МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Система SPA на фрейморку Vue.js з використанням firebase»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Проценко О.Б.**

Студента групи ІН – 63 **Коваленко О.В.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедри Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-63спеціальності "Інформатика" денної форминавчання Коваленко Олександра Вячеславовича.

Тема: " Система SPA на фрейморку Vue.js з використанням firebase "

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд доступних методів розробки; 2) постановка задачі й налаштування робочої середи; 3) огляд методів та технологій створення SPA на фрейморку Vue.js 4) розробка програмного продукту та налаштування бази даних 5) перевірка кінцевого результату.

Дата видачі завдання "_____" _____2020р.

Керівник випускної роботи _____Проценко О.Б.

Завдання прийняв до виконання _____Коваленко О.В.

РЕФЕРАТ

Записка:. 70 стор., 26 рис., 1 додаток, 15 джерел.

Об’єкт дослідження — SPA інтернет магазин.

Мета роботи — розробка односторінкового додатоку на фреймворку Vue.js з використанням хмарної бази даних firebase.

Методи дослідження — аналіз схожих веб додатків.

Результати — створено мініатюрний інтернет магазин на якому зареєстрований користувач може придбати або поставити на продажу будь-який товар. SPA було реалізовано на популярному фреймворку Vue.js. При цьому сайт не містить зайвої інформації тому є легким у використанні та швидким у завантаженні.

СИСТЕМА SPA НА ФРЕЙМОРКУ VUE.JS З ВИКОРИСТАННЯМ FIREBASE

SPA SYSTEM ON VUE.JS FRAMEWORKS WITH USING FIREBASE

ЗМІСТ

ВСТУП

Робота присвячена темі розробки Single Page Application на фрейморку vue.js з використанням бібліотек розроблених під нього.

Single Page Application - скорочено SPA, в перекладі на українську мову означає "Односторінковий додаток". Іншими словами SPA - це web-додаток, розміщений на одній web-сторінці, яка для забезпечення роботи завантажує весь необхідний код разом із завантаженням самої сторінки. Додаток такого типу з'явилися порівняно недавно, з початком ери SPA є типовим представником додатків на HTML5.

Фреймворк -інфраструктура програмних рішень, что полегшує розробка складних систем. Дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду.

Для того щоб почати етап розробки сайту потрібно розібратись з основними можливостями фреймворку та дізнатись в чому він кращій або гірший в порівнянні з іншими популярними рішеннями.

1. ОГЛЯД ВІДОМИХ РІШЕНЬ

1.1 Популярні фреймворки

Вибір правильного фреймворка є одним з ключових моментів, що впливають на якість майбутнього сайту. Маже кожну з описаних технологій можна назвати повною в усіх відношеннях, так як їх використовують, як для frontend, так і для backend розробки.

На даний момент часу можна виділити п'ять головних frontend-фреймворків для веб-розробки. Це: Angular, React, Vue, Ember та Backbone. Більшість з них є повноцінними в усіх відношеннях, бо їх можна використовують, як для backend, так і для frontend розробки.[6]

Backbone та Ember не зовсім не підходять для розробки тому їх розберемо не так детально як інші фреймворки, зазначені нижче переваги та недоліки дають зрозуміти чому ці рішення не можна використовувати в даному проекті[2].

Переваги і недоліки при їх використанні:

Ember:

- Простий в налаштуванні
- Розгортає великі інтерфейси
- Двостороння прив'язка даних
- Жорстка структура проектів
- Немає стандартного набору UI-елементів

Backbone:

- Бібліотека досить потужна і зручна,
- Легко розширюється
- Для використаня Backbone потрібен jQuery
- Це не фреймворк, це бібліотека

Тепер, коли вибір йде між трьома фреймворками є можливість роздивитись їх більш детально і підібрати той, який задовільнить наші потреби, тобто буде простим у використанні і матиме високу швидкість завантаження сторінки.

Першим, краще за все, буде роздивитись фреймворк react, бо він має найбільшу популярність, рисунок 1.1, серед розробників веб-додатків.

React

- Девіз React: «Вивчи один раз - пиши всюди»
- Free and Open Source
- Має можливість використовувати вже написаний код
- Має підтримку віртуальньної функціональності DOM
- Алгоритм Virtual DOM є доволі повільним і неточний
- Потрібно використовувати складне асинхронне програмування підчас спілкуванні з сервером

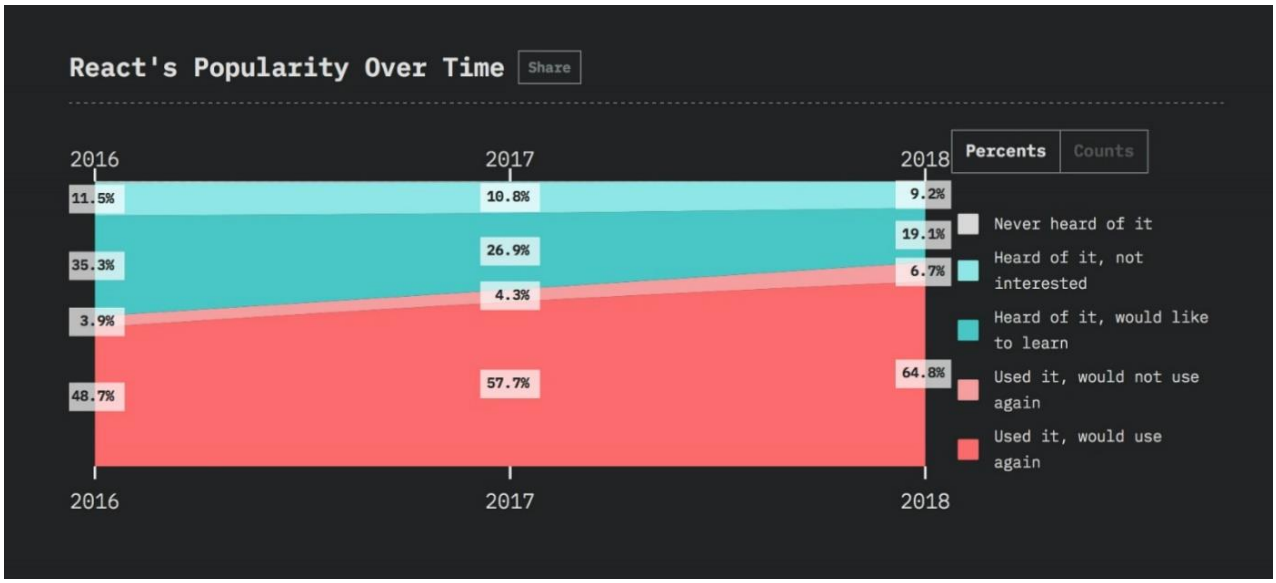

Рисунок 1.1 – Графік популярності React

React (більше 110 тисяч зірок на GitHub) - це ефективна і гнучка декларативна бібліотека JavaScript для збірки UI від команди Facebook. Вона дозволяє без зусиль створювати інтерактивний користувальницький інтерфейс.[15]

React не просто підтримує збірку об'єктно-центричних додатків - він її заохочує. Крім того, творці фреймворка з усією серйозністю поставилися до зворотної сумісності, так що в довговічності свого застосування можете бути впевнені. За графіком вище добре видно, що за останні кілька років рівень

обізнаності про React значно підвищився. Ось чому бібліотека стане чудовою відправною точкою вашої подорожі у фронтенд-розробку.

Можна створити новий додаток на React, використовуючи призначений для цього тулчейн create-react-app, на даний момент найпопулярніший з існуючих. Щоб почати працювати з ним, запустіть наступні рядки в командному рядку папки проекту:

```
npm create-react-app my-app
```

```
cd my-app
```

```
npm start
```

Переваги та недолікі використання фреймворку Angular:

- Відкритий вихідний код
- Збереження фрагментів коду для подальшого використання
- Менша кількіспm помилок, бо є прив'язка даних яка будується на базі Angular-елементів
- Підтримуються різні елементи MVC
- Добре працює в середовищі Agile
- Маса інструментів для тестів
- Angular доволі складний для початківців
- API Angular дуже велика, тому потрібно розібратися з багатьма концепціями

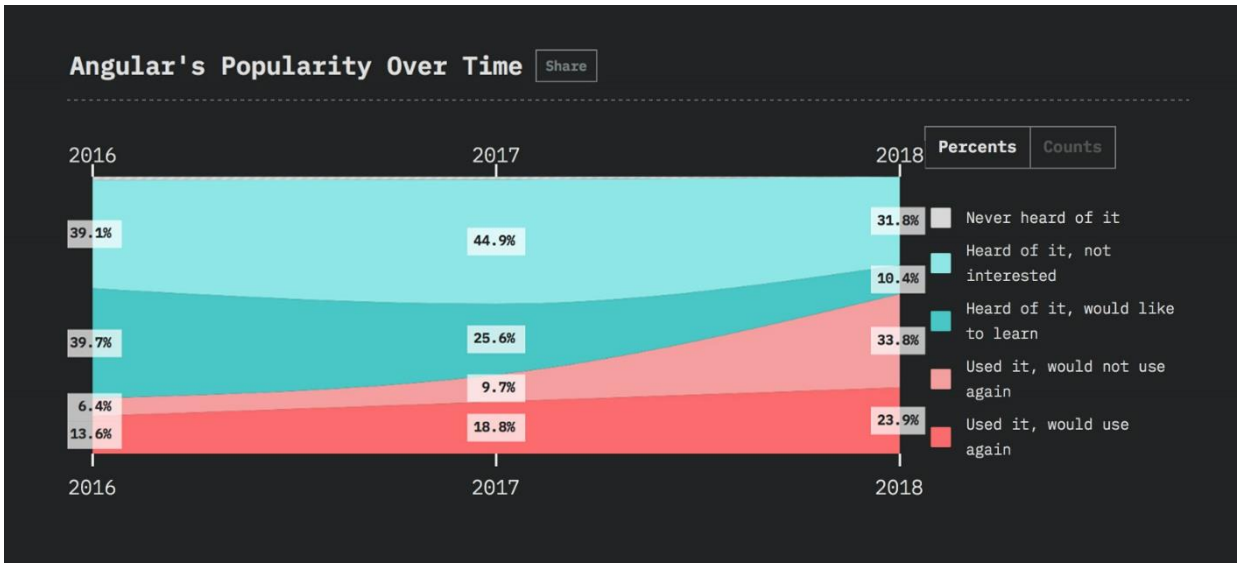

Рисунок 1.2 – Графік популярності Angular

Angular - фреймворк від компанії Google, який отримав майже 44 тисячі зірок на GitHub. Являє собою платформу, яка спрощує збірку додатків в інтернет. У Angular поєднуються декларативні шаблони, впровадження залежності, двостороннє зв'язування даних і кращі практики вирішення проблем розробки. Ця платформа дозволяє збирати додатки для веб, мобільних пристроїв і настільних ПК. У ній пропонується самий зручний і зрозумілий для початківців інтерфейс командного рядка (CLI) і навіть консоль (Console) - клієнт з графічним інтерфейсом.[13]

Дивлячись на графік на рисунку 1.2, можна помітити величезну різницю в порівнянні з графіками описаних вище фреймворків. Справа в необ'єктивності даних, що виникла через формулювання питань, яка виключає AngularJS - незалежний від Angular фреймворк. Ось чому цю візуалізацію краще не брати до уваги. Для установки CLI за допомогою npm відкрийте вікно командного рядка або консолі і введіть наступну команду:

```
npm install -g @angular/cli
```

1.2 Постановка завдання

Завданням роботи є :

- дослідження популярних фреймворків для розробки SPA і вибір потрібного для розробки ;
- огляд популярних інтернет магазинів та створення прототипу проекту який відрізняється тим, що не матиме зайвої інформації про товар і буде мати високу швидкість звавантаження;
- розробка frontend частини веб додатку, підключення потрібних бібліотек та верстка компонентів. Проект має мати сторінки для реєстрації та входу, також необхідно розробити місце для створення або редагування картки товару. Продавець повинен бачити всі його замовлення ;
- розробка backend частини веб додатку, підключення firebase та прив'язка бази даних до проекту;
- після завершення розробки необхідно перевірити додаток на баги та виправити їх.

2. ВИБІР МЕТОДУ РІШЕННЯ

2.1 Основні переваги та недоліки мови розробки на Vue

Теоретично vue є альтернативою jQuery. Але в реальності він доволі успішно конкурує з React.JS – очевидним лідером у сфері view. Також на ньому можна робити те саме що і на Angular або Ember.

Усі фреймворки можна привести під один і той самий мінус – складність освоєння. Але vue є простим як у вивченні так і у використанні. Окрім цього фреймворк має одну з найкращих офіційних документацій, що ще більше полегшує його вивчення[3].

Під час розробки Vue.JS брали кращі практики з перерахованих технологій З React.JS команда Vue запозичила ідею віртуального DOM. Цей підхід виключає пряму взаємодію з вузлами інтерфейсу. Початкова робота ведеться з його virtual DOM. І тільки після цього зміни застосовуються до реальних вузлів інтерфейсу. Паралельно відбувається порівняння реального DOM дерева і його віртуальної копії. Таким чином виявляється різниця і перемальовується тільки те, що зазнало змін [9].

