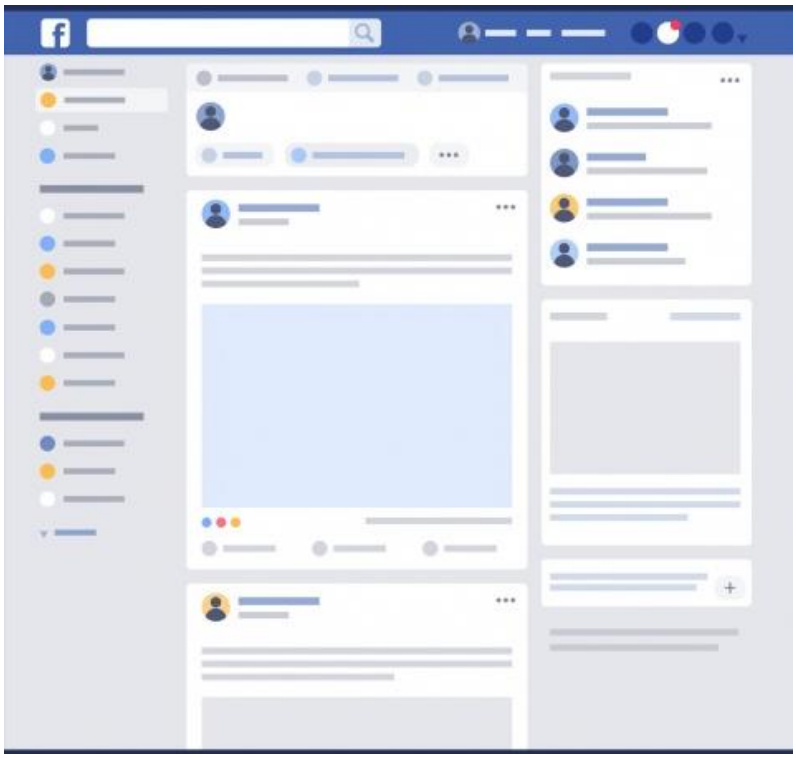

Рис. 1.7. Інтерфейс Facebook

1.2.2 Історія

Походження терміну «односторінковий додаток» незрозуміле, хоча концепція обговорювалася, принайні, ще в 2003 році. Стюарт Морріс, студент-програміст в Кардіфського університету, Уельс, написав автономний веб-сайт на slashdotslash.com з тими ж цілями і функціями в квітні 2002 року, а пізніше в тому ж році Лукас Бірд, Кевін Хакман, Майкл Пічі і Кліффорд Йе описали виконання односторінкового додатку в патенті США 8 136 109.

JavaScript можна використовувати в веб-браузері для відображення призначеного для користувача інтерфейсу та запустити програму і зв'язку з веб-сервером. Доступні зрілі бібліотеки із зовнішнім вихідним кодом, який включає в себе створення SPA, який включає обсяг коду JavaScript, який доводиться писати розробникам. [14][15]

1.2.3 AJAX

Асинхронний Javascript і XML або AJAX (Asynchronous JavaScript And XML.) – методологія веб-розробки, що використовує веб-технології для

створення асинхронних веб-додатків. За допомогою даного підходу веб-додатки можуть відправляти і отримувати дані з сервера асинхронно (в фоновому режимі), не заважаючи відображенню і поведінки існуючої сторінки. Відокремивши рівень обміну даними від рівня представлення, Ajax дозволяє веб-сторінкам і, в більш широкому сенсі, веб-додаткам динамічно змінювати вміст без необхідності перезавантажувати всю сторінку.

Ім'я AJAX вводить в оману. Додатки AJAX можуть використовувати XML для передачі даних, але не менш поширене перенесення даних у вигляді звичайного тексту або тексту JSON, який використовується найчастіше.

Ajax - це не окрема технологія, а термін, який описує підхід до використання існуючих технологій разом. HTML і CSS можна використовувати в комбінації для розмітки і стилізації інформації. Потім веб-сторінку можна змінити за допомогою JavaScript і DOM API, щоб вона динамічно відображала нову інформацію і дозволяла користувачеві взаємодіяти з нею. Вбудований в браузер об'єкт XMLHttpRequest зазвичай використовується для запиту даних з веб-сервера, дозволяючи веб-сайтам завантажувати контент на екран без оновлення сторінки. [16]

Схема роботи AJAX наступна(рис 1.8):

- На веб-сторінці відбувається подія (сторінка завантажується, натискається кнопка)
- Об'єкт XMLHttpRequest створюється за допомогою JavaScript.
- Об'єкт XMLHttpRequest відправляє запит на веб-сервер.
- Сервер обробляє запит.
- Сервер відправляє відповідь на веб-сторінку.
- Відповідь читається за допомогою JavaScript.
- Відповідна дія (наприклад, оновлення сторінки) виконується за допомогою JavaScript. [17]

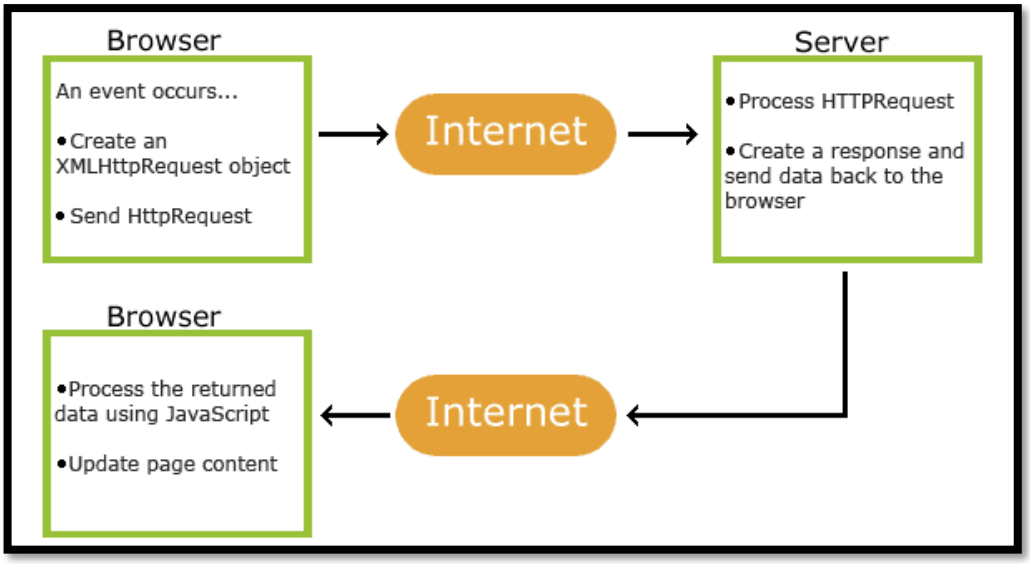

Рис. 1.8. Як працює Ajax

1.3 Vue.Js

1.3.1 Огляд

Vue - це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від інших монолітних фреймворків, Vue розроблений з нуля для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue також відмінно підтримує складні односторінкові додатки (SPA, Single-Page Applications) при використанні в поєднанні з сучасними інструментами та допоміжними бібліотеками.

Це JavaScript-фреймворк з відкритим вихідним кодом для створення інтерфейсів користувача. Легко інтегрується в проекти з використанням інших JavaScript-бібліотек. Може функціонувати як веб-фреймворк для розробки односторінкових програм у реактивному стилі.

На даний момент підтримується творцем Еваном Ю. і іншими активними членами основної команди з різних компаній, таких як Netlify, Netguru, Baidu, Livestorm,.

Vue.js простий в освоєнні, для роботи можна використовувати тільки знання JavaScript і HTML. Також можна застосувати Typescript.

У Vue.js реалізується шаблон MVVM, Vue.js пропонує можливість прив 'язки даних на Javascript, так що виведення і введення даних пов' язуються безпосередньо з джерелом даних. Таким чином, режим ручного визначення даних (наприклад, через jQuery) з HTML-DOM не потрібен. При цьому немає необхідності в ніяких додаткових анотаціях, як в Knockout.js, оголошені в Vue-Element звичайні змінні JavaScript включаються в якості реактивних елементів. [18] [19]

1.3.1.1 Концепції Vue.js

Основними концепціями Vue є [20]:

- конструктор
- компоненти
- директиви
- переходи

Конструктор

Робота з Vue.js починається зі створення нового інстансу new Vue. В el елемент, за яким стежить Vue. У template обран (або прописан інлайново) елемент, куди Vue буде рендерити. У data зберігається поточний стан інстансу, а метод computed надає вираховувані властивості.

У прикладі обчислювана властивість full_name відстежує first_name і last_name як залежності і автоматично синхронізується.

У methods можна виділити такі кастомні методи і методи життєвого циклуVue:

- beforeCreate - дивиться дані та ініціалізує події.
- created - дивиться, чи є el або template. Якщо є, то рендерить у них; якщо ні, то шукає метод render
- beforeMount - створює vm. $ el і замінює el
- mounted - елемент відреендерен

При зміні стану:

- beforeUpdate - знову рендерить VDOM і порівнює з реальним DOM-ом, застосовує зміни
- updated - Зміни відрендерені.
- beforeDestroy - Повний демонтаж вотчерів, внутрішніх компонентів і слухачів подій.
- destroyed - Викликається, коли виконання дії зупиняється.

```
new Vue({
   el: '<jQueryStyleSelector>',
   template: '<id || inline template>',
   data: {
      props: 'Це переданий параметр',
      first_name: "Іван",
      last_name: "Іванов"
   },
   computed: {
      full_name: function(){
         return this.first_name + this.last_name; //Іван Іванов
      }
   },
   methods: {
      // методи життєвого циклу
      beforeCreate: function(){},
      created: function(){},
      beforeMount: function(){},
      mounted: function(){},
      beforeUpdate: function(){},
      updated: function(){},
      beforeDestroy: function(){},
      destroyed: function(){},
```

```
    customMethodsAlso: function(){
    }
  }
})
```

Директиви

Директиви - HTML атрибути з префіксом -v, які дозволяють розширяти функціонал HTML-елементів. Vue.js надає вбудовані і користувальницькі директиви.

Розглянемо вбудовані директиви :

- V-text - Оновлює текстовий вміст елемента textContent.
- V-html - Оновлює innerHTML елемента.
- V-show - Перемикає CSS-властивість display елемента, якщо значення виразу істинне.
- v-if - Здійснює умовний рендер елемента за умови, що переданий вираз істинний.
- V-else - Визначає «блок else» для v-if або ланцюжка v-if / v-else-if.
- V-else-if - Позначає «блок else if» для v-if. Можна об'єднувати в ланцюжки.
- V-for – Циклічно рендерит элемент или блок шаблона, основываясь на переданных данных.
- V-on - Пов 'язує слухача події з елементом.
- V-bind - Динамічно пов'язує один або більше атрибутів, або вхідний параметр компонента з виразом.
- V-model - Пов'язує елемент введення даних або компонент зі змінною.
- V-slot - Вказує іменовані слоти або слоти з вхідними параметрами.
- V-pre - Пропустити компіляцію для цього елемента і всіх його дочірніх елементів.
- V-cloak - використовується для приховування нескомпільованих «прив'язок вусів» до тих пір, поки екземпляр Vue не буде готовий.