З Angular Vue.JS запозичив two-way data binding. Це дозволяє проектувати інтерфейси: по-перше, декларативно; по-друге, з використанням Vue в шаблонизатор. Таких як Haml або Pug. Правда, відзначимо, що такий підхід практикувався і раніше. Наприклад, у фреймворку Knockout.JS.

Ядро Vue.JS, подібно React, містить лише необхідний функціонал для роботи з інтерфейсом. Тому воно компактно, легко інтегрується з іншими технологіями, в тому числі з jQuery і навіть може використовуватися замість нього (для розробки простих інтерфейсів).

Крім того, Vue доступний ряд модулів, що реалізують сучасний підхід до розробки веб-додатків. Наприклад, практично всі React додатки проектуються в тандемі з технологією контролю станів Redux, яка є окремою бібліотекою і реалізує flux-архітектуру. Підхід, що практикується бібліотекою Redux виявився досить зручним і успішним. Тому для Vue.JS була розроблена своя

технологія контролю стану програми - Vuex. Він повністю запозичує ідеї Redux, але ступінь інтеграції цієї бібліотеки з Vue набагато вище, ніж в разі React і Redux. А це трансформується в швидкодію і зручність[10].

Не варто забувати і про те, що успішність технології прямо залежить від активності її користувачів. Тобто - спільноти розробників. Так React, будучи, мабуть, найпопулярнішою технологією, має величезну кількість розширень основного функціоналу. Вони створюються учасниками спільноти, і у React розробників є інструментарій на всі випадки життя. У Vue.JS в цьому плані справи трішки гірші, рисунок 2.1. У всякому разі - поки що.

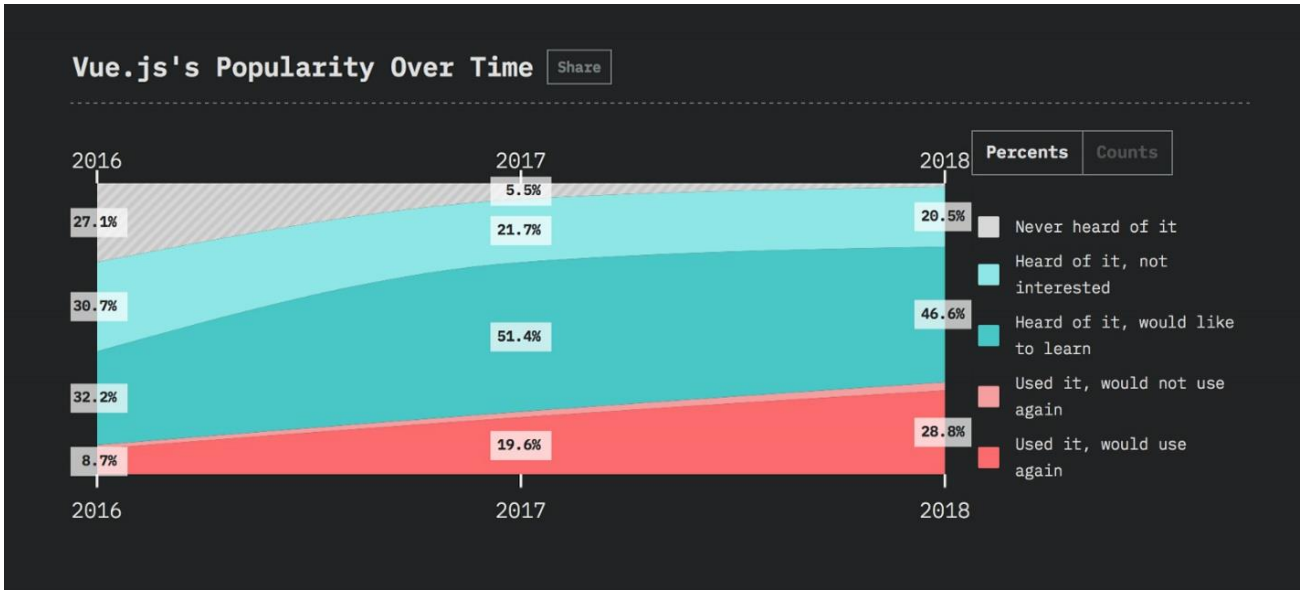

Рисунок 2.1 – Графік популярності Vue

Vue.js, прогресивний фреймворк який використовують для збірки користувацьких інтерфейсів, створений Еваном Ю і ще двома стами тридцяти чотирма ентузіастами, набрав більше ста двадцяти однієї тисячі зірок на GitHub. Він включає доступну кореневу бібліотеку, яка в першу чергу вирішує завдання рівня уявлення, і екосистему додаткових бібліотек, що дозволяє створювати складні і об'ємні односторінкові додатки (Single-Page Applications).[1]

Як видно на графіку, Vue вдалося перестрибнути маркетингову прірву: про нього чув майже кожен розробник. Можна припустити, що це сталося завдяки величезним зусиллям, які прикладали Еван і його команда з 2017 року,

відвідуючи різноманітні збори та конференції, а також організовуючи власні. Проте, пробіл у знаннях розробників все ще існує, і щоб його закрити, в 2019 році необхідно створити більше навчальних матеріалів по роботі з Vue.js. Встановіть цей фреймворк за допомогою npm:

```
npm install vue
```

Переваги та недоліки феймворку Vue

- За замовчуванням не потребує будь-яких компіляторів для початку роботи.
- Трансформація з бібліотеки до фреймворка в процесі використання
- Управління просунутими односторінкового додатками
- Гарний баланс між ремонтопридатністю і написання самого коду, а також відмінна читаність
- Виводить Runtime помилки прямо в шаблонах

2.2 Огляд розробки на вибраному фреймворку

Для того щоб запустити тестовий проект нам потрібно створити html файл. Після цього треба підключити сам фреймворк, це можливо зробити різними способами. Для початку використаємо спосіб підключення через скрипт.

<! - версія для розробки, відображає корисні попередження в консолі ->

<Script src = "https://cdn.jsdelivr.net/npm/vue/dist/vue.js"> </ script>

або:

<! - production-версія, оптимізована для розміру і швидкості ->

<Script src = "https://cdn.jsdelivr.net/npm/vue"> </ script>

В ядрі Vue.js знаходиться система, яка дозволяє декларативно відображати дані в DOM за допомогою простих шаблонів [11].

Приклад простого Vue-додатку зображено на рисутку 2.2

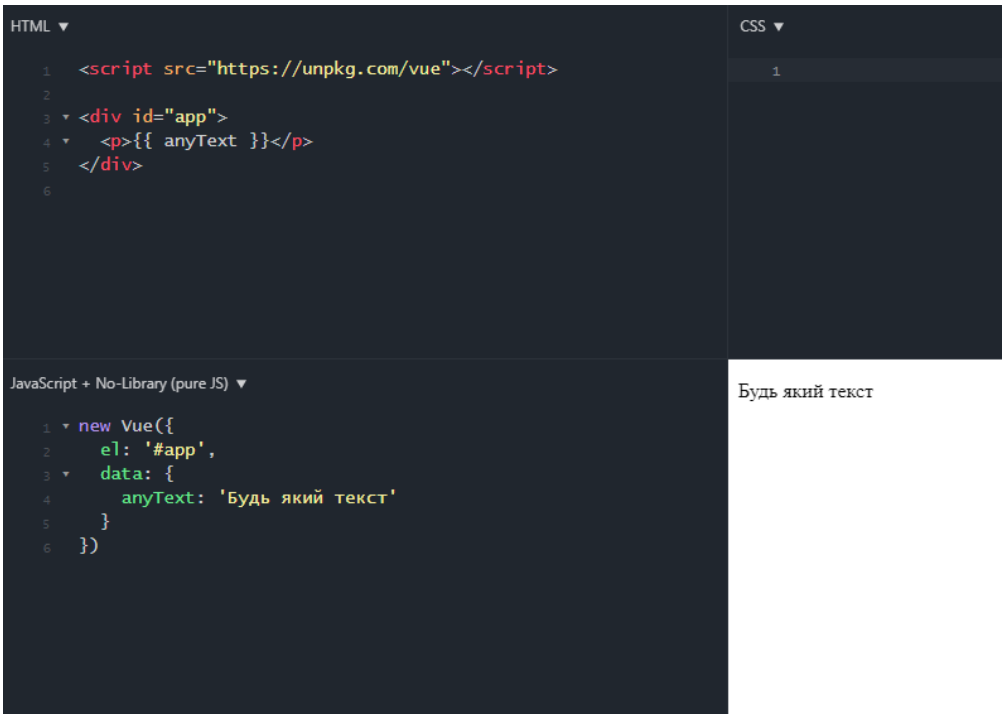

Рисунок 2.2 - Приклад простого vue коду

Для створення об'єкта додатка в Vue.js застосовується об'єкт Vue. Цей об'єкт, по-перше, визначає кореневий елемент додатки на веб-сторінці за допомогою параметра el:

Тобто кореневих елементом додатка буде елемент з id рівним app.Також об'єкт визначає використовувані дані через параметр data.

В даному випадку визначено тільки властивість message, яка зберігає рядок.

В елементі з id = app на веб-сторінці, використовуючи подвійні фігурні дужки можна вивести значення властивості message і більш того зв'язати ділянку веб-станиці з цим елементом.

Одним з головних переваг фреймворків є двостороння прив'язка. Для створення подібної прив'язки використовується директива v-model. Наприклад зображено на рисунку 2.3:

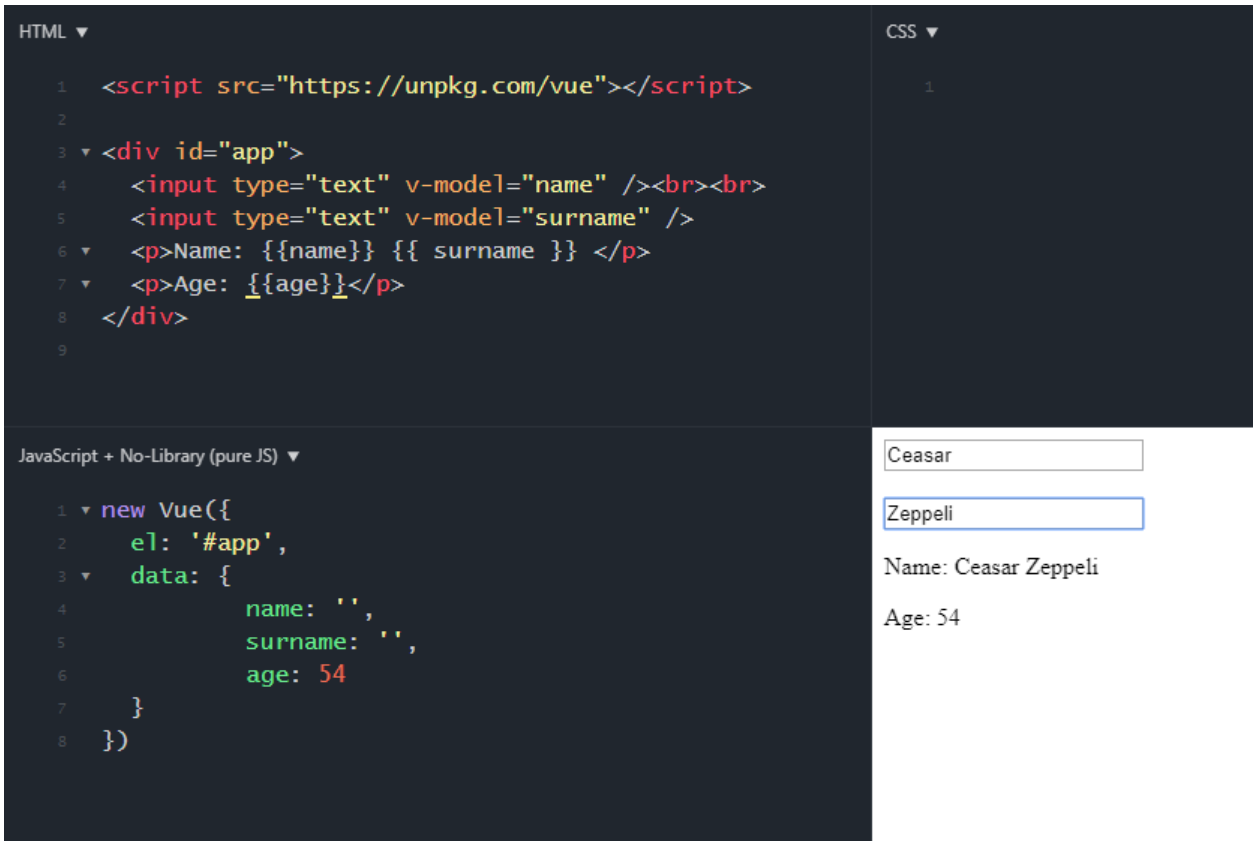

Рисунок 2.3 - Приклад двосторонньої прив'язки

Тут на сторінці визначено два текстових поля, і обидва вони прив'язані до властивостей з об'єкта Vue. У цьому випадку нам не треба додавати до текстового поля обробник події введення, і в цьому обробнику міняти вручну значення властивості name. При зміні тексту в одному з полів автоматично зміниться значення в тексті. Тобто спрацює двостороння прив'язка.[4]

2.3 Бібліотеки vue.js

2.3.1 Vue Router

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

- Вкладені маршрути / уявлення
- Модульна конфігурація маршрутизатора
- Доступ до параметрів маршруту, query, wildcards

- Анімація переходів уявлень 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 що настроює поведінку прокрутки сторінки

Створювати односторінкові додатки (SPA) використовуючи Vue + Vue Router дуже просто. За допомогою Vue.js, компонуємо свій додаток з компонентів. Додаючи Vue Router,порівнюємо наші компоненти з маршрутами і пояснюємо Vue Router де їх відображати[8]. Ось простий приклад:

.HTML

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>

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

використовуємо компонент router-link для навігації

вхідний параметр `to` визначає URL для переходу

`<router-link>` за замовчуванням відображається тегом `<a>`

```
  <router-link to="/foo">Перейти до Foo</router-link>
  <router-link to="/bar">Перейти до Bar</router-link>
 </p>
```

відображаємо тут компонент, для якого збігається маршрут

```
 <router-view></router-view>
</div>
```

.JS

```
// 0. Якщо використовуємо модульну систему (наприклад через vue-cli),
// імпортуємо Vue і VueRouter і потім викликаємо `Vue.use(VueRouter)`.
```

1. Визначаємо компоненти для маршрутів.

Вони можуть бути імпортовані з інших файлів

```
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }
```

2. Визначаємо кілька маршрутів

Кожен маршрут повинен вказувати на компонент.

"Компонентом" може бути як конструктор компонента, створений через `Vue.extend()`, так і просто об'єкт з опціями компонента.

Ми поговоримо про вкладені маршрутах пізніше.

```
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]
```

3. Створюємо примірник маршрутизатора і передаємо маршрути в опції `routes`

Моєжна передавати і додаткові опції, але поки не будемо ускладнювати.

```
const router = new VueRouter({
  routes // скорочений запис для `routes: routes`
})
```

4. Створюємо і монтуємо кореневої екземпляр додатку.

Переконайтеся, що передали екземпляр маршрутизатора в опції `router`, щоб дозволити додатком знати про його наявність.

```
const app = new Vue({
  router
}).$mount('#app')
```

Готово, додаток працює.

2.3.2 Vuex

Vuex використовує єдине дерево стану - коли один об'єкт містить все глобальне 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 налагодження.

У центрі будь-якого Vuex-додатку знаходиться сховище. «Сховище» - це контейнер, в якому зберігається стан вашого застосування. Два моменти відрізняють сховище Vuex від простого глобального об'єкта:

- Сховище Vuex реактивно. Коли компоненти Vue покладаються на його стан, то вони будуть реактивно і ефективно оновлюватися, якщо стан сховища змінюється.
- Не можна безпосередньо змінювати стан сховища. Єдиний спосіб внести зміни - явно викликати мутацію. Це гарантує, що будь-яка зміна стану залишає слід і дозволяє використовувати інструментарій, щоб краще розуміти хід роботи програми.

Після установки Vuex, можна створювати сховище[5]. Це досить просто - необхідно вказати початковий об'єкт стану і деякі мутації:

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

Тепер можна отримати доступ до об'єкта стану через store.state і викликати зміну стану за допомогою методу store.commit:

```
store.commit('increment')


console.log(store.state.count) // -> 1
```

Причина, по якій викликано мутацію, замість зміни store.state.count безпосередньо, в тому, що потрыбно явно відстежувати її. Це просте угоду робить наш намір більш явним, що спрощує розуміння змін, що відбуваються стану програми при читанні коду. Крім того, це дозволяє використовувати інструменти для відстеження кожної мутації, створення знімків стану або навіть використання «машини часу» для налагодження.

Використання стану сховища в компоненті передбачає просто повернення необхідної частини стану в обчислюваному властивості, оскільки стан сховища реактивно. Ініціювання змін - це просто запуск мутацій в методах компонентів.

2.3.3 Vuetify

Vuetify - це фреймворк для розробки Vue.js додатків з використанням Material Design. Корисна річ для тих, хто використовує дизайн гугла. У фреймворка великий набір різних компонентів, гарна екосистема і активна спільнота на гітхабі[7].

Vuetify також можна встановити за допомогою Vue UI, нової візуальної програми для @ vue / cli. Переконайтеся, що у вас встановлена остання версія Vue CLI, а потім нпишіть в вашого терміналі команду:

```
$ vue ui
```

Це запустить Vue UI і відкриє нове вікно в вашому браузері. Зліва, натисніть Plugins. Потім, введіть Vuetify в рядок пошуку як показно на рисунку 2.4.

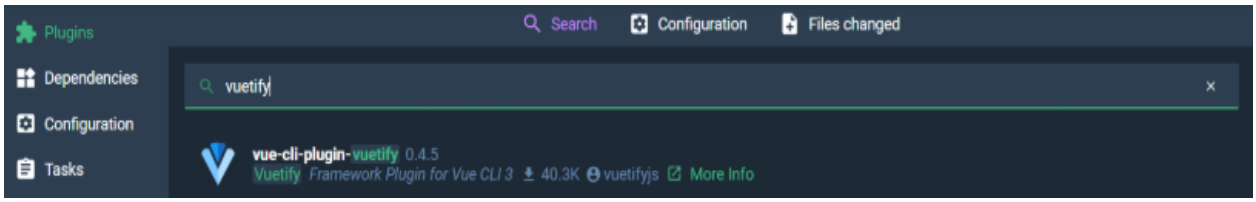

Рисунок 2.4 - Установка плагіна Vuetify

Після установки, буде можливо налаштувати опції Vuetify. Приклад можна побачити на на рисунку 2.5.

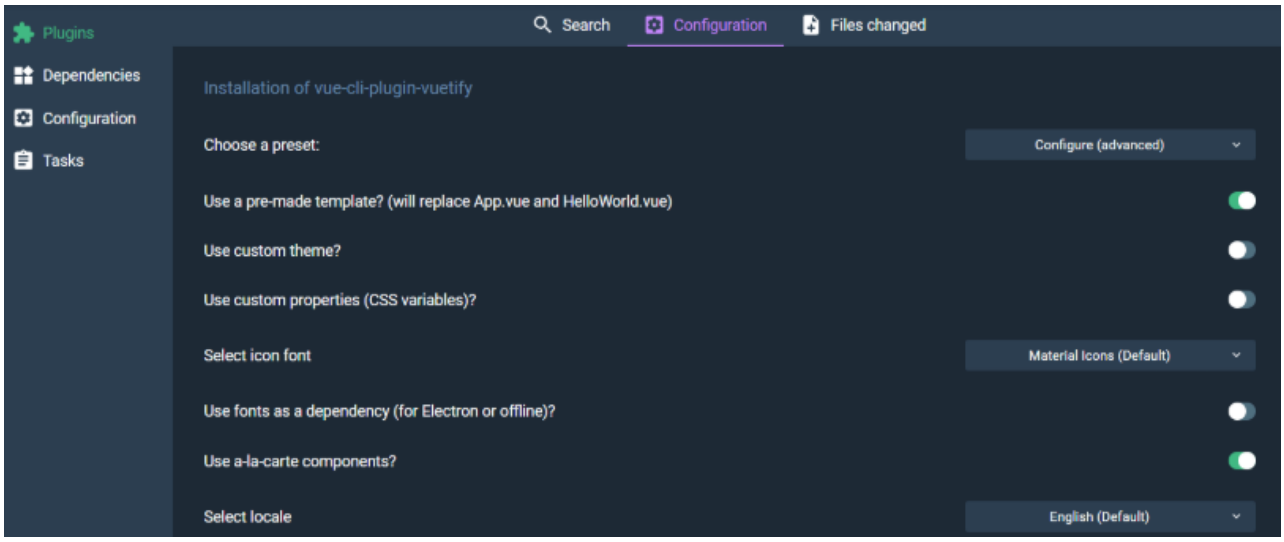

Рисунок 2.5 - Налаштування плагіна Vuetify

Щоб включити Vuetify в існуючий проект, потрібно підключити його в свої node_modules. Для цього можна використовувати npm або yarn. Це обидва менеджера пакетів, які дозволяють вам контролювати, які ресурси доступні для вашої програми. Якщо один з них вже встановлені на вашому ПК то для запуску потрібно консолі написати:

```
$ yarn add vuetify
// або
$ npm install vuetify –save
```

2.4 Вибір середовища розробки

Visual Studio Code - редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS

Позиціонується як «легкий» редактор коду для кроссплатформенной розробки веб-і хмарних додатків.

VS Code дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ.

У редакторі присутні вбудований відладчик, інструменти для роботи з Git і засоби рефакторинга, навігації по коду, автодоповнення типових конструкцій і контекстної підказки.

Продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легковажним рішення, яке дозволяє обійтися без повної інтегрованого середовища розробки. Великим плюсом редактора є підтримка великої кількості мов, таких як C ++, C #, Python, PHP, JavaScript та інших.

Переваги сервісу

- безліч налаштувань (як всієї програми, так і інтерфейсу);
- розширювана бібліотека доповнень і готових рішень;
- мультифункціональність (редактор підтримує майже всі мови, які використовуються для створення додатків);
- простота і гнучкість.

2.5 Вибір бази даних

Firebase це база даних у якій зміни відбуваються у реальному часі і зберігає отримані данні в форматі JSON.Під час змін в базі даних відбувається синхронізація між усіма користувачами або додатками, які її використовують. Отже всі оновлення інформації є моментальними.

Також у Firebase окрім сховища є і аутентифікація користувачів, при цьому вся інформація транспортується через SSL з'єднання яке є повністю захищене. Можна підібрати різні комбінації паролю та логіну для перевірки на автентичність, будь-якого сервісу[14].

Після переходу на головну сторінку console.firebase можна побачити всі доступні проекти. Її вид зображено на рисунку 2.6.

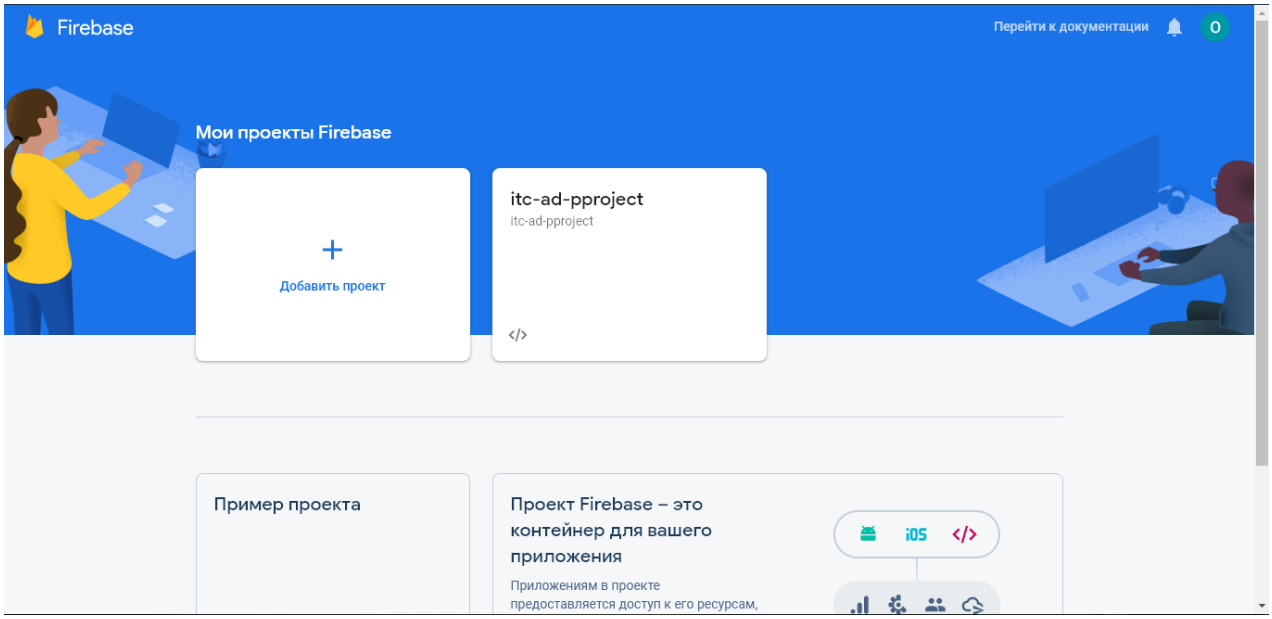

Рисунок 2.6 - Головна сторінка console.firebase.com

Для того щоб перейти до бази даних проекту необхідно натиснути на картку з назвою проекту або якщо БД ще не створена то потрібно додати новий проект.