- V-once - рендерить елемент і компонент тільки один раз. [21] [22]

Компоненти

Компоненти [23] - це багаторазово використовувані екземпляри Vue зі своїм ім'ям:

```
//Визначає новий компонент, який називається button-counter
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">Лічильник кліків — {{ count }}</button>'
})
```

В наведеному вище прикладі це <button-counter>. Його можна використовувати як користувальницький тег всередині кореневого екземпляру Vue, створеного за допомогою new Vue:

```
<div id="components-demo">
  <button-counter></button-counter>
</div>

new Vue ({el: '# components-demo'})
```

У Vue.js немає особливих вимог до імен компонентів, але хороша практика - дотримуватися правил W3C щодо кастомних компонентів, тобто букви нижнього регістру і поділу через дефіс як у наведеному прикладі.

Зазвичай додаток організовується у вигляді дерева вкладених компонентів(рис. 1.9):

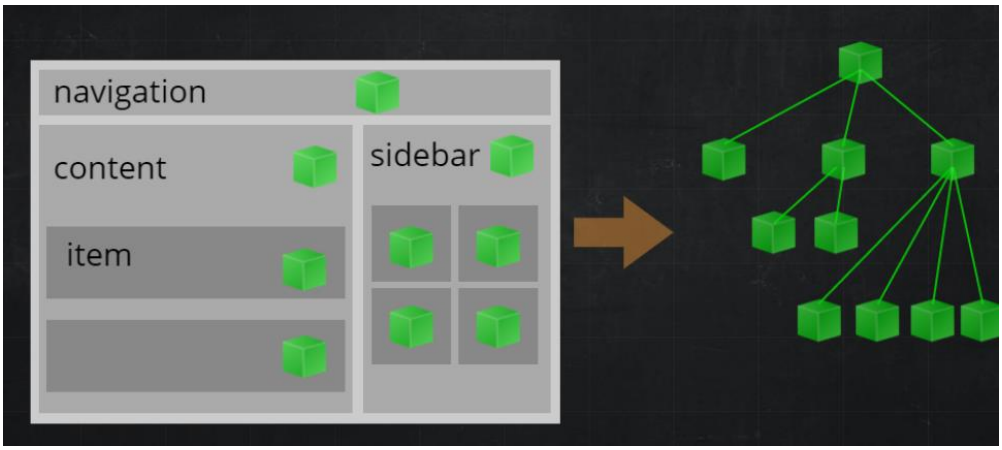

Рис 1.9. Компоненти

Комунікація між vue-компонентами здійснюється за принципом "Props in, Events out". Тобто від батьківського елемента до дочірнього інформація передається через пропси, а назад - викликаються події.

Також у Vue.js є так звані однофайлові компоненти. Ми створюємо файл з розширенням .vue і пишемо туди стилі, шаблон і логіку. Писати можна на будь-якому зручному препроцесорі (SASS, Stylus, PostCSS, Jade,...) і мові, що компілюється в JS (CoffeeScript, TypeScript):

```
<style lang="sass">
button {
   border: 1px solid gray;
   &.blue { border-color: blue; }
}
</style>
<template lang="jade">
avatar(:user='user')
input(type='text', v-model='content')
button.blue(@click='submitComment')
</template>
<script>
import Comment from '../models'
import avatar  from './components/avatar.vue'
```

```
export default {
    props: ['user'],
    components: {
        avatar
    },
    data () {
        return {
            content: ''
        }
    },
    methods: {
        submitComment (e) {
            e.preventDefault();
            var comment = new Comment(this.content)
            comment.save().then(() => {
                alert('o_O')
                this.content = ''
            })
        }
    }
}
</script>
```

Переходи

Vue надає різні способи застосування анімаційних ефектів, коли елементи відображаються, оновлені або видаляються з DOM. Вони включають в себе інструменти для [24]:

- Автоматичного застосування класів для переходів CSS та анімацій
- інтеграції сторонніх бібліотек для CSS-анімацій, таких як Animate.css
- використання JavaScript для маніпуляції DOM-ом

- інтеграції сторонніх JavaScript бібліотек для анімацій, таких як Velocity.js

Наприклад:

```
<div id="demo">
  <button @click="show = !show">Toggle show</button>
  <transition name="bounce">
    <p v-if="show">Look at me!</p>
  </transition>
</div>

new Vue({
  el: '#demo',
  data: {
    show: true
  }})
```

1.3.1.2 Екосистема фреймворку

Роутінг

У Vue.js за маршрутизацію відповідає окремий пакет vue-router. Він підтримує вкладені маршрути до вкладених компонентів, пропонує спрощене API для навігаційних хуків, керовану поведінку скролу і просунутий контроль переходів.

app.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import App from './app.vue'
import ViewA from './view-a.vue'
import ViewB from './view-b.vue'
```

```
Vue.use(VueRouter)
const router = new VueRouter()
router.map({
  '/a': { component: ViewA },
  '/b': { component: ViewB }
})
router.start(App, '#app')
```

app.vue

```
<div>
  <h1> Це шаблон, який не буде змінюватися</h1>
  <router-view><!-- вибрані компоненти змінюються --></router-view>
</div>
```

Ajax-запити

Для роботи з Ajax-запитами існує плагін vue-resource. Він надає можливості для створення веб-запитів та обробки відповідей за допомогою XMLHttpRequest або JSONP. Також особливістю плагіну є підтримка Promise API і URI шаблонів.

```
{
  // GET /someUrl
  this.$http.get('/someUrl').then((response) => {
    // успіх
  }, (response) => {
    // або помилка
  });
}
```

Керування станом

Vuex - патерн і бібліотека управління станом для додатків на Vue.js. Він надає централізований загальний стан для всіх компонентів у додатку і правила, що забезпечують передбачувану зміну стану(рис. 1.10).

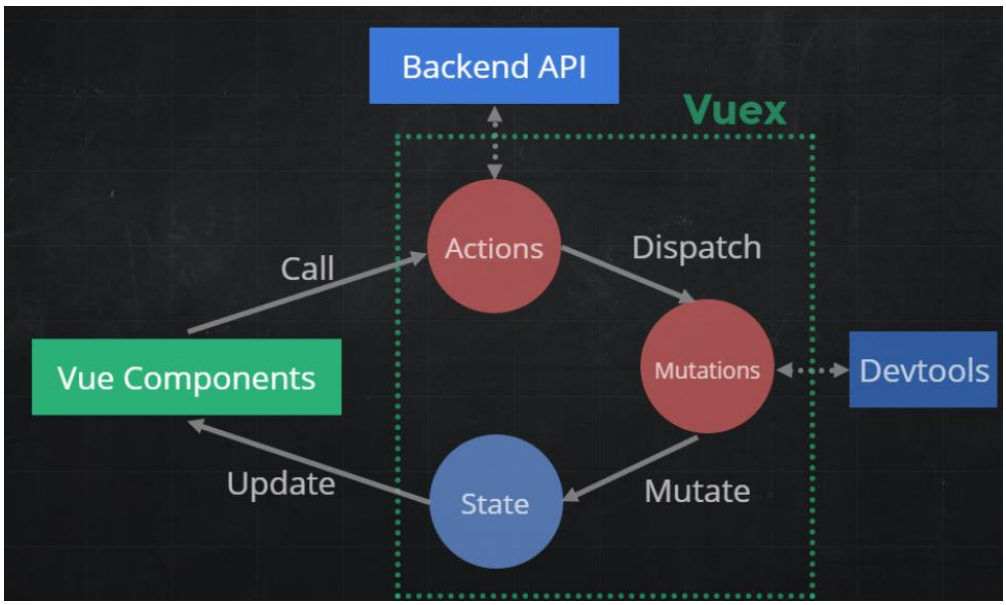

Рис 1.10. Vuex

На рисунку вище представлено додаток на Vue + Vuex з наступними частинами:

- Стан (State), який є єдиним джерелом даних для компонентів.
- Vue-компоненти (Vue-components), які по суті є лише декларативним відображенням статків.
- Дії (Actions), які відловлюють подію, що сталася, збирають дані з зовнішніх API і запускають потрібні мутації.
- Мутації (Mutations) - єдина частина, яка може змінювати стан і, отримавши дані від Actions, застосовує їх на стані().

vue-class-component

Vue-class-component - це бібліотека, що дозволяє створювати компоненти Vue в синтаксисі стилю класу. Наприклад, нижче наведено

простий компонент лічильника, написаний за допомогою компонента класу Vue:

```
<template>
  <div>
    <button v-on:click="decrement">-</button>
    {{ count }}
    <button v-on:click="increment">+</button>
  </div>
</template>
<script>
import Vue from 'vue'
import Component from 'vue-class-component'
// Define the component in class-style
@Component
export default class Counter extends Vue {
  // Class properties will be component data
  count = 0
  // Methods will be component methods
  increment() {
    this.count++
  }
  decrement() {
    this.count--
  }
}
</script>
```

Визначаючи компонент у стилі класу, можна не тільки змінити синтаксис, а й використовувати деякі функції мови ECMAScript.

vue-router

Vue Router - офіційна бібліотека маршрутизації для Vue.js. Вона глибоко інтегрується з Vue.js і дозволяє легко створювати SPA-додатки. Включає такі можливості:

- Вкладені маршрути/перегляди.
- Модульні налаштування маршрутизатора.
- Доступ до параметрів маршруту, query, wildcards.
- Анімація переходів подань на основі Vue.js
- Зручний контроль навігації.
- Автоматичне проставляння активного CSS класу для посилань.
- Режими роботи HTML5 history або хеш, з авто-перемиканням в IE9.
- Налаштовувана поведінка прокрутки сторінки.

Vuex

Vuex - патерн управління станом + бібліотека для додатків на Vue.js. Він служить централізованим сховищем даних для всіх компонентів програми з правилами. Стан може бути змінено тільки передбачуваним чином. Vuex інтегрується з офіційним розширенням vue-devtools , надаючи "з коробки" такі можливості, як "машину часу" для налагодження та експорт/імпорт зліпків стану даних.