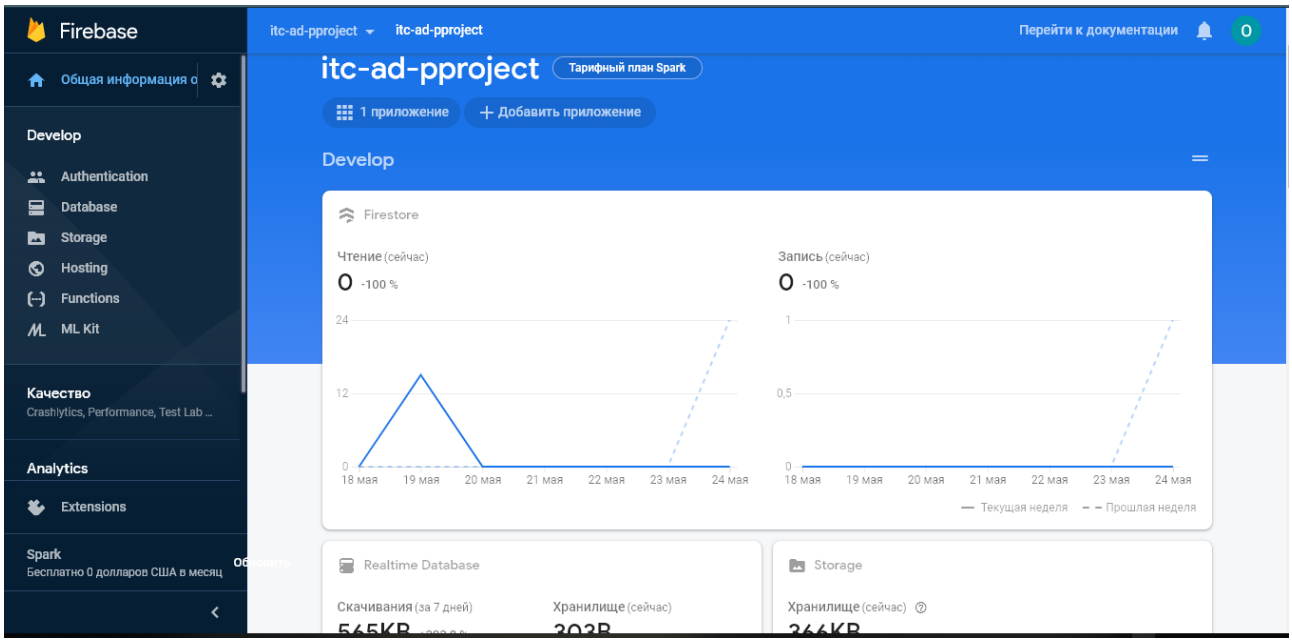

Рисунок 2.7 - Сторінка бази даних проекту

На рисунку 2.7 можна бачити графіки використання бази даних. Інформація про вагу знаходиться під графіками Також з лівої сторони є меню з усіма потрібними посиланнями

Спочатку перейдемо до сторінки аутентифікації

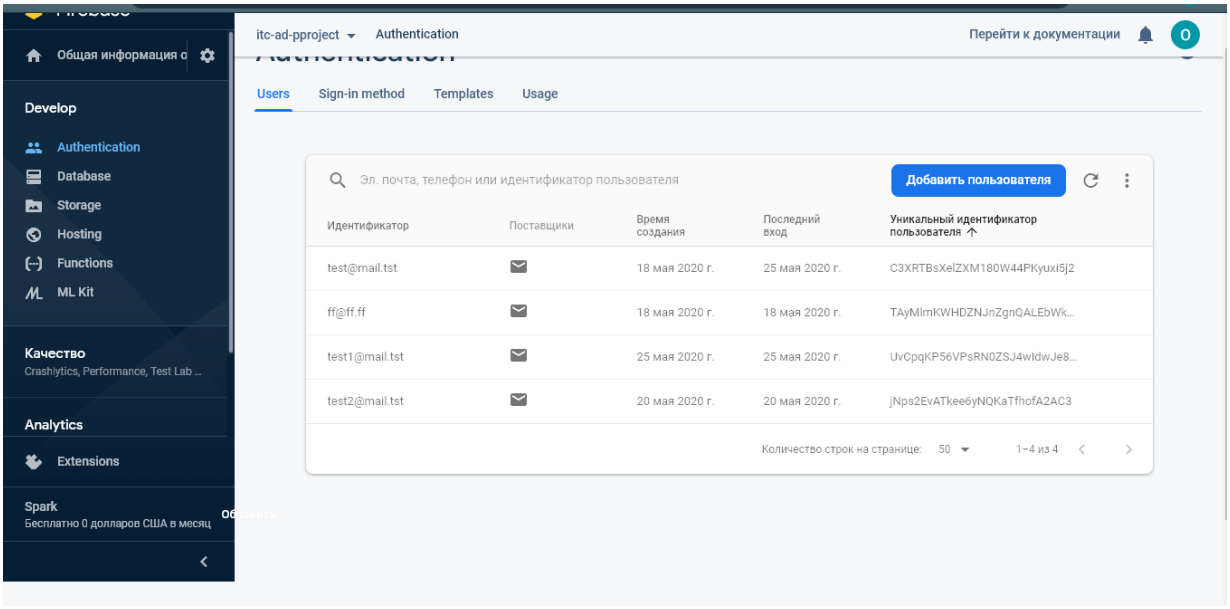

Рисунок 2.8 - Сторінка аутентифікації

На рисунку 2.8 зображено сторінку аутентифікації. Тут відображено email користувача та його унікальний ідентифікатор. Також є можливість знайти або додати нового юзера.

Для початку роботи потрібно настроїти метод аутентифікації, їх у firebase вдосталь. У даному проекті був використаний метод входу через пошту, але в майбутньому потрібно додати вхід через Google та Facebook. На наступній вкладці можна налаштувати підтвердження електронної пошти, зміну пароля та email, або створити SMS для підтвердження номера телефону.

Рисунок 2.9 - Сторінка бази даних

Перед початком роботи з базою краще настроїти правила. Для даної сторінки кращим варіантом буде надати доступ лише зареєстрованим користувачам.. Сторніка зображена на рисунку 2.9.

```
{
	"rules":
	{
		".read": "auth != null",
		".write": "auth != null"
	}
}
```

В платній версії є можливість увімкніть щоденне автоматичне резервне копіювання даних і правил безпеки.

В центрі сторінки можемо бачити дані товару з сайту. Вони потрапляють сюди після створення продукту на сторінці. Також під час редагування товару дані будуть змінюватись і в БД відповідно.

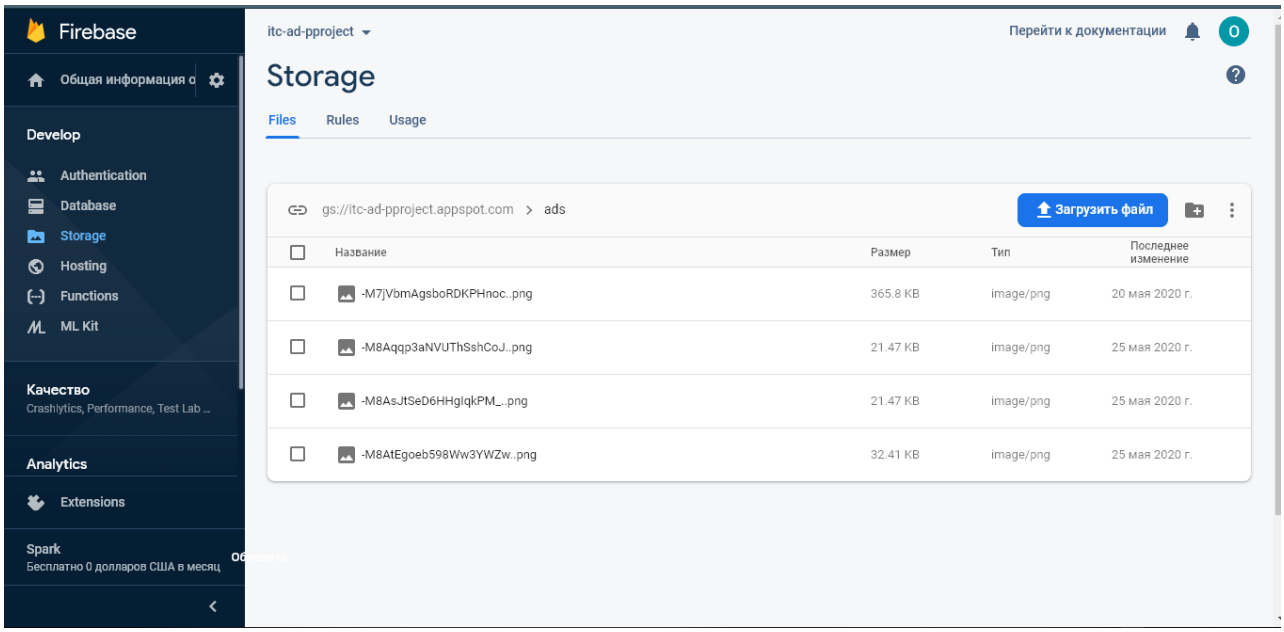

Рисунок 2.10 – Сторінка складу

Відповідно до сторінки бази даних тут також є можливість налаштувати правила запису та зчитування зображення. Кожне зображення має свою назву, розмір, тип і дату останнього редагування. Її можна побачити на рисунку 2.10.

Переваги Firebase:

- база даних в реальному часі
- система callback-ів при отриманні даних.
- Firebase є клієнтської середовищем, тому він повністю незалежний від вашого серверного коду.

Недоліки Firebase:

- його область застосування набагато менше, ніж у NoSQL-рішення;
- Firebase сильно обмежує при вибірці даних і при необхідності записати дані в декілька місць одночасно;
- Не з усіма структурами даних зручно працювати в Firebase.

Якщо потрібно зберегти велику кількість даних в базі або зареєструвати більше ніж один проект в одному акаунті, то для цього потрібно придбати

розширену версію firebase. Для даного проекту цілком вистачить функцій які йдуть в безкоштовному пакеті.

Хоча мінусів більше ніж плюсів для даного проекту є доцільним використання цієї БД бо на невеликому проекті наведені недоліки не будуть відчуватись а переваги зможуть розкритись у повному обсязі[12].

3. ПРОГРАМНА РЕАЛІЗАЦІЯ SPA

3.1 Проектування додатку

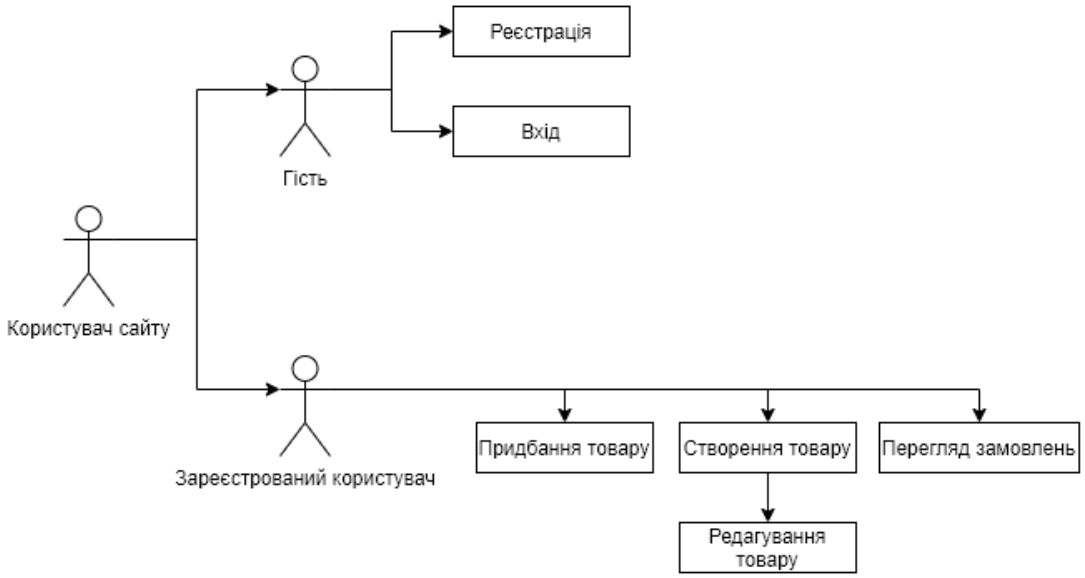

Рисунок 3.1 – UML діаграмма

На uml діаграммі, рисунок 3.1, можна бачити різні доступи для різних типів користувачів. На сайті немає адміністратора, є тільки гість або користувач. Без реєстрації людина не може дивитись, купляти або створювати товари. Усі ці функції будуть надані тільки після входу в аккаунт користувача. Також функція редагування товару доступна тільки в тому випадку, якщо даний юзер створював цей товар.

Рисунок 3.2 –DFD діаграмма

На рисунку 3.2 можна бачити діаграму потоків даних яка описує роботу сайту. Спочатку йде замовлення товару. Після цього інформація йде в обробку і передається продавцю. А він у свою чергу відправляє замовлення на адресу замовника і надсилає йому номер накладної.

3.2 Програмна реалізація

Проект є реалізацією фронт енд частини простого інтернет магазину на фреймворку Vue.js з використанням бібліотек: Vuetify, Vue Router, Vuelidate та Vuex.

Проект складається з:

- папки node_modules ,
- папки public ,
- папки src ,
- файлів json та js ,
- файлу readme.

Вся файлова структура проекту зображена на рис.3.

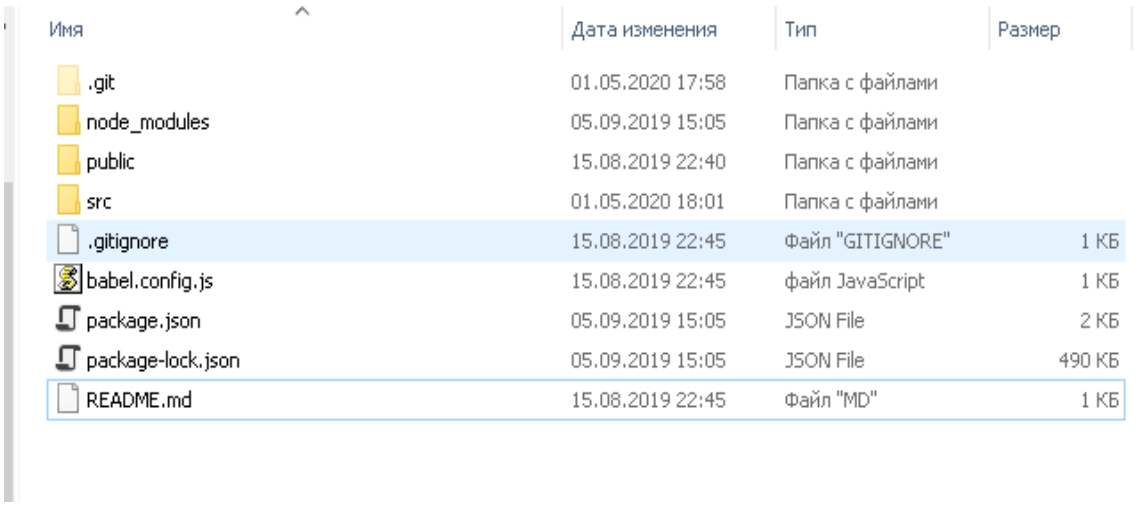

Рисунок 3.3 – Структура файлів проекту

В папці node_modules знаходяться всі модулі, вони потрібна під час розробки, тобто там знаходяться різні бібліотеки які можна використовувати в проекті.

В папці public знаходяться файл index.html - шаблон, який буде оброблятися html-webpack-plugin. На етапі складання, посилання на всі ресурси будуть впроваджуватися автоматично. Крім того, Vue CLI автоматично впроваджує підказки для ресурсів (preload / prefetch), посилання на маніфест / іконки (коли використовується PWA-плагін), і посилання на ресурси для файлів JavaScript і CSS, створених в процесі побудови.

Папка src містити власне код програми. Також вона містить всі зображення та необхідні бібліотеки. Більш детальний огляд вмісту папки буде зроблено далі

Основою проекту є файл App.vue. Зовнішній вигляд сторінки показано на рисунку 3.4. В ньому знаходиться хедер сайту та <router-view></router-view> - це точка, в якій буде відображено компонент, відповідний до маршруту верхнього рівня.

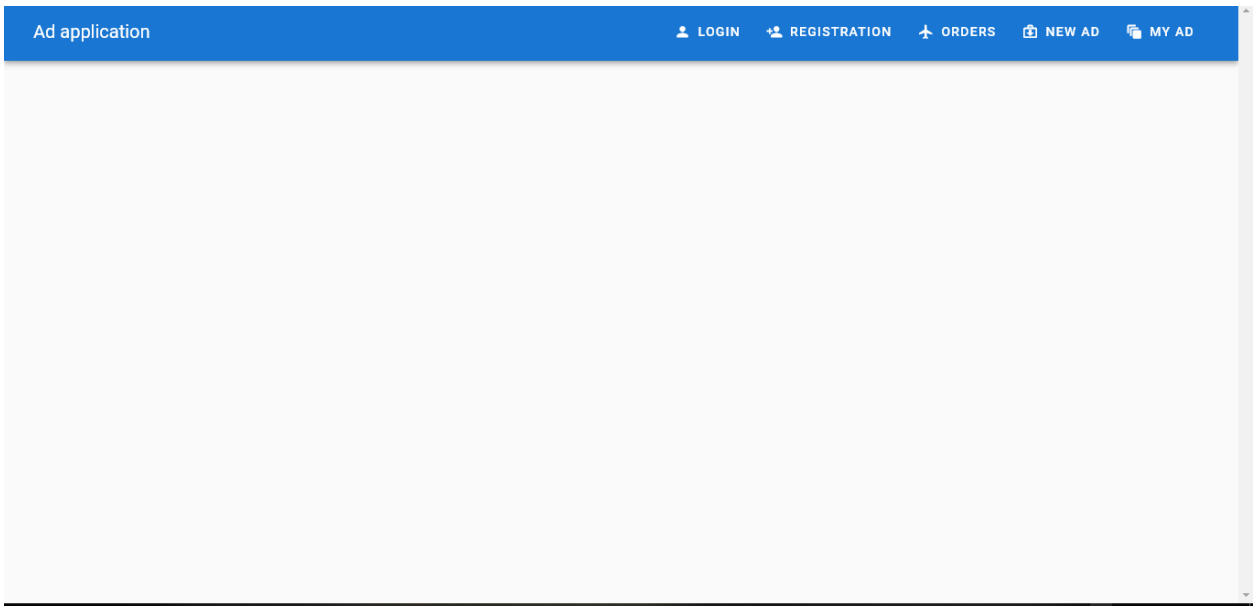

Рисунок 3.4 – Зовнішній вигляд головної сторінки без контенту

Для того щоб відобразити контент сторінки потрібно розкоментувати <!-- <router-view></router-view> -->. Для корректної роботи сайту необхідно правильно налаштувати цей файл.

Д початку необхідно імпорвувати сторінки які знаходяться в проекті. Після цього для них потрібно задати шлях у браузері, ім’я та компоненту.

Контентом головної сторінки є слайдер, який зображений на рисунку 3.5, з головними продуктами під яким знаходяться картки з усіма пропозиціями. На картці товару є можливість придбати його або відкрити повну сторінку, також під фото є короткий опис пропозиції. Зовнішній вигляд картки відображений на рисунку 3.6.

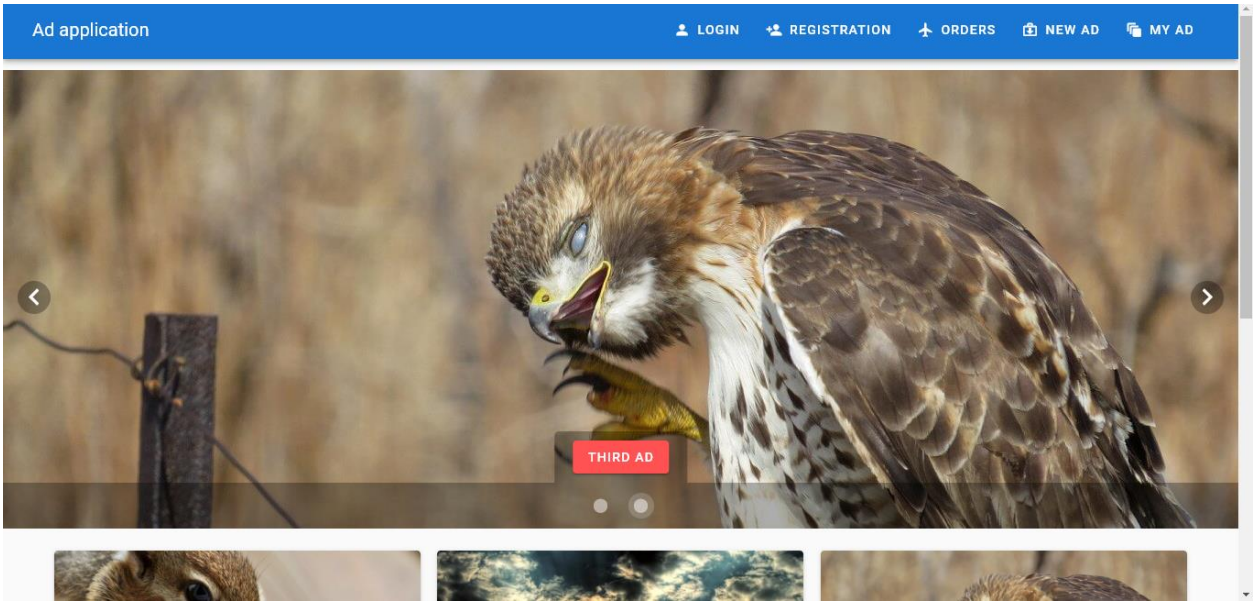

Рисунок 3.5 – Зовнішній вигляд верхньої частини головної сторінки

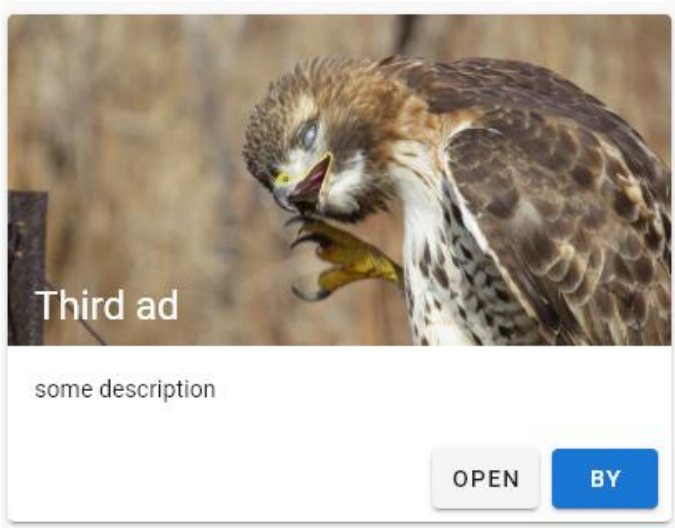

Рисунок 3.6 – Зовнішній вигляд картки товару

Також на головній сторінці використовується сховище. «Сховище» - це контейнер, в якому зберігається стан додатку. Два моменти відрізняють сховище Vuex від простого глобального об'єкта:

- Сховище Vuex реактивно. Коли компоненти Vue покладаються на його стан, то вони будуть реактивно і ефективно оновлюватися, якщо стан сховища змінюється.
- Не можна безпосередньо змінювати стан сховища. Єдиний спосіб змінити - явно викликати мутацію. Це гарантує, що будь-яка зміна стану залишає слід і дозволяє використовувати інструментарій, щоб краще розуміти хід роботи програми.

Інформацію для заповнення картки товару додаток бере з firebase

```
async fetchAds ({commit}) {
      commit('clearError')
      commit('setLoading', true)

      const resultAds = []

      try {
        const fbVal = await fb.database().ref('ads').once('value')
        const ads = fbVal.val()

        Object.keys(ads).forEach(key => {
          const ad = ads[key]
          resultAds.push(
            new Ad(ad.title, ad.description, ad.ownerId, ad.imageSrc, ad.promo, k
ey)
          )
        })

        commit('loadAds', resultAds)
        commit('setLoading', false)
      } catch (error) {
        commit('setError', error.message)
        commit('setLoading', false)
        throw error
      }
    },
```

На сторінці продукту знаходиться його велике зображення, опис, можливість придбати а також є кнопка яка дозволяє редагувати інформацію про товар, якщо цей він був зроблений вами. Зовнішній вигляд картки відображений на рисунку 3.7.

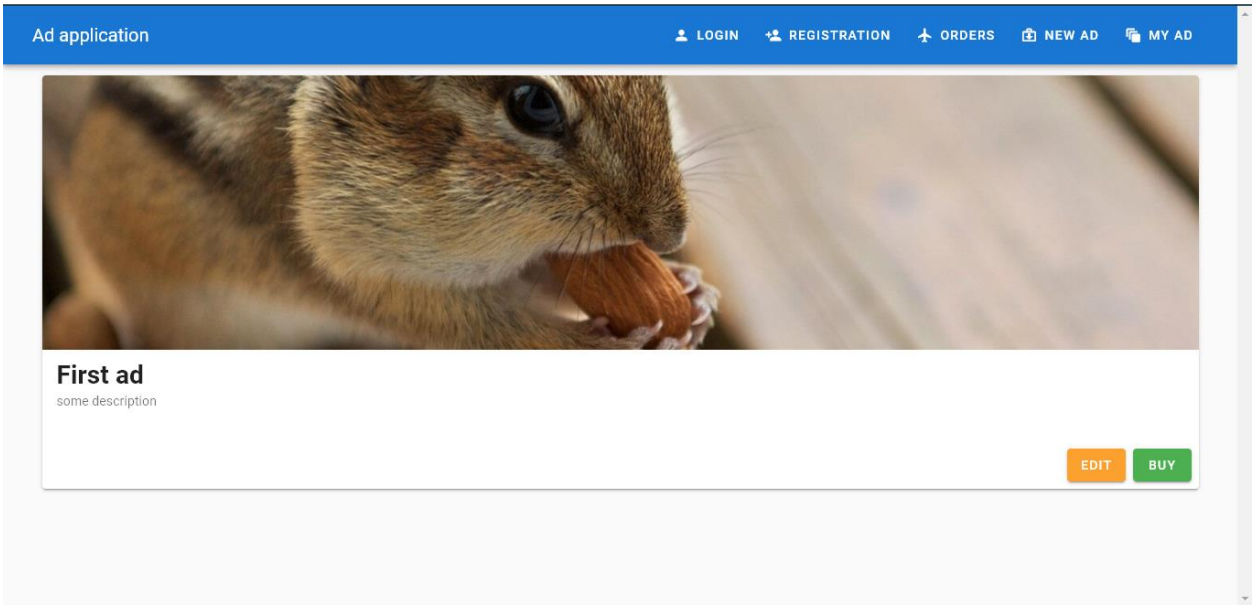

Рисунок 3.7 – Зовнішній вигляд сторінки товару

Наступні сторінки це Вхід та Реєстрація. Їх вигляд зображено на рисунку 3.8 та рисунку 3.8. Для швидкого та зручного налаштування валідації форми можна використовувати бібліотеку vuelidate, але в данному випадку було вирішено, що краще написати регулярний вираз для обробки поля e-mail:

```
const emailRegex = /^\w+([.-]?\w+)*@\w+([.-]?\w+)*(\.\w{2,3})+$/
```