```
new Vue({
 // стан
 data() {
  return {
   count: 0
  };
 },
 // перегляд
 template: `
  <div>{{ count }}</div>
```

```
  `,
  // дії
  methods: {
    increment() {
      this.count++;
    }
  }
});
```

Ця самостійна програма складається з таких частин:

1.Стан - "джерело істини", що управляє додатком;

2.Подання - відображення стану задане декларативно;

3.Дії - можливі шляхи зміни стану програми у відповідь на взаємодію користувача з поданням.

Однак простота швидко зникає, коли у нас з 'являється кілька компонентів, що ґрунтуються на одному і тому ж стані:

Декілька відображень можуть залежати від однієї частини стану програми.

Дії з різних уявлень можуть впливати на одні й ті самі частини стану програми.

Вирішуючи першу проблему, доведеться передавати одні й ті самі дані вхідними параметрами в глибоко вкладені компоненти. Це часто складно і втомливо, а для сусідніх компонентів таке і зовсім не спрацює. Вирішуючи другу проблему, можна прийти до таких рішень, як звернення за посиланнями до батьківських/дочірніх примірників або спроб змінювати і синхронізувати кілька копій стану через події. Обидва підходи крихки і швидко призводять до появи коду, який неможливо підтримувати.

Так чому б не винести весь загальний стан програми з компонентів і керувати ним в глобальному синглтоні? При цьому наше дерево компонентів стає одним великим "уявленням", і будь-який компонент може отримати доступ до стану програми або викликати дії для зміни стану, незалежно від того, де вони знаходяться в дереві!

Чітко визначаючи і поділяючи концепції, що виникають при управлінні станом, і вимагаючи дотримання певних правил, які підтримують незалежність між уявленнями і станами, ми краще структуруємо код і полегшуємо його підтримку.

Це основна ідея Vuex, натхненного Flux (opens new window), Redux (opens new window) і Архітектурою Elm (opens new window). На відміну від інших набоїв, Vuex реалізований у вигляді бібліотеки, спеціально призначеної для Vue.js, щоб використовувати його систему реактивності для ефективного оновлення.

1.3.1.3 Основні поняття

Єдине дерево стану

Vuex використовує єдине дерево стану - коли один об 'єкт містить весь глобальний стан програми і служить "єдиним джерелом істини". Це також означає, що в додатку буде тільки одне таке сховище. Єдине дерево стану дозволяє легко знайти потрібну його частину або робити знімки поточного стану програми з метою зневадження. Дані, які зберігаються у Vuex повинні слідувати тим же правилам, що і data в екземплярі Vue, тобто об 'єкт стану повинен бути простим.

Мутації

Єдиним способом зміни стану сховища у Vuex є мутації. Мутації у Vuex дуже схожі на події: кожна мутація має рядковий тип і функцію-обробник. У цьому обробнику і відбуваються зміни стану, переданого до функції першим аргументом:

```
const store = new Vuex.Store({
  state: {
    count: 1
  },
  mutations: {
```

```
  increment(state) {
    // изменяем состояние
    state.count++;
  }
 }
});
```

Викликати функцію-обробник безпосередньо - не можна. Це більше схоже на обробку події: "Коли мутація типу increment ініційована, викликається цей обробник". Щоб ініціювати обробку мутації, необхідно викликати store.commit, вказавши її тип:

```
store.commit('increment');
```

Дії

Дії - схожі на мутації з кількома відмінностями: Замість того, щоб безпосередньо змінювати стан, дії ініціюють мутації; Дії можуть використовуватися для асинхронних операцій.

Зареєструємо просту дію:

```
const store = new Vuex.Store({
 state: {
  count: 0
 },
 mutations: {
  increment(state) {
   state.count++;
  }
 },
 actions: {
  increment(context) {
   context.commit('increment');
  }
```

```
  }
});
```

Обробники дій отримують об 'єкт контексту, що містить ті ж методи і властивості, що і сам екземпляр сховища. На практиці для спрощення коду часто використовується деструктуризація аргументів (opens new window) з ES2015 (особливо при необхідності багаторазового виклику commit):

```
actions: {
  increment ({ commit }) {
    commit('increment')
  }
}
```

Додатки

Використовуючи єдиний дерево стану, всі стани програми містяться всередині одного великого об 'єкта. Однак, у міру зростання і масштабування програми, сховище може істотно роздутися [25].

Щоб допомогти в цій біді, Vuex дозволяє розділяти сховище на модулі. Кожен модуль може містити власний стан, мутації, дії, геттери і навіть вбудовані підмодулі - структура фрактальна:

```
const moduleA = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... },
  getters: { ... }
}
const moduleB = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... }
}
```

1.3.2 Порівняння з відомими аналогами

1.3.2.1 React

React і Vue багато в чому схожі. Вони обидва:

- використовувати віртуальну модель DOM
- забезпечить реактивність і компонентну структуру
- фокусуються на додаткових бібліотеках, виносячи інші питання, такі як роутинг або управління глобальним станом додатки, в додаткових бібліотеках.

У порівнянні вказується, де React перевершує Vue, наприклад – у багатстві екосистеми і достатку доступних для користувача засобів відтворення.

Швидкість виконання

Як React, так і Vue - виключно швидкі, тому швидкість навряд чи буде вирішальним фактором при виборі між ними.

Зусилля для оптимізації

У React коли стан компонента змінюється, він запускає повторну відрисовку всього піддерева компонента, починаючи з себе. Щоб уникнути непотрібної повторної відрисовки 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. Однак, в деяких випадках ви не можете розраховувати на таку оптимізацію, тому що PureComponent / shouldComponentUpdate припускають, що відображення всього поддерева визначається даними поточного компонента. Якщо це не так, то така оптимізація може призвести до неузгодженим станом DOM.

У Vue залежності компонента автоматично відслідковуються під час відрисовки, 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 для розробника, що дозволяє більше зосередитися на побудові самого додатка в міру його масштабування.

HTML & CSS

У React абсолютно все - це JavaScript. Не тільки структури HTML, виражені через JSX, останні тенденції також включають управління CSS всередині JavaScript. Цей підхід має свої переваги, але також змушує йти на компроміси, які можуть здатися неефективними для кожного розробника. Vue охоплює класичні веб-технології та ґрунтується на них.

JSX vs Шаблони

У React всі компоненти реалізують свій UI в render-функціях з використанням JSX, декларативним XML-подібним синтаксисом, який працює в JavaScript.

Render-функції з JSX мають кілька переваг:

- Ви можете використовувати всі можливості мови програмування (JavaScript) для побудови свого уявлення. Це включає в себе тимчасові змінні, управління розгалуженням і прямі посилання на значення JavaScript в області видимості.

- Підтримка інструментів (наприклад, лінтінг, перевірка типів, автодоповнення в редакторі) для JSX в деяких відношеннях більш просунута, ніж те, що є в даний час для шаблонів Vue. У Vue у нас також є render-функції і навіть підтримка JSX, тому що іноді вам потрібні ці можливості. Проте, за замовчуванням ми пропонується шаблони як більш проста альтернативу.

Будь-валідний HTML також буде дійсним шаблоном Vue, і це призводить до виникненню кількох переваг:

- Для багатьох розробників, які працюють з HTML, шаблони просто більш природні для читання і написання. Саме ця перевага може бути кілька суб'єктивною, але якщо це робить розробника більш продуктивним, то перевага очевидна.
- HTML-шаблони полегшують поступову міграцію існуючих додатків для використання можливостей реактивності Vue.
- 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.

Деякі стверджують, що важливо вивчити додаткову мову DSL (Domain-Specific Language) для написання шаблонів - ця різниця в кращому випадку поверхнева. По-перше, JSX не означає, що користувачеві не потрібно нічого вчити - це додатковий синтаксис на основі простого JavaScript, тому він простий в освоєнні для будь-якого, хто знайомий з JavaScript, але говорити, що це просто для всіх, було б помилкою.

Точно також шаблон є просто додатковим синтаксисом поверх простого HTML і, отже, має дуже низький поріг входження для тих, хто вже знайомий з HTML. За допомогою DSL можна зробити більше з меншою кількістю коду (наприклад, з v-on модифікаторами). Схожа задача може включати в себе набагато більше коду при використанні простих функцій JSX або render-функцій.

На більш високому рівні можна розділити компоненти на дві категорії: презентаційні та логічні. Рекомендується використовувати шаблони для презентаційних компонентів і render-функції / JSX для логічних. Процентне співвідношення цих компонентів залежить від типу вашої програми, але зазвичай презентаційні компоненти більш поширені.

Модульный (компонентний) CSS

За винятком випадків поділу компонентів на кілька файлів (наприклад, за допомогою CSS-модулів), для обмеження області видимості CSS в React зазвичай використовується підхід CSS-in-JS (наприклад, styled-components і emotion). Це являє собою новий компонентно-орієнтований підхід до стилізації, який відрізняється від звичайного процесу розробки CSS. Крім того, незважаючи на підтримку вилучення CSS в окремий файл стилів на етапі складання, як і раніше може бути необхідно, щоб під час виконання були підключені в збірку для коректної роботи стилізації. У той час, як ви отримуєте доступ до динамічності JavaScript при створенні ваших стилів, цей компроміс часто збільшує розмір збірки і час виконання.

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.

Однофайловий компоненти надають вам повний доступ до CSS в тому ж файлі, що і решта коду компонента.

```
<style scoped>
  @media (min-width: 250px) {
    .list-container:hover {
      background: orange;
    }
  }
</style>
```

Опціональний атрибут scoped автоматично обмежує область видимості CSS поточним компонентом, додаючи елементам унікальні атрибути (такі як data-v-1-21e5b78), і компілюючи .list-container: hover будь-що-небудь на зразок .list-container [data-v-1 -21e5b78]: hover.

Нарешті, стилізація в однофайлових компонентах Vue дуже гнучка. За допомогою vue-loader, ви можете використовувати будь-який препроцесор,

постпроцесор і навіть глибоку інтеграцію з CSS-модулями - все в елементі <style>.

Масштабування вгору

Для великих додатків, як Vue так і React надають надійні рішення для роутинга. Спільнота React також породила досить інноваційні рішення в галузі управління станом додатків (див. Flux / Redux). Ці підходи, і навіть сам Redux легко інтегруються в додатки на Vue. Насправді, Vue зробив наступний крок, створивши Vuex - натхненну Elm реалізацію паттерна управління станом додатки. Vuex глибоко інтегрований з Vue, що, на наш погляд, неабияк полегшує життя розробникам.

В якості ще одної важливої відмінності між React і Vue можна згадати той факт, що всі додаткові бібліотеки Vue, включаючи бібліотеки для керування станом додатка і для роутинга офіційно підтримуються в актуальному відповідно до ядру 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.

Нарешті, у Vue є CLI генератор проектів, який дозволяє легко почати новий проект за допомогою інтерактивного майстра. Його також можна використовувати для миттєвого прототипування компонента. React також робить успіхи в цій галузі за допомогою create-react-app, але в даний час у нього є кілька обмежень:

- Він не допускає ніякої конфігурації під час створення проекту, в той час як Vue CLI працює поверх оновлюваної runtime-залежності, яка легко розширюється за допомогою плагінів.
- Існує тільки один шаблон для односторінкового додатка, в той час як Vue пропонує широкий вибір опцій за замовчуванням для різних цілей і систем збірки.

- Немає можливості створювати проекти з користувацьких пресетів налаштувань, що може бути особливо корисно для enterprise-оточень з усталеними раніше угодами.

Важливо зауважити, що багато з цих обмежень - слідства свідомо прийнятих рішень команди create-react-app, і в них є і свої плюси. Наприклад, якщо потреби вашого проекту дуже прості і вам ніколи не потрібно «витягувати» для налаштування та автоматизація виробництва, ви зможете оновити його як залежність.

Масштабування вниз

React відомій своєю досить крутою кривій вивчення. До моменту, коли новачок зможе щось написати, Йому придется дізнатіся про JSX, як ймовірно - і про ES2015 +, оскількі багато хто використовує синтаксис ES2015-класів. Крім того, існує можлівість розібратіся з системою збирання, оскількі, хоча технічно и існує можлівість використовуват Babel самостійно для live-компіляції коду, для production цею підхід в будь-якому випадку не годиться.

Vue масштабується вгору нітрохи не гірше, ніж React, і в той же час його можна масштабувати і вниз - аж до варіанту використання разом з jQuery. Починаючи з цього моменту можна писати код на Vue, і навіть використовувати production-версію.

Оскільки знання JSX, ES2015 і систем збірки не потрібно для початку роботи з Vue, в середньому у нових розробників йде не більше дня на вивчення технології, що дозволяє дізнатися досить для побудови нетривіальних додатків.

Нативна відрисовка

React Native дозволяє писати нативні додатки для iOS і Android, використовуючи ту ж саму модель компонентів React. Це дозволяє

розробникам застосувати знання одного і того ж фреймворка на різних платформах. У цій області, Vue офіційно підтримує проект Weex - багатоплатформовий UI-фреймворк, створений Alibaba Group і Apache Software Foundation (ASF). Weex дозволяє використовувати той же синтаксис Vue для створення компонентів, які не тільки можуть відображатися в браузері, а й також нативними елементами в iOS і Android.

На даний момент Weex все ще перебуває в активній фазі розробки, і ще не настільки перевірений досвідом, як React Native. Однак, його розробка мотивується реальними вимогами найбільшого бізнесу електронної комерції в світі. Команда розробки Vue також активно взаємодіє з розробниками Weex, гарантуючи відсутність несподіванок для Vue-розробників. Ще один варіант NativeScript-Vue - плагін для NativeScript для створення по-справжньому нативних додатків за допомогою Vue.js.

Порівняння з MobX

MobX став досить популярним в співтоваристві React. Він використовує майже ідентичну Vue систему реактивності. У певному сенсі, зв'язку React + MobX можна вважати дещо багатослівним варіантом Vue, так що якщо ви використовуєте її, і вона вам подобається, перехід на Vue може виявитися наступним логічним кроком.

Preact та інші React-подібні бібліотеки

React-подібні бібліотеки зазвичай намагаються використовувати якомога більше своїх API і екосистеми React наскільки це можливо. З цієї причини більшість порівнянь, наведених вище, також застосовні і до них. Головною відмінністю, як правило, буде зменшення доступної екосистеми (часто значно) в порівнянні з React. Оскільки ці бібліотеки не можуть бути на 100% сумісні з усім в екосистемі React, деякі бібліотеки інструментів або супутні бібліотеки можуть не використовуватися. Або, навіть якщо схоже що

вони працюють, вони можуть зламатися в будь-який час, якщо ваша конкретна React-подібна бібліотека не підтримується нарівні з React.

1.3.2.2. Angular

TypeScript

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.

Швидкість виконання

В аспекті продуктивності, обидва фреймворка вельми швидкі з дуже схожими метриками в тестах. Ви можете вивчити конкретні цифри для більш детального порівняння, але швидкість навряд чи стане вирішальним фактором.

Розмір

Останні версії Angular, з AOT-компіляцією і tree-shaking, змогли значно зменшити розмір збірок. Однак повнофункціональний проект Vue з включеними Vuex + Vue Router (~ 30Кб gzip) як і раніше значно легше з коробки, ніж AOT-скомпільований додаток, створене за допомогою angular-cli (~ 65КБ gzip).

Гнучкість

Vue на відміну від Angular підтримує безліч різних систем збірок, не обмежуючи розробників в тому, яку структуру використовувати для додатка. Багатьом розробникам ця свобода подобається, хоча є і ті, хто вважають за краще мати єдино правильний спосіб побудови програми.

Крива навчання

Все що необхідно для початку роботи з Vue - це знайомство з HTML і звичайним (ES5) javascript'ом. З цими базовими навичками ви вже можете почати будувати нетривіальні програми після менш ніж одноденного вивчення документації.

Крива навчання у Angular набагато крутіше. API фреймворка просто величезна і вам як користувачу потрібно буде розібратися з великою кількістю концепцій, перш ніж стати продуктивним. Очевидно, що складність Angular багато в чому обумовлена його спрямованістю тільки на великі, комплексні програми - але це робить платформу набагато важчою для менш досвідчених розробників. [26]

РОЗДІЛ 2
ПРОЕКТНІ РІШЕННЯ ПО ЗАВДАННЮ

2.1. Вибір і обгрунтування рішення поставленої задачі

Предметом розробки є веб-сайт аналізу виїзду мешканців України закордон. Сайт призначений для у встановленні максимальної точної кількості українців, які виїжджають за кордон. Сайт може використовуватися: державних установах для аналізу причин та кількості міграції українців; пересічними громадянами для ознайомлення динаміки міграції населення України.

Метою розробки є створення веб-сайту який містить статистику міграції мешканців та є зручним у використаннях для будь-яких пристроїв.

При створенні сайту були пройдені наступні етапи розробки:

1. Створення структури сайта;
2. Створення дизайну сайта;
3. Поступове впровадження Vue.Js.
4. Організація сховища данних у файлі в фоматі JSON та отримання їх за допомогою Axios-запиту, і виведення їх на сторінку з використанням Chart.js.
5. Фіналізація.

2.2. Вибір і обгрунтування використання програмних засобів

В якості середовище розробки був обраний текстовий редактор коду **Visual Studio Code**[27], який має зручний та інтуїтивно зрозумілий інтерфейс(рис. 2.1) та є зручним для роботи в галузі веб розробки. Він підтримує ряд мов програмування, підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігацію по коду, підтримку Git та інші можливості. Visual Studio також дозволяє замінювати кодову сторінку при збереженні документа, символи перекладу рядка і мову програмування

поточного документа. Файли та папки проектів зберігаються у вигляді у вигляді древовидної структури.

На березень 2019 року за допомогою вбудованого в продукт призначеного для користувача інтерфейсу можна завантажити і встановити кілька тисяч розширень тільки в категорії «programming languages» (мови програмування).

Також розширення дозволяють отримати більш зручний доступ до програм, таким як Docker, Git та інші. В розширеннях можна знайти Лінтер коду, теми для редактора і підтримку синтаксису окремих мов. Крім того в редакторі є вбудований термінал який можна відкрити і прибрати поєднанням клавіш ctrl + `, що є зручно для і не потрібно перемикатися між вікнами в процесі розробки, якщо потрібен доступ до командного рядка.

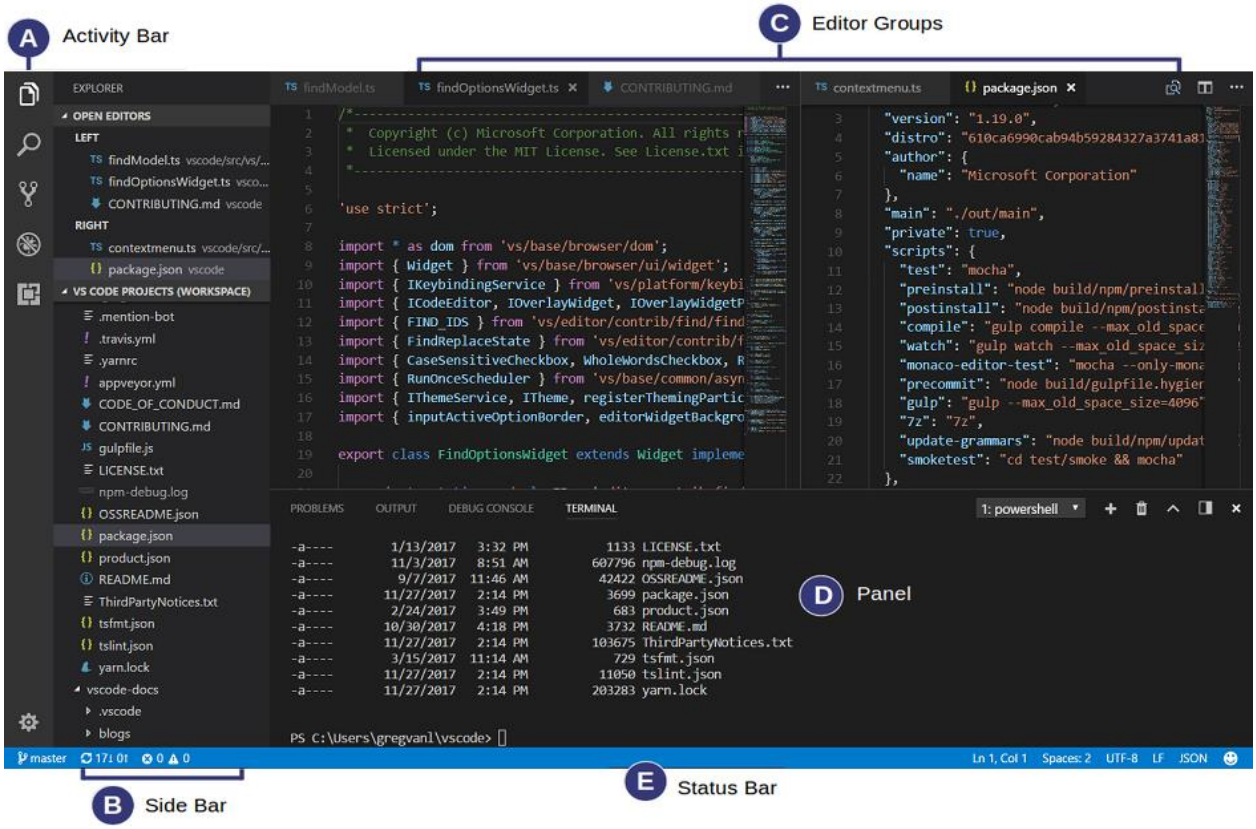

Рис 2.1. Інтерфейс VsCode

Сайт розроблявся на мові програмування javascript та мові розмітки html і таблиць стилів css з використанням препроцесору SCSS, фреймворку

vue.js(його поступовим впровадженням) та бібліотеки для створення графіків chart.js. Дані зберігаються у форматі JSON їх отримання організується за допомогою бібліотеки Axios шляхом axios-запитів до файлу JSON, у якому зберігаються ці дані.

Vue.js – javaScript-фреймворк з відкритим вихідним кодом для створення користувацьких інтерфейсів, легко інтегрується в проекти з використанням інших JavaScript-бібліотек і може бути використаний для поступового впровадження. Може функціонувати як веб-фреймворк для розробки односторінкових додатків в реактивному стилі.

SCSS(Sassy css) – один з синтаксисів препроцесору SASS. Навідміно від класичного SASS, більше схожий на структуру CSS, і є зручнішим у використанні. Препроцесор має багато можливостей та велику спільноту розробників і може бути інтегрований у Vue.js.

Axios – це широко відома JavaScript-бібліотека. Вона являє собою HTTP-клієнт, заснований на промісах і призначених для браузерів і для Node.js.

Приклад get-запиту:

```
axios.get("https://jsonplaceholder.typicode.com/todos/1".then(response=>
console.log("response", response.data))
```

JSON (Javascript Object Notation) - це відкритий стандартний формат файлу і формат обміну даними, для зберігання і передачі об'єктів даних, що складаються з пар атрибут-значення і масивів. Це дуже поширений формат даних з різноманітним набором додатків, одним із прикладів яких є веб-додатки, які взаємодіють з сервером. JSON - це формат даних, що не залежить від мови. Він був отриманий з JavaScript, але багато сучасних мов програмування включають код для генерації та аналізу даних в форматі JSON. Імена файлів JSON використовують розширення .json. [28] [29]

Chart.js –це популярний інструмент, який призначений для створення графіків і діаграм. Можна створювати адаптивні діаграми будь-якої складності на основі HTML5 Canvas.

2.3. Розробка сайту

На етапі розробки сайту створювалась його структура за допомогою HTML, оформлення та медіазапити для адаптації сайту під різні пристрої за допомогою препроцесору SCSS, інтерактивність сайту була реалізована за допомогою javascript, з поступовим впровадженням фреймворку Vue.js. Сховище даних організовулося у файлі data.json.

Отримання цих данних було реалізовано за допомогою axios get-запиту. Далі ці дані були виведені на сторінку через діаграми з використанням популярної бібліотеки chart.js.

2.3.1 Створення структури сайту

Структура була створена за допомогою мови гіпертекстової розмітки(рис. 2.2) HTML за сучасними стандартами шляхом блокової верстки, коли кожна частина сайту поділяється на контейнери і семантика відокремлюється від стилів.

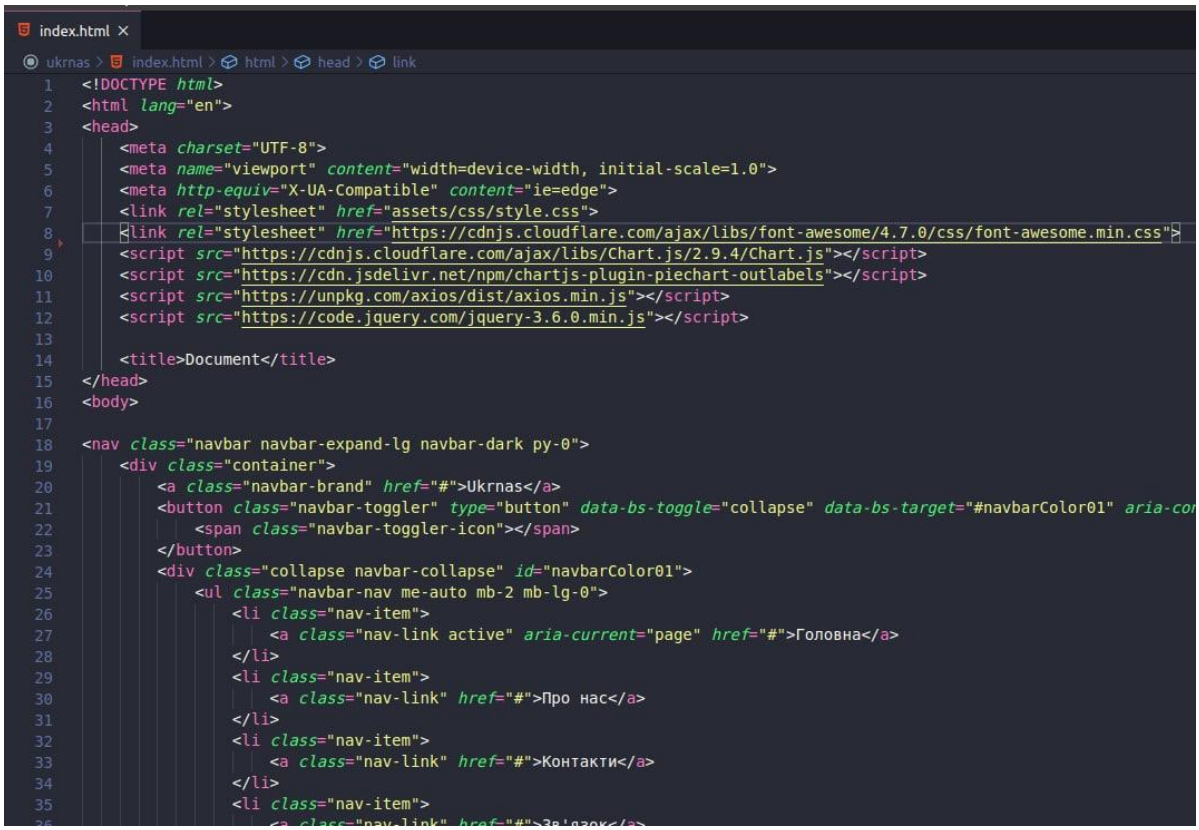

Рис. 2.2. Розмітка сайту

2.3.2. Створення дизайну сайта

При розробки дизайну сайта були використані стандартні технології для оформлення сайту за допомогою CSS-препроцесору SCSS(рис. 2.3) для лаконічності та чистоти коду, а такожнаписання медіазапитів(рис. 2.4) для адаптивності сайту під різні пристрої.

```scss
* {
    margin: 0;
    padding: 0;
    outline: none;
    box-sizing: border-box;
}
body {
    background-color: #24252F !important;
}
nav {
    background-color: #242F39;
    .navbar-brand {
        font-family: Arial, Helvetica, sans-serif;
        text-transform: uppercase;
        font-size: 20px;
        background-color: #0000d6;
        background-image: linear-gradient(#0000d6, #ffda00);
        padding: 15px;
        letter-spacing: -1px;
        font-weight: 600;
        color: #ffffff;
    }
    li {
        a {
            &:hover {
                color: #B22222 !important;
            }
        }
    }
}
```

Рис. 2.3. SCSS стилі

```scss
@media (max-width: 1245px) {
    nav {
        padding: 0 5%;
    }
}
@media (max-width: 1140px) {
    nav {
        padding: 0px;
        .logo {
            flex: 2;
            text-align: center;
        }
        .nav-links {
            position: fixed;
            z-index: 99;
            top: 70px;
            width: 100%;
            left: -100%;
            height: 100%;
            display: inline-block;
            padding: 10px 50px 0 50px;
```

Рис 2.4. Медіазапити

2.3.3. Поступове впровадження Vue.JS

Сайт створений за допомогою мови javascript з використанням фреймворку Vue.js шляхом його поступового впровадження. Створимо два компоненти(рис. 2.5) App.vue – компонент навігації сайту і About.vue – компонент футеру сайту. Які знаходяться, у папки views.

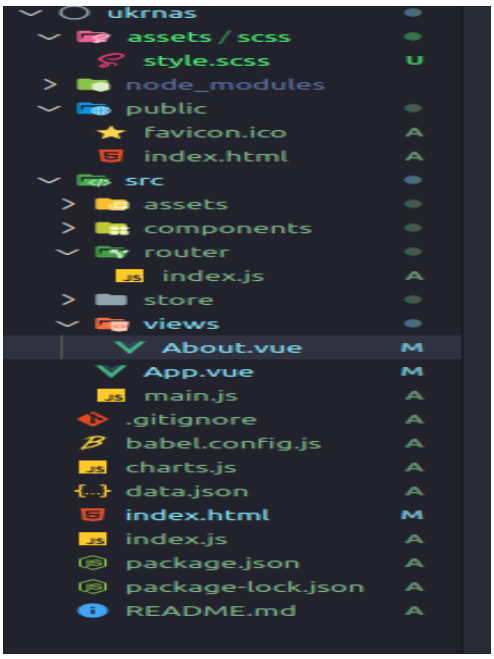

Рис 2.5. Файлова структура Vue.js

Для виводу розмітки через vue.js потрібно вставити її у контейнер template, який служить для виводу вмісту(рис. 2.6).

```
<template>
  <div id="app">
    <div id="nav">
      <router-link to="/">Головна</router-link> |
      <nav class="navbar navbar-expand-lg navbar-dark py-0">
    <div class="container">
        <a class="navbar-brand" href="#">Ukrnas</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarColor01">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="#">Го
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Про нас</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Контакти</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Зв'язок</a>
                </li>
            </ul>
            <form class="d-flex">
                <input class="form-control me-2 bg-dark text-white border-0" t
                <button class="btn btn-outline-light bg-danger border-0" type=
            </form>
        </div>
    </div>
</nav>
    </div>
    <router-view/>
  </div>
</template>
```

Рис 2.6. Розмітка vue-компоненту

Далі можна вивести стилі в тегу style з атрибутом style scoped, який означає, що підключені стили діють тільки всередині певного компоненту, в даному випадку App.vue

```scss
<style scoped lang="scss">
nav {
    background-color: #242F39;
    .navbar-brand {
        font-family: Arial, Helvetica, sans-serif;
        text-transform: uppercase;
        font-size: 20px;
        background-color: #0000d6;
        background-image: linear-gradient(#0000d6, #ffda00);
        padding: 15px;
        letter-spacing: -1px;
        font-weight: 600;
        color: #ffffff;
    }
    li {
        a {
            &:hover {
                color: #B22222 !important;
            }
        }
    }
}
</style>
```

Рис. 2.7. Стилі компоненту App.vue

Тепер файли index.html(рис. 2.8) та style.scss(рис. 2.9) виглядають наступним чином:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="assets/css/style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-piechart-outlabels"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <title>Document</title>
</head>
<body>


<center>
    <h4 class="text-white mt-4 mb-2">Налення з 90 по 2021 рік в мільйонах</h4>
    <canvas id="from90_to21" style="width:100%;max-width:800px; background:white;margin-bottom:40px"></canvas>

    <h4 class="text-white mb-2">Населення по областях(тис)</h4>
    <canvas id="Oblasti" style="width:100%;max-width:800px;height:500px;background:white;"></canvas>

    <div class="text-white my-4">
        <h4>Кількість населення за поточний місяць, млн</h4>
        <p id="currentpop"></p>

        <h4>Виїхало з країни за поточний місяць</h4>
        <p id="currentmonth"></p>

        <h4>Українська діаспора згідно всесвітнього конгресу України/h4>
        <p id="vkudiasp"></p>
    </div>
</center>


  <script src="charts.js"></script>
  <script src="index.js"></script>
</body>
</html>
```

Рис. 2.8. Файл index.html

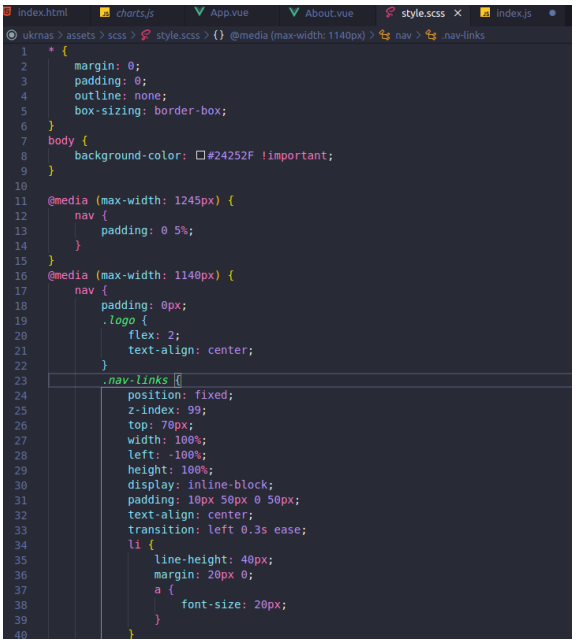

Рис. 2.9. Файл style.scss

2.3.4. Організаці зберігання даних та їх отримання

Дані зберігаються у файлі файлі data.json.