Також для e-mail і паролю були зазначені додаткові правила:

```
emailRules: [
          v => !!v || 'E-mail is required',
          v => emailRegex.test(v) || 'E-mail must be valid'
        ],
        passwordRules: [
          v => !!v || 'Password is required',
          v => (v && v.length >= 6) || 'Password must be equal or more than 6
                                        characters'
        ],
        confirmPasswordRules: [
          v => !!v || 'Password is required',
          v => v === this.password || 'Password should match'
        ]
      }
```

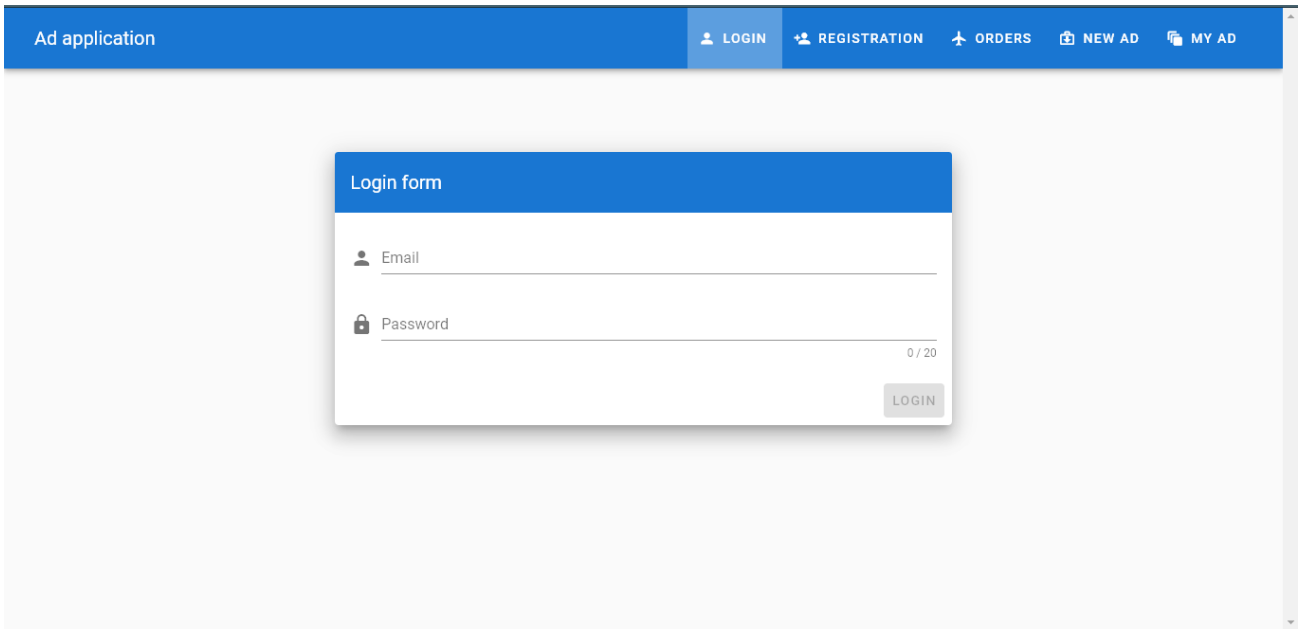

Рисунок 3.8 – Зовнішній вигляд сторінки Входу

На сторінці Реєстрації в порівнянні з сторінкою Входу є додаткове поле confirmPassword яке відповідає за додаткову перевірку паролю, для того щоб успішно завершити процедуру створення аккаунту необхідно в нього повторно ввести пароль.

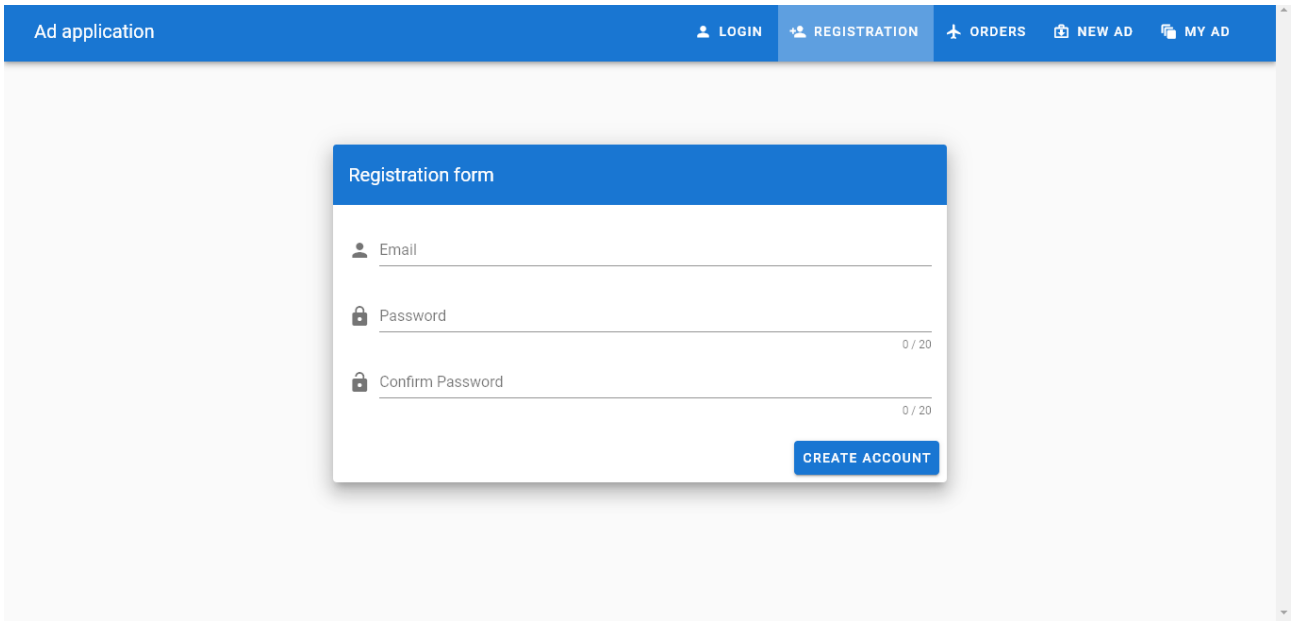

Рисунок 3.9 – Зовнішній вигляд сторінки Реєстрації

Далі по черзі сторінка замовлень. Вона зображена на рисунку 3.10. В ній продавець може бачити замовлення: номер телефону людини, її ім’я, і кнопку open, при натисканні на яку переходить на окрему сторінку де є можливість

побачити більше інформації. Також при натисканні на чекбокс можна помітити замовлення як виконане.

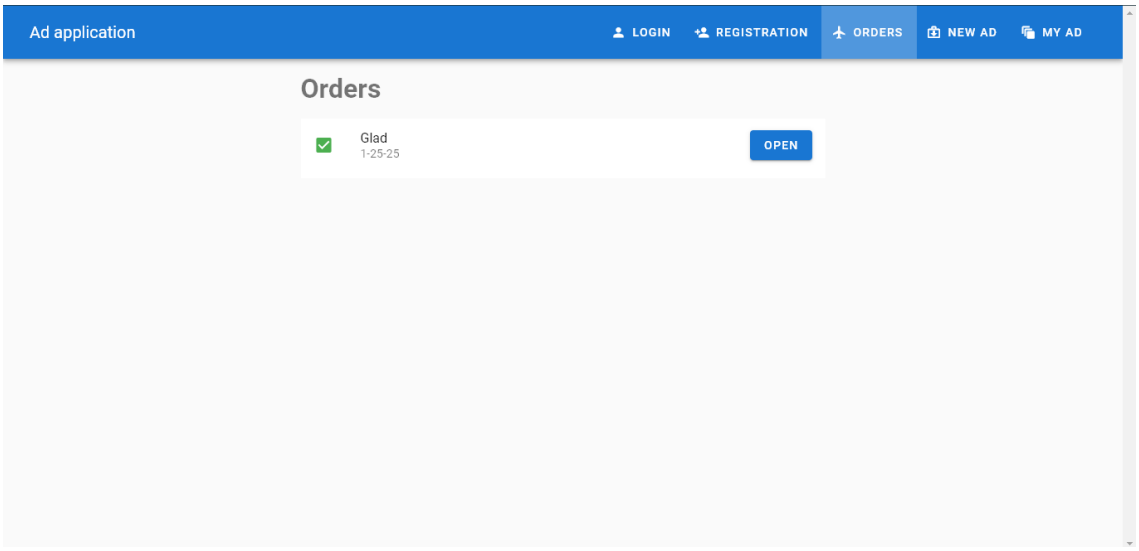

Рисунок 3.10 – Зовнішній вигляд сторінки Замовлень

Наступною є сторінка створення товару, яка зображена на рисунку 3.11. На ній є можливість заповнити наступні поля:

- title – назва товару
- description – опис товару
- upload – дозволяє завантажити зображення продукту
- show on main page – надає можливість відобразити товар у слайдері на головній сторінці

Після налаштування з'являється можливість додати товар.

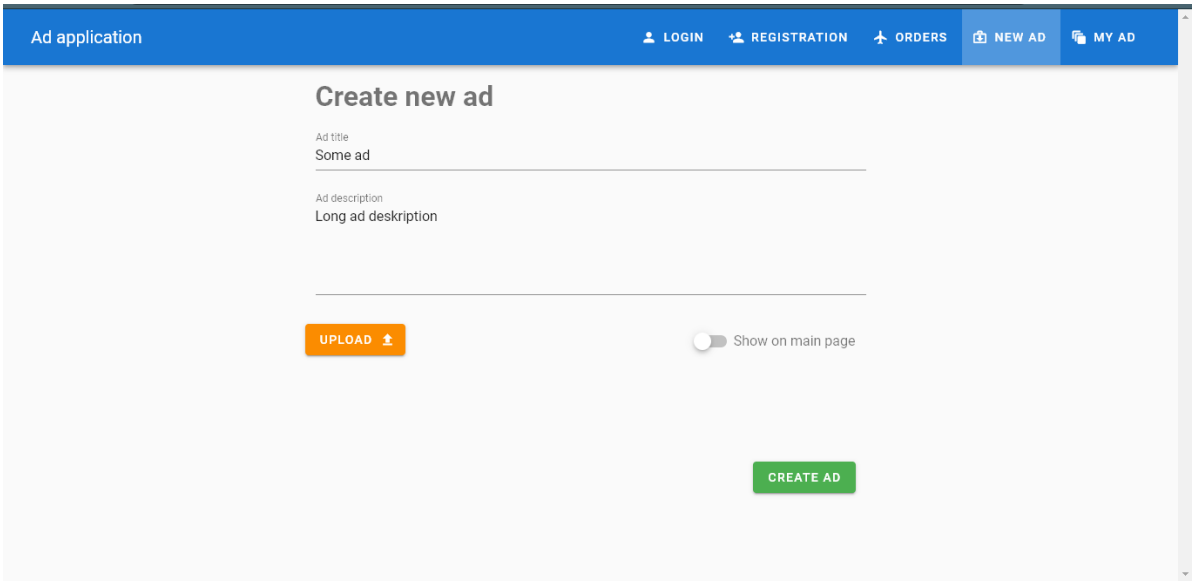

Рисунок 3.11 – Зовнішній вигляд сторінки Замовлень

Після натискання клавіші create ad в базі даних буде створено новий товар з параметрами які були введені в поля та id аккаунта який створив товар

```
class Ad {
  constructor (title, description, ownerId, imageSrc = '', promo = false,
id = null) {
    this.title = title
    this.description = description
    this.ownerId = ownerId
    this.imageSrc = imageSrc
    this.promo = promo
    this.id = id
  }
}
```

Останньою є сторінка доданих товарів, рисунок 3.12. На ній продавець може подивитись усі те що він продає. Біля кожного товару є кнопка open яка дозволяє відкрити повну версію сторінки товару для його перегляду або редагування.

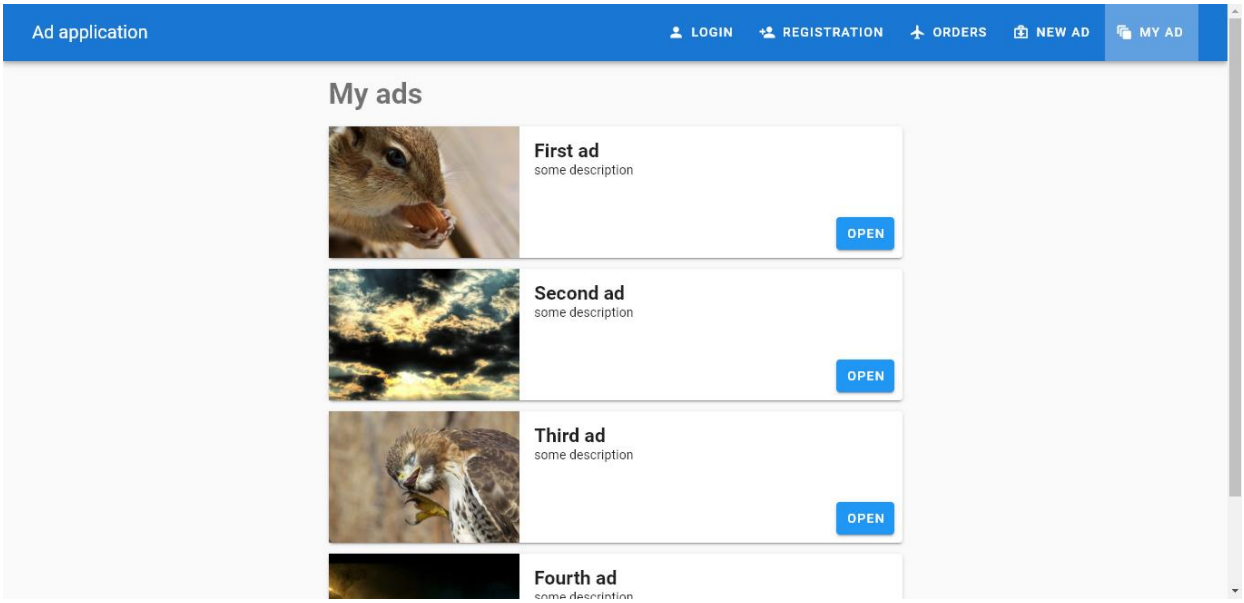

Рисунок 3.12 – Зовнішній вигляд сторінки замовлень

Також сайт містить декілька спливаючих вікон які зображені на рисунку 3.13 і 3.14. Вони знаходяться на сторінці товару і дозволяють редагувати або придбати продукт.