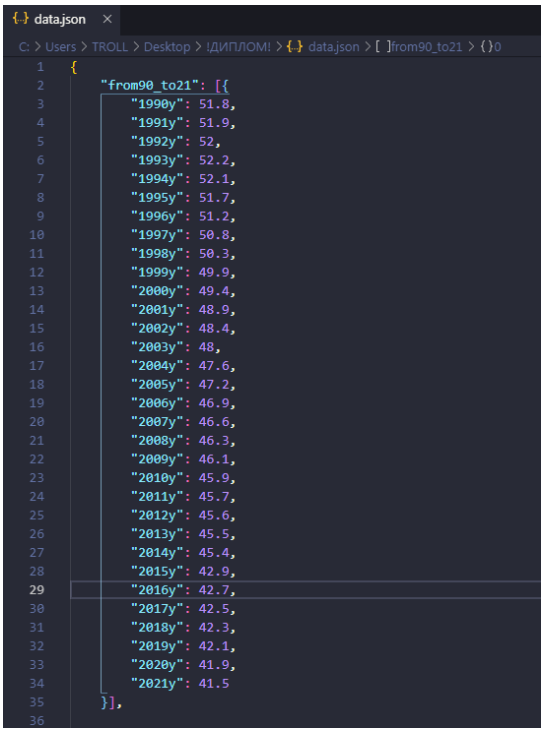

Рис. 2.10. Файл data.json

Вивід даних реалізований за допомогою бібліотеки axiox, потім вони передаються у діагарами chart.js. Спочатку здійснюється axios-запит до файлу

data.json, потім створюється код діагарм і вивід цих даних на сторінку через бібліотеку chart.js.

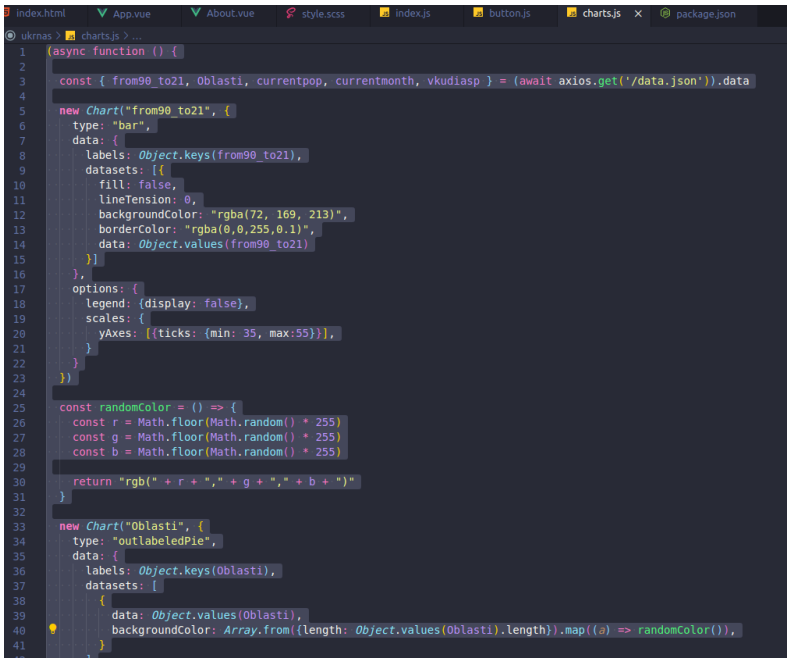

Рис. 2.11. Вивід даних методом axios-запиту

Код на рисунку(2.11.) (await axios.get('/data.json')).data –цеAxios get-запит до файлу data.json, передаються дані через chart.js.

2.3.5. Фіналізація проекту

На фінальному етапі розробки потрібно розмістити сайт у мережу, щоб його могли переглядати онлайн. Для публікації сайту необхідно зробити 2 основних етапи:

1. Арендувати доменне ім’я.
2. Арендувати хостінг.

Сайт створювався у навчальних цілях тому не призначений для масового поширення. Замість покупки доменного ім’я та аренди хостінгу для глобального доступу у мережі – він буде завантажений на безкоштовну версію хостінгу webhost(рис. 2.12).

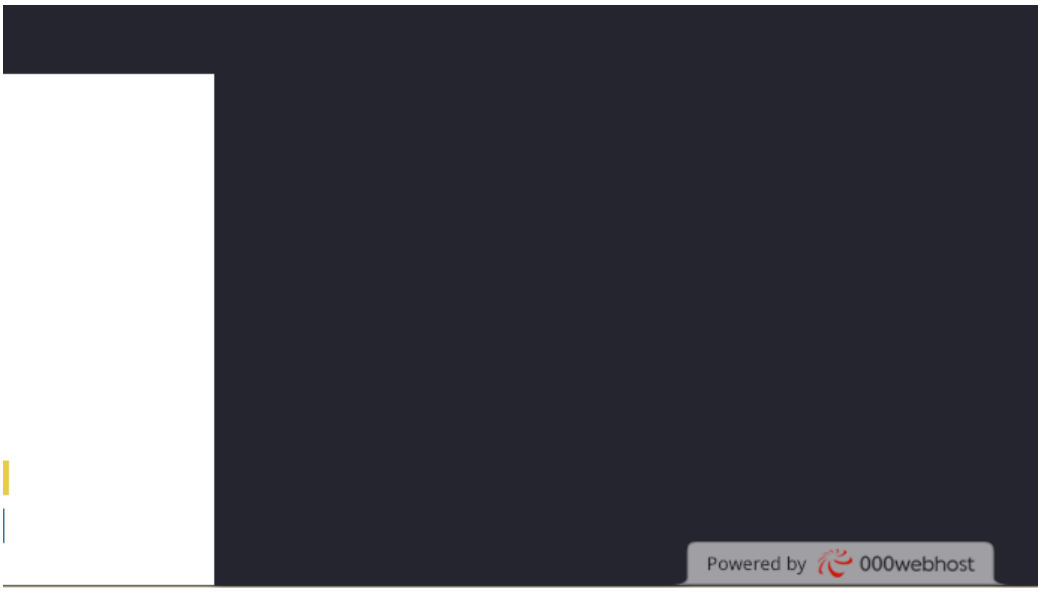

Рис 2.12. Розміщення сайту на webhost.

Актуальна версія сайту доступна за посиланням:
https://ukrnas.000webhostapp.com/

ВИСНОВОК

Мета дипломної роботи – розробка аналітичного веб-сайту аналізу виїзду мешканців України за кордон, з використанням javascript-фреймворку Vue.js, бібліотеки axios, для отримання даних з файлу формату JSON та бібліотеки для створення діаграм chart.js.

Розробка проводилася поетапно. Спочатку була сформована ідея проекту та створена структура сайту за допомогою HTML. Обрана мова програмування Javascript, як найбільш адаптована для веб-розробки. Для оформлення дизайну сайта та створення його адаптивності була використана технологія CSS з використанням препроцесору SCSS, та впровадженню медіазапитів, для надання можливості адаптуватися під будь-які пристрої.

Далі була продемонстрована можливість поступово в проваджувать vue.js у проект, частина сайту була створена у вигляді окремих компонентів.

Збереження даних було організоване у файлі формату JSON та їх отримання було здійснене за допомогою бібліотеки axios шляхом get-запиту. Потім здійснювалася організація виводу цих даних на сторінку з використанням діаграм chart.js.

На фінальній стадії розробки сайт був розміщений на хостинг webhostapp. Розроблений сайт може бути використаний в будь-яких державних установах для аналізу причин та кількості міграції українців та доступне для пересічних громадян для ознайомлення динаміки міграції населення України.

У результаті проведеної роботи над дипломним проектом був розроблений веб-сайт та розміщено його на хостинг у вільний доступ.

В майбутньому планується повне впровадження фреймворку vue.js, додання інтерактивності та поліпшення дизайну.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SUMDU_Sushchenko. SUMDU/out2018/Sushchenko_bak_rob.pdf

2. Лебединець. Лебединець Максим_перевірка_Інформаційне забезпечення підтримки Web-сайту рекламного інтернет агентства.docx

3. Мустафаєв. МУСТАФАЄВ short.doc

4. HTML. (Електрон. Ресурс) / Спосіб доступу: URL: https://en.wikipedia.org/wiki/HTML

5. HTML. (Електрон. Ресурс) / Спосіб доступу: URL: https://developer.mozilla.org/ru/docs/Web/HTML

6. CSS. (Електрон. Ресурс) / Спосіб доступу: URL: https://en.wikipedia.org/wiki/CSS

7. Препроцесори. (Електрон. Ресурс) / Спосіб доступу: URL: https://mrmlnc.gitbooks.io/less-guidebook-for-beginners/content/chapter_1/css-reprocessors.html

8. Javascript. (Електрон. Ресурс) / Спосіб доступу: URL: https://developer.mozilla.org/ru/docs/Web/JavaScript

9. Javascript. (Електрон. Ресурс) / Спосіб доступу: URL: https://en.wikipedia.org/wiki/JavaScript

10. Переваги й недоліки Javascript. (Електрон. Ресурс) / Спосіб доступу: URL: https://ruseller.com/lessons.php?rub=32&id=2852

11. Chart.js. (Електрон. Ресурс) / Спосіб доступу: URL: https://ruseller.com/lessons.php?rub=32&id=2852

12. DOM. (Електрон. Ресурс) / Спосіб доступу: URL: https://developer.mozilla.org/en-US/docs/Glossary/DOM

13. DOM Standards. (Електрон. Ресурс) / Спосіб доступу: URL: https://en.wikipedia.org/wiki/Document_Object_Model

14. Single-page application. (Електрон. Ресурс) / Спосіб доступу: URL: https://en.wikipedia.org/wiki/Single-page_application

15. Single-page application. (Електрон. Ресурс) / Спосіб доступу: URL:

https://webpromoexperts.net/blog/razbiraemsya-s-spa-i-pwa/#:~:text=%D0%92%20%D1%81%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D0%B8%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20SPA%2D%20%D0%B8,%D0%BE%D1%84%D0%BB%D0%B0%D0%B9%D0%BD%2D%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5%20%D1%81%20%D0%BC%D0%BE%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%BE%D0%B9.

16. Ajax. (Електрон. Ресурс) / Спосіб доступу: URL: https://en.wikipedia.org/wiki/Ajax_(programming)

17. Як працює Ajax. (Електрон. Ресурс) / Спосіб доступу: URL: https://www.w3schools.com/xml/ajax_intro.asp

18. VueJs. (Електрон. Ресурс) / Спосіб доступу: URL: https://ru.vuejs.org/v2/guide/index.html

19. VueJs. (Електрон. Ресурс) / Спосіб доступу: URL: https://ru.wikipedia.org/wiki/Vue.js

20. Концепції VueJs. (Електрон. Ресурс) / Спосіб доступу: URL: https://www.c-sharpcorner.com/article/an-overview-of-vue-js-frontend-framework/

21. Директиви. (Електрон. Ресурс) / Спосіб доступу: URL: https://ru.vuejs.org/v2/api/index.html#%D0%94%D0%B8%D1%80%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D1%8B

22. Призначення директив. (Електрон. Ресурс) / Спосіб доступу: URL: https://www.w3schools.com/whatis/whatis_vue.asp

23. Компоненти. (Електрон. Ресурс) / Спосіб доступу: URL: https://ru.vuejs.org/v2/guide/components.html

24. Ткаченко. Диплом Ткаченко 126-17-1.pdf

25. Основні поняття. (Електрон. Ресурс) / Спосіб доступу: URL: https://habr.com/ru/post/329452/

26. Comparison. (Електрон. Ресурс) / Спосіб доступу: URL: https://ru.vuejs.org/v2/guide/comparison.html

27. VsCode. (Електрон. Ресурс) / Спосіб доступу: URL:
https://ru.wikipedia.org/wiki/Visual_Studio_Code
28. JSON. (Електрон. Ресурс) / Спосіб доступу: URL:
https://en.wikipedia.org/wiki/JSON
29. JSON. 2021_ФеІ-43_Андрушко_П.М._текст.pdf

ДОДАТОК А

ДИЗАЙН САЙТА

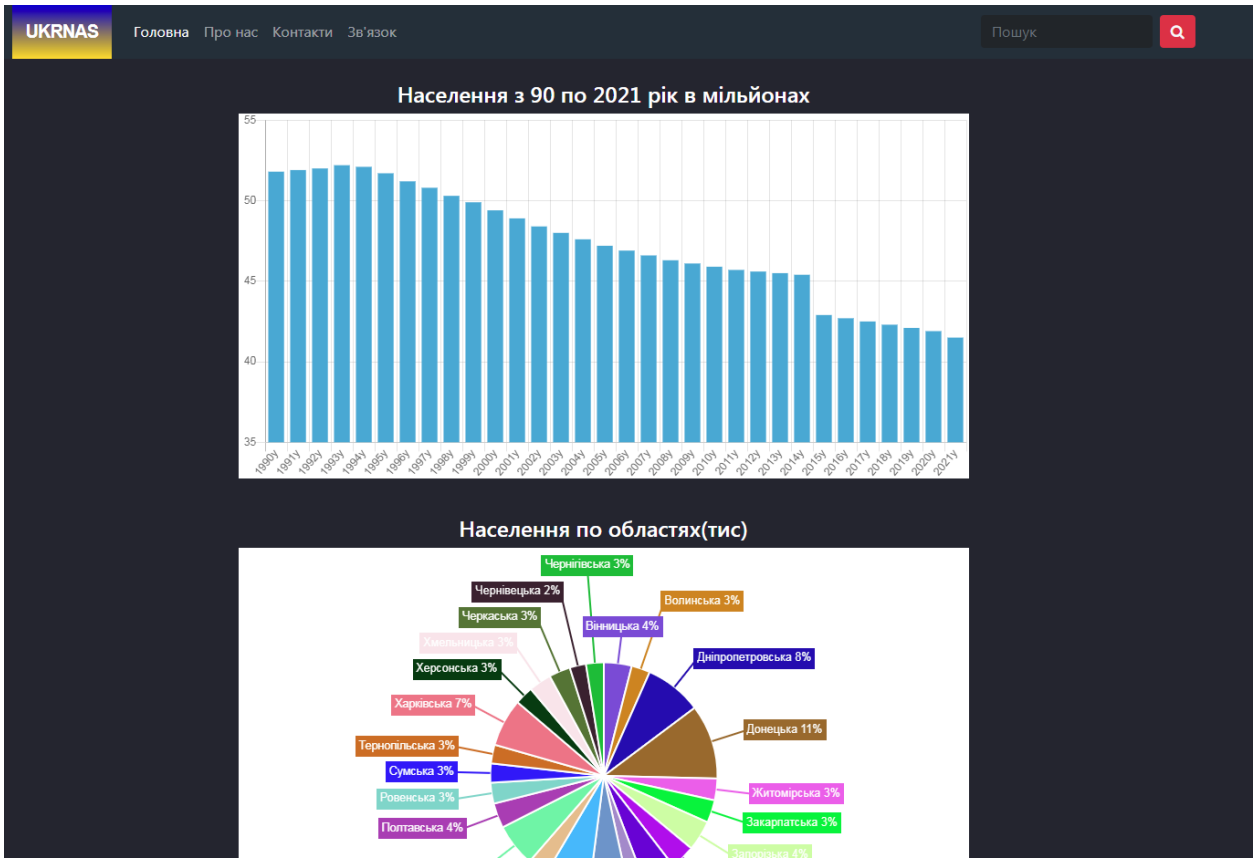

Рис. А.1.1 Верхня частина сайта

Рис. А.1.2 Нижня частина сайта

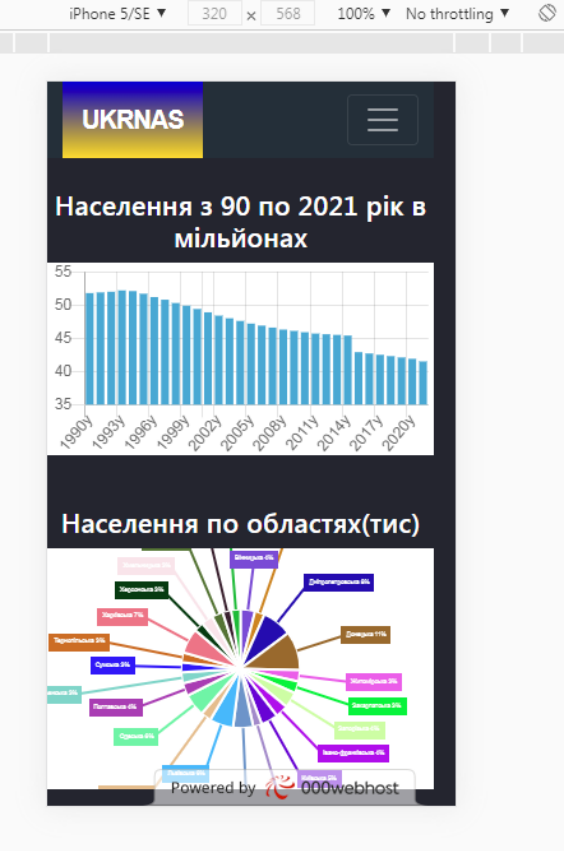

Рис. А.1.3 Мобільна версія

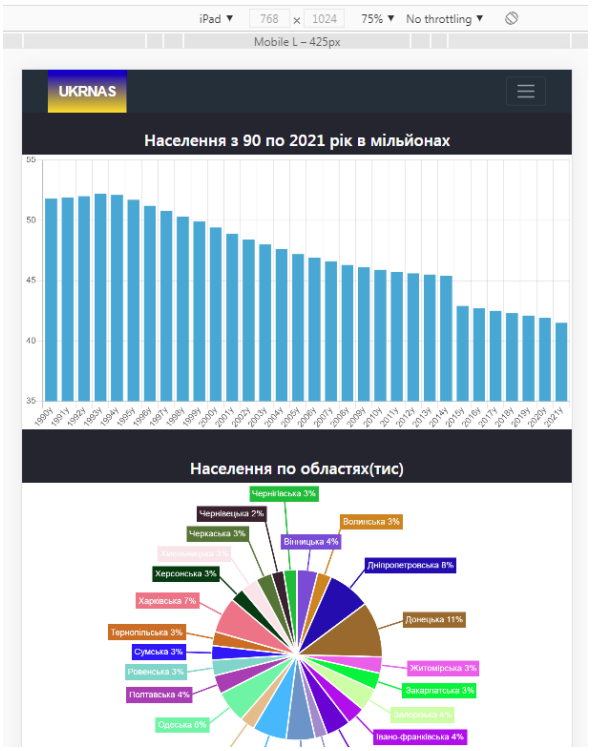

Рис. А.1.4 Вигляд на планшеті

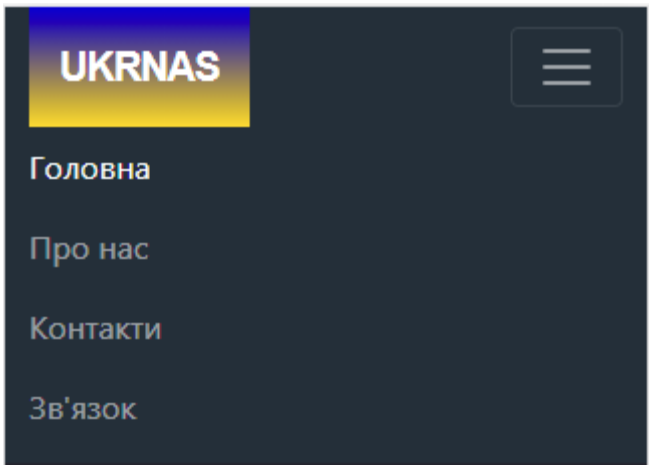

Рис. А.1.5 Бургер меню

ДОДАТОК Б

ВИХІДНИЙ КОД САЙТА

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="assets/css/style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-piechart-outlabels"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <title>Document</title>
</head>
<body>

<center>
    <h4 class="text-white mt-4 mb-2">Налення 3 90 по 2021 рік в мільйонах</h4>
    <canvas id="from90_to21" style="width:100%;max-width:800px; background:white;margin-bottom:40px"></canvas>

    <h4 class="text-white mb-2">Населення по областях(тис)</h4>
    <canvas id="Oblasti" style="width:100%;max-width:800px;height:500px;background:white;"></canvas>

    <div class="text-white my-4">
        <h4>Кількість населення за поточний місяць, млн</h4>
        <p id="currentpop"></p>

        <h4>Виїхало з країни за поточний місяць</h4>
        <p id="currentmonth"></p>

        <h4>Українська діаспора згідно всесвітнього конгресу України/h4>
        <p id="vkudiasp"></p>
    </div>
</center>


  <script src="charts.js"></script>
  <script src="index.js"></script>
</body>
</html>
```

App.vue

```
<template>
  <div id="app">
```

```
    <div id="nav">
      <router-link to="/">Головна</router-link> |
        <nav class="navbar navbar-expand-lg navbar-dark py-0">
    <div class="container">
        <a class="navbar-brand" href="#">Ukrnas</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarColor01">
          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link active" aria-current="page" href="#">Головна</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Про нас</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Контакти</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Зв'язок</a>
            </li>
          </ul>
          <form class="d-flex">
            <input class="form-control me-2 bg-dark text-white border-0" type="search" placeholder="Пошук" aria-label="Search">
            <button class="btn btn-outline-light bg-danger border-0" type="submit"><i class="fa fa-search"></i></button>
          </form>
        </div>
    </div>
</nav>
    </div>
    <router-view/>
  </div>
</template>

<script>
// @ is an alias to /src
import nav from '@/components/App.vue'

export default {
  name: '',
  components: {
    Nav
  }
}
</script>

<style scoped lang="scss">
nav {
  background-color: #242F39;
  .navbar-brand {
```