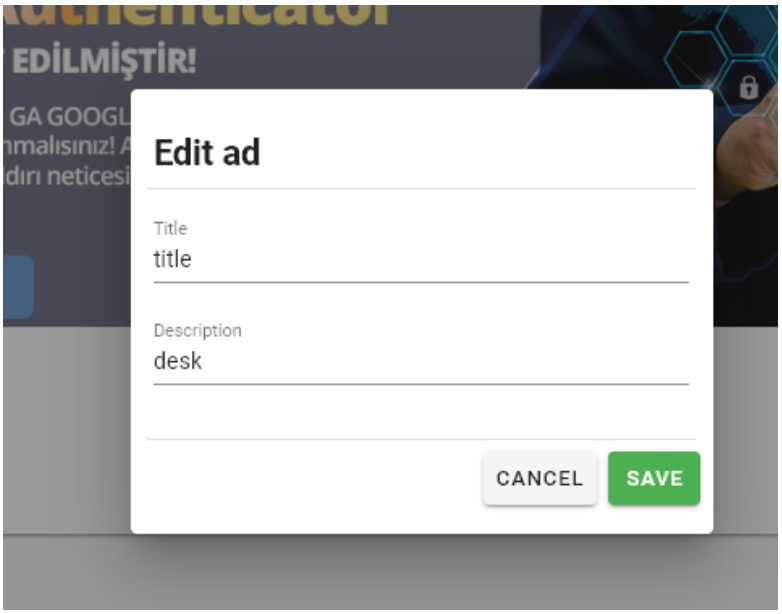

Рисунок 3.13 – Зовнішній вигляд редагування товару

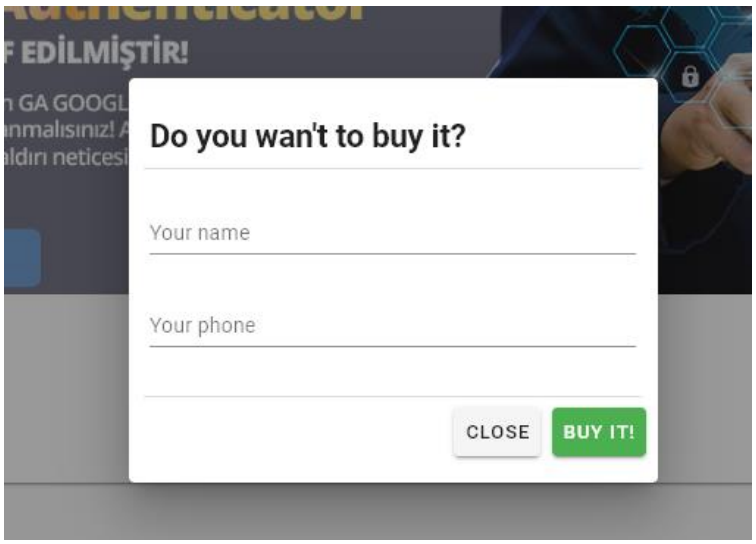

Рисунок 3.14 – Зовнішній вигляд замовлення товару

Після зміни інформації про товар всі виправлення будуть занесені до бази даних. Для того щоб зміни були збережені і надіслані до БД необхідно натиснути на кнопку save.

```
onSave () {
      if (this.editedDescription !== '' && this.editedTitle !== '') {
        this.$store.dispatch('updateAd', {
          title: this.editedTitle,
          description: this.editedDescription,
          id: this.ad.id
        })
        this.modal = false
      }
    }
```

Якщо людина передумає редагувати картку то вона може закрити вспливаюче вікно на кнопку close, або просто клацнути за його межами.

```
    onCancel () {
      this.editedDescription = this.ad.description
      this.editedTitle = this.ad.title
      this.modal = false
    },
```

При придбанні товару інформація про замовлення буде відображена у продавця на сторінці замовлень.

Вся інформація зберігається в базі даних firebase, де є можливість налаштувати засоби хешування паролю для його захисту. На даний час стоять такі параметри:

```
hash_config {
  algorithm: SCRYPT,
  base64_signer_key:mRqlycOrmcaOdNX6Jdl8MTu1GMjj1QekGGkB7j/jVGBz
WXqqrFMMqOIVMb49ix/dV8dYCUAiKj6LDXgOOsaz/w==,
  base64_salt_separator: Bw==,
  rounds: 8,
  mem_cost: 14,
}
```

ВИСНОВКИ

Робота виконана на фреймоврку vue.js. Це прогресивний JavaScript-фреймворк, який використовується для створення користувацьких інтерфейсів.

Було проведено огляд усіх популярних метадологій після якого вирішено що доцільніше всього буде зробити завдання на vue.js а як базу даних використовувати firebase.

Даний проект являє собою SPA інтернет магазину на якому зареєстрований користувач може купити або продати будь який товар.

Готовий продукт є швидким у завантаженні та не містить зайвої інформації, це досягнено завдяки мінімалістичного дизайну та використання новітніх технологій у сфері веб розробки.

СПИСОК ЛІТЕРАТУРИ

1. Керівництво по Vue.js [Електронний ресурс] – Режим доступу: https://metanit.com/web/vuejs/
2. Javascript-фреймворки: тенденції 2019 року [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/plarium/blog/433926/
3. Огляд Vue.js [Електронний ресурс] – Режим доступу: https://timeweb.com/ru/community/articles/obzor-vue-js-1
4. Документація Vue.js [Електронний ресурс] – Режим доступу: https://vuejs.org/
5. Документація Vuex [Електронний ресурс] – Режим доступу: https://vuex.vuejs.org/ru/guide/
6. ТОП 10 кращих фреймворків для Front-end Dev https://web-academy.com.ua/stati/336-10-front-end-dev-2018
7. Документація Vuetify.js [Електронний ресурс] – Режим доступу: https://vuetifyjs.com/ru/components/api-explorer/
8. Документація Vue router [Електронний ресурс] – Режим доступу: https://router.vuejs.org/ru/
9. Стандартний інструментарій для розробки на Vue.js [Електронний ресурс] – Режим доступу: https://cli.vuejs.org/ru/
10. Evan Y. The Majesty of Vue.js 2. -NY:Publisher, 2017 - 240 p.
11. Листуон Б. Vue.js в действии. - СПб: Видавництво Пітер, 2019 - 356 с.
12. Gerardus B. Firebase The Ultimate Step-By-Step Guide. - 5STARCooks, 2018 - 176 с.
13. Фримен А. Angular для профессионалов. - СПб: Видавництво Пітер, 2018 - 800 с.
14. Laurence M. The Definitive Guide to Firebase Apress.- Видавництво 1st ed. edition, 2017 - 275 с.

15. Banks A. React: Functional Web Development with React and Redux. - Видавництво O'Reilly Media; 1 edition, 2017 - 350 с

ДОДАТОК

Програмний код проекту

Папка src/components/Ads:

Файл Ad.vue

```
<template>
  <v-container>
    <v-layout row>
      <v-flex xs12>
        <v-card>
          <v-img
            :src="ad.imageSrc"
            height="300px"
          ></v-img>
          <v-card-text>
            <h1 class="text--primary mb-2">{{ ad.title }}</h1>
            <p>{{ ad.description }}</p>
          </v-card-text>
          <v-card-actions>
            <v-spacer></v-spacer>
            <v-btn class="warning" flat>Edit</v-btn>
            <v-btn class="success">Buy</v-btn>

          </v-card-actions>
        </v-card>
      </v-flex>
    </v-layout>
  </v-container>
</template>
```

```
<script>
  export default {
  props: ['id'],
  computed: {
    ad(state) {
      const id = this.id
      return this.$store.getters.adById(id)
    }
  }
}
</script>
```

Файл AdList.vue

```
<template>
  <v-container>
    <v-layout row>
      <v-flex xs12 sm6 offset-sm3>
        <h1 class="text--secondary mb-3">My ads</h1>

        <v-card v-for="ad in myAds" :key="ad.id" evelation-10 class="mb-3">
          <v-layout>
            <v-flex sm4>
              <v-img
                :src="ad.imageSrc"
                height="145px"
              ></v-img>
            </v-flex>
            <v-flex sm8>
              <v-card-text>
```

```
            <h2 class="text--primary">{{ ad.title }}</h2>
            <p>{{ ad.description }}</p>
          </v-card-text>

          <v-card-actions>
            <v-spacer></v-spacer>
            <v-btn :to="'/ad' + ad.id" class="info">Open</v-btn>
          </v-card-actions>
        </v-flex>
      </v-layout>
    </v-card>
  </v-flex>
 </v-layout>
</v-container>
</template>

<script>
 export default {
 computed : {
  myAds() {
   return this.$store.getters.myAds
  }
 }
}
</script>
```

Файл NewAd.vue

```
<template>
 <v-container>
  <v-layout row>
```

```
<v-flex xs12 sm6 offset-sm3>
  <h1 class="text--secondary mb-4">Create new ad</h1>
    <v-form  v-model="valid" ref="form" validation>
      <v-text-field
        label="Ad title"
        name="title"
        type="text"
        v-model="title"
        required
        :rules="[v => !!v || 'Title is required']"

      ></v-text-field>

      <v-textarea
        label="Ad description"
        name="description"
        type="text"
        required
        no-resize
        :rules="[v => !!v || 'Description is required']"
        v-model="description"
      ></v-textarea>
    </v-form>
    <v-layout>
      <v-flex row justify-space-between>
        <v-btn
          class="warning mt-3 mb-3"
        >
          Upload
          <v-icon right dark>mdi-upload</v-icon>
```

```
            </v-btn>

            <v-switch
              v-model="promo"
              label="Show on main page"
            ></v-switch>
          </v-flex>
        </v-layout>
        <v-layout>
          <v-flex row>
            <img src="" height="100" alt="">
          </v-flex>
        </v-layout>
        <v-layout>
          <v-flex row>
            <v-spacer></v-spacer>
            <v-
btn :disabled="!valid" class=success @click="createAd">Create Ad</v-btn>
          </v-flex>
        </v-layout>
      </v-flex>
    </v-layout>
  </v-container>
</template>

<script>
  export default {
  data() {
    return {
      title: '',
```

```
      description: '',
      promo: false,
      valid: false
    }
  },
  methods: {
    createAd() {
      if(this.$refs.form.validate()) {
        const ad = {
          title: this.title,
          description: this.description,
          promo: this.promo,
          imageSrc: 'http://mgnews.ru/image/normal/65554/the-witcher-3-wild-hunt-
1427359379296851.jpeg'
        }

        this.$store.dispatch('createAd', ad)
      }
    }
  }
}
</script>
```

Папка src/components/Auth:

Файл Login.vue

```
<template>
      <v-content>
      <v-container
        class="fill-height"
        fluid
```

```
>
  <v-row
    align="center"
    justify="center"
  >
    <v-col
      cols="12"
      sm="8"
      md="6"
    >
      <v-card class="elevation-12">
        <v-toolbar
          color="primary"
          dark
          flat
        >
          <v-toolbar-title>Login form</v-toolbar-title>

        </v-toolbar>
        <v-card-text>
          <v-form  v-model="valid" ref="form" validation>
            <v-text-field
              label="Email"
              name="email"
              prepend-icon="mdi-account"
              type="email"
              v-model="email"
              :rules="emailRules"

            ></v-text-field>
```

```
            <v-text-field
              id="password"
              label="Password"
              name="password"
              prepend-icon="mdi-lock"
              type="password"
              :rules="passRules"
              :counter="20"
              v-model="password"
            ></v-text-field>
          </v-form>
        </v-card-text>
        <v-card-actions>
          <v-spacer></v-spacer>
          <v-btn color="primary"
            :disabled="!valid"
            @click="onSubmit"
          >Login</v-btn>
        </v-card-actions>
      </v-card>
    </v-col>
  </v-row>
 </v-container>
</v-content>
</template>

<script>
 export default {
  data() {
```

```
    return {
      valid: false,
      emailRules: [
        v => !!v || 'E-mail is required',
        v => /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,3}))$/.test(v) || 'E-mail must be valid',
      ],
      passRules: [
        v => !!v || 'Password is required',
        v => (v && v.length>=6 && v.length <= 30) || 'Password must be lonher than
 6 and less then 30 characters',
      ],
      email: '',
      password: ''
    }
  },
  methods: {
   onSubmit () {
    if (this.$refs.form.validate()) {
        const user = {
         email: this.email,
         password: this.password
        }
        console.log(user);
       }
    }
   }
  }
</script>
```