```
    font-family: Arial, Helvetica, sans-serif;
    text-transform: uppercase;
    font-size: 20px;
    background-color: #0000d6;
    background-image: linear-gradient(#0000d6, #ffda00);
    padding: 15px;
    letter-spacing: -1px;
    font-weight: 600;
    color: #ffffff;
  }
  li {
    a {
      &:hover {
        color: #B22222 !important;
      }
    }
  }
}
</style>
```

About.vue

```
<template>
  <div class="footer">
    <footer>
    <div class="main-content">
      <div class="left box">
        <h2>Про нас</h2>
        <div class="content">
          <p>Сайт створений для... </p>
          <div class="social">
            <a href="#"><span class="fa fa-telegram"></span></a>
            <a href="#"><span class="fa fa-github"></span></a>
            <a href="#"><span class="fa fa-facebook"></span></a>
            <a href="#"><span class="fa fa-codepen"></span></a>
          </div>
        </div>
      </div>
      <div class="center box">
        <h2>Контакти</h2>
        <div class="content">
          <div class="place">
            <span class="fa fa-map-marker"></span>
            <span class="text">Дніпро, Україна</span>
          </div>
          <div class="phone">
            <span class="fa fa-phone"></span>
            <span class="text">+089-765432100</span>
          </div>
          <div class="email">
            <span class="fa fa-envelope"></span>
            <span class="text">abc@gmail.com</span>
          </div>
```

```
      </div>
    </div>
    <div class="right box">
     <h2>Зв'язок</h2>
     <div class="content">
      <form action="#">
       <div class="email">
        <div class="text">Email *</div>
        <input type="email" required>
       </div>
       <div class="msg">
        <div class="text">Повідомлення *</div>
        <textarea rows="2" cols="25" required></textarea>
       </div>
       <div class="btn px-0 w-100">
        <button type="submit">Відправити</button>
       </div>
      </form>
     </div>
    </div>
   </div>
   <div class="bottom">
    <center>
     <span class="fa fa-copyright"></span><span> 2021 Всі права захищені .</span>
    </center>
   </div>
  </footer>
  </div>
</template>
<style scoped lang="scss">
footer {
  position: relative;
  width: 100%;
  background: #242f39;
  color: #ffffff;
}
.main-content {
  display: flex;
  .box {
    flex-basis: 50%;
    padding: 10px 20px;
  }
}
.box {
  h2 {
    font-size: 1.125rem;
    font-weight: 600;
    text-transform: uppercase;
  }
  .content {
    margin: 20px 0 0 0;
    position: relative;
```

```
    &:before {
      position: absolute;
      content: '';
      top: -10px;
      height: 2px;
      width: 100%;
      background: #24252f;
    }
    &:after {
      position: absolute;
      content: '';
      height: 2px;
      width: 15%;
      background: #B22222;
      top: -10px;
    }
  }
}
.left {
  .content {
    p {
      text-align: justify;
    }
    .social {
      margin: 20px 0 0 0;
      a {
        padding: 0 2px;
        color: #ffffff;
        span {
          height: 40px;
          width: 40px;
          background: #24252f;
          line-height: 40px;
          text-align: center;
          font-size: 18px;
          border-radius: 5px;
          transition: 0.3s;
          &:hover {
            background: #f12020;
          }
        }
      }
    }
  }
}
.center {
  .content {
    .fas {
      font-size: 1.4375rem;
      background: #242f39;
      height: 45px;
      width: 45px;
```

```
      line-height: 45px;
      text-align: center;
      border-radius: 50%;
      transition: 0.3s;
      cursor: pointer;
      &:hover {
        background: #b22222;
      }
    }
    .text {
      font-size: 1.0625rem;
      font-weight: 500;
      padding-left: 10px;
    }
    .phone {
      margin: 15px 0;
    }
  }
}
.right {
  form {
    .text {
      font-size: 1.0625rem;
      margin-bottom: 2px;
      color: #656565;
    }
    .msg {
      margin-top: 10px;
    }
    input {
      width: 100%;
      font-size: 1.0625rem;
      background: #24252f;
      padding-left: 10px;
      border: 1px solid #222222;
      color: #ffffff;
      height: 35px;
      color: #ffffff;
      &:focus {
        outline-color: #3498db;
      }
    }
    textarea {
      width: 100%;
      font-size: 1.0625rem;
      background: #24252f;
      padding-left: 10px;
      border: 1px solid #222222;
      color: #ffffff;
&:focus {
        outline-color: #3498db;
      }
```

```
    }
    .btn {
      margin-top: 10px;
      button {
        height: 40px;
        width: 100%;
        border: none;
        outline: none;
        background: #b22222;
        font-size: 1.0625rem;
        font-weight: 500;
        cursor: pointer;
        color: #ffffff;
        transition: .3s;
        &:hover {
          background: #b22222;
        }
      }
    }
  }
}
.bottom {
  center {
    padding: 5px;
    font-size: 0.9375rem;
    background: #24252f;
    span {
      color: #656565;
    }
    a {
      color: #f12020;
      text-decoration: none;
      &:hover {
        text-decoration: underline;
      }
    }
  }
}
</style>
```

Style.scss

```
* {
  margin: 0;
  padding: 0;
  outline: none;
  box-sizing: border-box;
}
body {
  background-color: #24252F !important;
}

@media (max-width: 1245px) {
```

```
  nav {
    padding: 0 5%;
  }
}
@media (max-width: 1140px) {
  nav {
    padding: 0px;
    .logo {
      flex: 2;
      text-align: center;
    }
    .nav-links {
      position: fixed;
      z-index: 99;
      top: 70px;
      width: 100%;
      left: -100%;
      height: 100%;
      display: inline-block;
      padding: 10px 50px 0 50px;
      text-align: center;
      transition: left 0.3s ease;
      li {
        line-height: 40px;
        margin: 20px 0;
        a {
          font-size: 20px;
        }
      }
    }
    .nav-links.active {
      left: 0%;
    }
    form {
      position: absolute;
      top: 65px;
      right: 55px;
      background-color: #24252F;
      opacity: 0;
      pointer-events: none;
      transition: top 0.3s ease, opacity 0.1s ease;
      &::before {
        position: absolute;
        content: "";
        top: -13px;
        right: 0;
        width: 0;
        height: 0;
        z-index: -1;
        margin: -20px 0 0;
        border: 10px solid transparent;
        border-bottom-color: #242F39;
```

```
      }
      &::after {
        position: absolute;
        content: "";
        height: 50px;
        width: calc(100% + 20px);
        padding: 2px;
        background-color: #242F39;
        border-radius: 2px;
        z-index: -2;
        left: 50%;
        top: 50%;
        transform: translate(-50%, -50%);
      }
    }
    form.active {
      opacity: 1;
      top: 80px;
      pointer-events: auto;
    }
    .menu-icon {
      display: block;
      span {
        display: block;
      }
      span.hide {
        display: none;
      }
    }
    .search-icon {
      display: block;
    }
    .search-icon.hide {
      display: none;
    }
    .cancel-icon.show {
      display: block;
    }
  }
}
@media screen and (max-width: 900px) {
  footer {
    position: relative;
    bottom: 0px;
  }
  .main-content {
    flex-wrap: wrap;
    flex-direction: column;
    .box {
      margin: 5px 0;
    }
  }
```

```
}
@media (max-width:350px) {
  nav {
    .search-icon {
      margin: 0 10px;
      font-size: 16px;
    }
    .cancel-icon {
      margin: 0 10px;
      font-size: 16px;
    }
    .menu-icon {
      margin: 0 10px;
      font-size: 16px;
    }
    form {
      right: 30px;
    }
  }
}
```

index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    component: () => import(/* webpackChunkName: "about" */ '../views/About.vue')
  }
]

const router = new VueRouter({
  routes
})

export default router
```

button.js

```
const menuBtn = document.querySelector(".menu-icon span");
const searchBtn = document.querySelector(".search-icon");
const cancelBtn = document.querySelector(".cancel-icon");
const links = document.querySelector(".nav-links");
```

```
const form = document.querySelector("form");
menuBtn.onclick = () => {
    links.classList.add("active");
    menuBtn.classList.add("hide");
    searchBtn.classList.add("hide");
    cancelBtn.classList.add("show");
}


cancelBtn.onclick = () => {
    links.classList.remove("active");
    menuBtn.classList.remove("hide");
    searchBtn.classList.remove("hide");
    cancelBtn.classList.remove("show");
    form.classList.remove("active");
    cancelBtn.style.color = "#B22222";
}

searchBtn.onclick = () => {
    form.classList.add("active");
    searchBtn.classList.add("hide");
    cancelBtn.classList.add("show");
}
```

charts.js

```
(async function () {

  const { from90_to21, Oblasti, currentpop, currentmonth, vkudiasp } = (await axios.get('/data.json')).data

  new Chart("from90_to21", {
    type: "bar",
    data: {
      labels: Object.keys(from90_to21),
      datasets: [{
        fill: false,
        lineTension: 0,
        backgroundColor: "rgba(72, 169, 213)",
        borderColor: "rgba(0,0,255,0.1)",
        data: Object.values(from90_to21)
      }]
    },
    options: {
      legend: {display: false},
      scales: {
        yAxes: [{ticks: {min: 35, max:55}}],
      }
    }
  })

  const randomColor = () => {
    const r = Math.floor(Math.random() * 255)
```

```
  const g = Math.floor(Math.random() * 255)
  const b = Math.floor(Math.random() * 255)

  return "rgb(" + r + "," + g + "," + b + ")"
 }

 new Chart("Oblasti", {
  type: "outlabeledPie",
  data: {
   labels: Object.keys(Oblasti),
   datasets: [
    {
     data: Object.values(Oblasti),
     backgroundColor: Array.from({length: Object.values(Oblasti).length}).map((a) => randomColor()),
    }
   ]
  },
  options: {
   // responsive: true,
   legend: {
    display: false
   }
  },
  plugins: {
   legend: false,
   outlabels: {
    text: '%l %p',
    color: 'white',
    stretch: 45,
    font: {
     resizable: true,
     minSize: 12,
     maxSize: 18
    }
   }
  }
 })

 $('#currentpop').html(currentpop)
 $('#currentmonth').html(currentmonth)
 $('#vkudiasp').html(vkudiasp)
})()
```

package.json

```
{
 "name": "ukrnas",
 "version": "0.1.0",
 "private": true,
 "scripts": {
  "serve": "vue-cli-service serve",
  "build": "vue-cli-service build"
 },
```

```
  "dependencies": {
    "core-js": "^3.6.5",
    "vue": "^2.6.11",
    "vue-router": "^3.2.0",
    "vuex": "^3.4.0"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~4.5.0",
    "@vue/cli-plugin-router": "~4.5.0",
    "@vue/cli-plugin-vuex": "~4.5.0",
    "@vue/cli-service": "~4.5.0",
    "node-sass": "^4.12.0",
    "sass-loader": "^8.0.2",
    "vue-template-compiler": "^2.6.11"
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead"
  ]
}
```