Файл Regestration.vue

```
<template>
    <v-content>
    <v-container
     class="fill-height"
     fluid
    >
     <v-row
      align="center"
      justify="center"
     >
      <v-col
       cols="12"
       sm="8"
       md="6"
      >
       <v-card class="elevation-12">
        <v-toolbar
         color="primary"
         dark
         flat
        >
         <v-toolbar-title>Registration form</v-toolbar-title>

        </v-toolbar>
        <v-card-text>
         <v-form  v-model="valid" ref="form" lazy-validation>
          <v-text-field
           label="Email"
```

```
  name="email"
  prepend-icon="mdi-account"
  type="email"
  v-model="email"
  :rules="emailRules"

></v-text-field>

<v-text-field
  id="password"
  label="Password"
  name="password"
  prepend-icon="mdi-lock"
  type="password"
  :rules="passRules"
  :counter="20"
  v-model="password"
></v-text-field>

<v-text-field
  id="password"
  label="Confirm Password"
  name="confirm-password"
  prepend-icon="mdi-lock-open"
  type="password"
  :rules="confirmPassRules"
  :counter="20"
  v-model="confirmPassword"
></v-text-field>
```

```
            </v-form>
          </v-card-text>
          <v-card-actions>
            <v-spacer></v-spacer>
            <v-btn color="primary"
              :disabled="!valid"
              @click="onSubmit"
            >Create account</v-btn>
          </v-card-actions>
        </v-card>
      </v-col>
    </v-row>
  </v-container>
 </v-content>
</template>

<script>
 export default {
  data() {
   return {
    valid: false,
    emailRules: [
     v => !!v || 'E-mail is required',
     v => /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,3}))$/.test(v) || 'E-mail must be valid',
    ],
    passRules: [
     v => !!v || 'Password is required',
```

```
      v => (v && v.length>=6 && v.length <= 30) || 'Password must be lonher than
 6 and less then 30 characters',
    ],
    confirmPassRules:  [
      v => !!v || 'Password is required',
       v => v === this.password || 'Password doesn\'t match'
    ],
    email: '',
    password: '',
    confirmPassword: ''
   }
  },
  methods: {
   onSubmit () {
    if (this.$refs.form.validate()) {
       const user = {
        email: this.email,
        password: this.password
       }
       this.$store.dispatch('registerUser', user)
      }
    }
   }
  }
</script>
```

Папка src/components/User:

Файл Orders.vue

```
<template>
  <v-container>
```

```
<v-layout row>
  <v-flex xs12 sm6 offset-sm3>
    <h1 class="text--secondary mb-3">Orders</h1>
    <v-list
      subheader
      two-line
      flat
    >

      <v-list-item-group
        v-model="settings"
        multiple
        v-for="order in orders"
        :key="order.id"
      >
        <v-list-item>
          <template>
            <v-list-item-action>
              <v-checkbox
                :input.value="order.done"
                color="success"
                @change="markDone(order)"
              ></v-checkbox>
            </v-list-item-action>

            <v-list-item-content>
              <v-list-item-title>{{ order.name }}</v-list-item-title>
              <v-list-item-subtitle>{{ order.phone }}</v-list-item-subtitle>
            </v-list-item-content>
            <v-list-tile-action>
```

```
                <v-btn :to="'/ad' + order.adId" class='primary'>open</v-btn>
              </v-list-tile-action>
            </template>
          </v-list-item>

        </v-list-item-group>
      </v-list>
    </v-flex>
  </v-layout>
 </v-container>
</template>

<script>
 export default {
 data() {
   return {
     orders: [
       {
         id: '1',
         name: 'Glad',
         phone: '1-25-25',
         adId: '12',
         done: false
       }
     ]
   }
 },
 methods: {
   markDone() {
     orders.done == true
```

```
    }
  }
}
</script>
```

Папка src/components:

Файл Home.vue

```
<template>
  <div class="">
    <v-container fluid>
      <v-layout row>
        <v-flex xs12>
          <v-carousel>
            <v-carousel-item
              v-for="ad in promoAds"
              :key="ad.id"
              :src="ad.imageSrc"
            >
              <div class="carousel-link">
                <v-btn class="error" :to="'/ad' + ad.id">{{ ad.title }}</v-btn>
              </div>
            </v-carousel-item>
            </v-carousel>
        </v-flex>
      </v-layout>
    </v-container>
    <v-container grid-list-lg>
      <v-layout row wrap>
        <v-flex xs12 sm6 md4 xl3
            v-for="ad in ads"
```

```
            :key="ad.id">
          <v-card
            class="mx-auto"
            max-width="400"
          >
            <v-img
              class="white--text"
              height="200px"
              :src="ad.imageSrc"
            >
              <v-card-title  class="align-end fill-height">{{ ad.title}}</v-card-title>
            </v-img>

            <v-card-text>
              <span class="text--primary">
                <span>{{ ad.description }}</span>
              </span>
            </v-card-text>

            <v-card-actions>
              <v-spacer></v-spacer>
              <v-btn :to="'/ad' + ad.id"  flat>Open</v-btn>
              <v-btn class="primary" raised >By</v-btn>
            </v-card-actions>
          </v-card>
        </v-flex>
      </v-layout>
    </v-container>
  </div>
</template>
```

```
<script>
 export default {
    computed: {
    promoAds() {
      return this.$store.getters.promoAds
    },
    ads() {
      return this.$store.getters.ads
    }
  }
  }
</script>

<style scoped>
  .carousel-link{
    position: absolute;
    bottom: 50px;
    left: 50%;
    background: rgba(0, 0, 0, .3);
    transform: translate(-50%, 0);
    padding: 10px 20px;
    border-top-right-radius: 5px;
    border-top-left-radius: 5px;
  }

</style>
```

Папка src/plugins:

Файл vuetify.js

```
import Vue from 'vue';
import Vuetify from 'vuetify/lib';

Vue.use(Vuetify);

export default new Vuetify({
  icons: {
    iconfont: 'mdi',
  },
});
```

Папка src/store:

Файл ads.js

```
export default {
    state: {
        ads: [
            {
                title: 'First ad',
                description: 'some description',
                promo: false,
                imageSrc: 'https://cdn.vuetifyjs.com/images/carousel/squirrel.jpg',
                id: '1'
            },
            {
                title: 'Second ad',
                description: 'some description',
                promo: true,
                imageSrc: 'https://cdn.vuetifyjs.com/images/carousel/sky.jpg',
                id: '2'
            },
```

```
        {
            title: 'Third ad',
            description: 'some description',
            promo: true,
            imageSrc: 'https://cdn.vuetifyjs.com/images/carousel/bird.jpg',
            id: '3'
        },
        {
            title: 'Fourth ad',
            description: 'some description',
            promo: false,
            imageSrc: 'https://cdn.vuetifyjs.com/images/carousel/planet.jpg',
            id: '4'
        }
    ]
},
mutations: {
        createAd (state, payload) {
            state.ads.push(payload)
        }
},
actions: {
        createAd ({commit}, payload) {
            payload.id = 'ffdd'
            commit('createAd', payload)
        }
},
getters: {
        ads (state) {
            return state.ads
```

```
        },
        promoAds (state) {
            return state.ads.filter(ad => {
                return ad.promo
            })
        },
        myAds (state) {
            return state.ads
        },
        adById (state) {
            return adId => {
                return state.ads.find(ad => ad.id === adId)
            }
        }
    }
}
```

Файл shared.js

```
export default {
    state: {
        loading: false,
        error: null
    },
    mutations: {
        setLoading(state, payload) {
            state.loading = payload
        },
        setError(state, payload) {
            state.error = payload
        },
```

```
        clearError(state) {
            state.error = null
        }
    },
    actions: {
        setLoading({commit}, payload) {
            commit('setLoading', payload)
        },
        setError({commit}, payload) {
            commit('setError', payload)
        },
        clearError({commit}) {
            commit('clearError')
        }
    },
    getters: {
        loading(state) {
            return state.loading
        },
        error(state) {
            return state.error
        }
    }
}
```

Файл users.js

```
import * as fb from 'firebase'

class user {
    constructor(id) {
```

```
        this.id = id
    }
}

export default {
    state: {
        user: null
    },
    mutations: {
        setUSer(state, payload) {
            state.user = payload
        }
    },
    actions: {
        registerUser({commit}, {email, password}) {
            commit('clearError')
            commit('setLoading', true)
            fb.auth().createUserWithEmailAndPassword(email, password)
                .then(user => {
                    commit('setUSer', new User(user.uid))
                    commit('setLoading', false)
                })
                .catch(error => {
                    commit('setLoading', false)
                    commit('setError', error.masage)
                })
        }
    },
    users: {
        user() {
```

```
        return state.user
      }
    }
}
```

Папка src:

Файл App.vue

```
<template>
  <v-app id="inspire">
    <v-navigation-drawer
      v-model="drawer"
      app
      dark
      color="primary"

    >

    <v-list  color="primary" dark>
      <v-subheader>Menu</v-subheader>
      <v-list-item-group
        v-for="link of links"
        :key="link.title"

      >
      <v-list-item
        :to="link.url"
      >
        <v-list-item-icon>
          <v-icon >{{ link.icon }}</v-icon>
        </v-list-item-icon>
```

```
        <v-list-item-content>
          <v-list-item-title v-text="link.title"></v-list-item-title>
        </v-list-item-content>
      </v-list-item>
    </v-list-item-group>
  </v-list>

  </v-navigation-drawer>

  <v-app-bar
    app
    dark
    color="primary"

  >
    <v-app-bar-nav-icon @click.stop="drawer = !drawer" class="d-sm-flex d-md-
none">
    </v-app-bar-nav-icon>
    <v-toolbar flat dark color="primary">
    <v-toolbar-title>
      <router-link to="/" tag="span" class="pointer">Ad application</router-link>
    </v-toolbar-title>

    <v-spacer></v-spacer>

    <v-toolbar-items
      v-for="link of links"
      :key="link.title"
      class="hidden-sm-and-down"
```

```
      >
        <v-btn text :to="link.url">
          <v-icon left>{{ link.icon }}</v-icon>
          {{ link.title }}
        </v-btn>

      </v-toolbar-items>


    </v-toolbar>
    </v-app-bar>

    <v-content>
      <router-view></router-view>
    </v-content>

  </v-app>
</template>

<script>
  export default {
    props: {
      source: String,
    },
    data: () => ({
      drawer: false,
      links: [
        {title: 'Login', icon: 'mdi-account', url: '/login'},
        {title: 'Registration', icon: 'mdi-account-plus', url: '/registration'},
        {title: 'Orders', icon: 'mdi-airplane', url: '/orders'},
```

```
        {title: 'New ad', icon: 'mdi-briefcase-download-outline', url: '/new'},
        {title: 'My ad', icon: 'mdi-animation', url: '/list'}
       ]
    }),
  }
</script>

<style scoped>
  .pointer{
    cursor: pointer
}

</style>
```

Файл main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import * as fb from 'firebase'
import vuetify from './plugins/vuetify';

Vue.config.productionTip = false

new Vue({
  router,
  store,
  vuetify,
  render: h => h(App),
  created() {
```

```
    fb.initializeApp({
      apiKey: "AIzaSyDc3mQYa00xGgWHQydpBo6wkyRpGjx_Z5Q",
      authDomain: "itc-ad-pproject.firebaseapp.com",
      databaseURL: "https://itc-ad-pproject.firebaseio.com",
      projectId: "itc-ad-pproject",
      storageBucket: "",
      messagingSenderId: "300784422937",
      appId: "1:300784422937:web:e1058b5187fb6c9f235a88"
    })
}
}).$mount('#app')
```

Файл router.js

```
import Vue from 'vue'
import Router from 'vue-router'

import MyHome from '@/components/Home.vue'
import Ad from '@/components/Ads/Ad.vue'
import AdList from '@/components/Ads/AdList.vue'
import NewAd from '@/components/Ads/NewAd.vue'
import Login from '@/components/Auth/Login.vue'
import Registration from '@/components/Auth/Registration.vue'
import Orders from '@/components/User/Orders.vue'

Vue.use(Router)

export default new Router({
  routes: [
    {
      path: '',
```

```
  name: 'home',
  component: MyHome
},
{
  path: '/ad:id',
  name: 'ad',
  props: true,
  component: Ad
},
{
  path: '/list',
  name: 'list',
  component: AdList
},
{
  path: '/new',
  name: 'newAd',
  component: NewAd
},
{
  path: '/login',
  name: 'login',
  component: Login
},
{
  path: '/registration',
  name: 'reg',
  component: Registration
},
{
```

```
      path: '/orders',
      name: 'orders',
      component: Orders
    }
  ],
  mode: 'history'
})
```

Файл store.js

```
import Vue from 'vue'
import Vuex from 'vuex'
import ads from './store/ads'
import user from './store/user'
import shared from './store/shared'

Vue.use(Vuex)

export default new Vuex.Store({
    modules: {
      ads,
      user,
      shared
    }
})
```