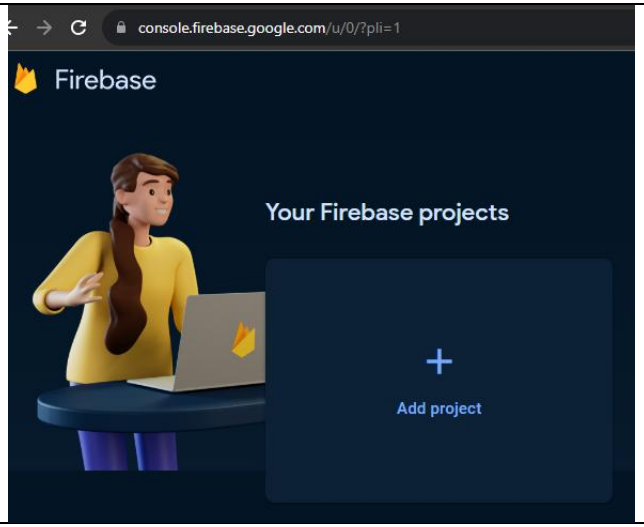
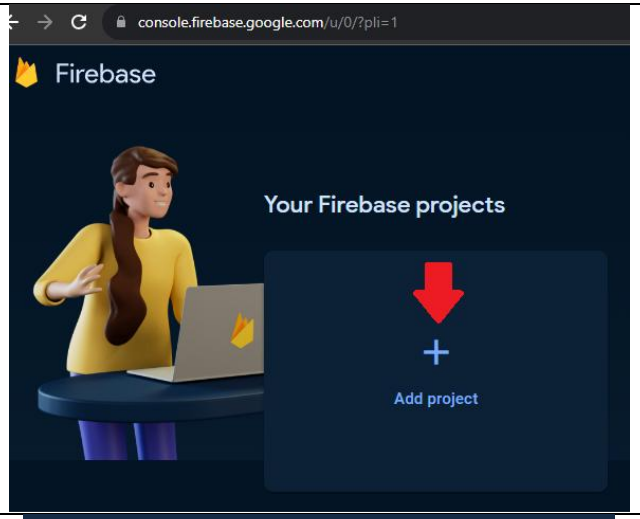
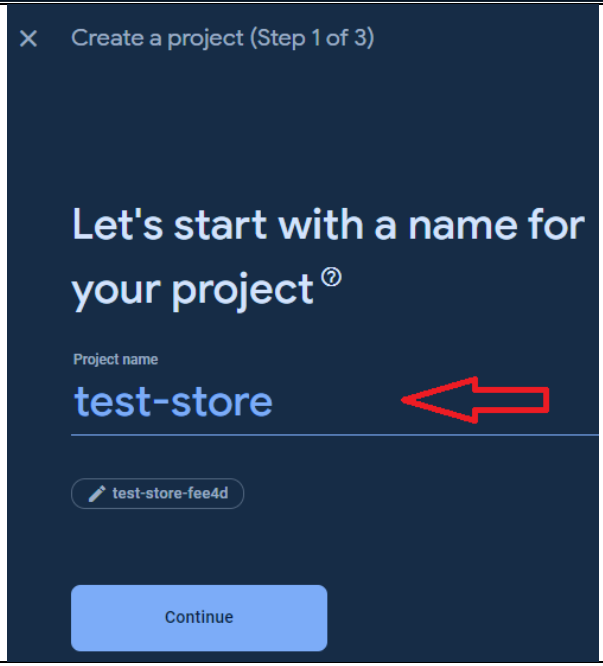
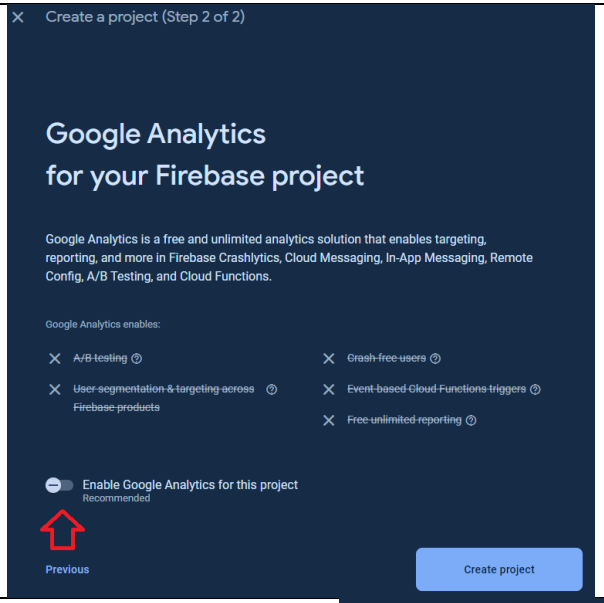
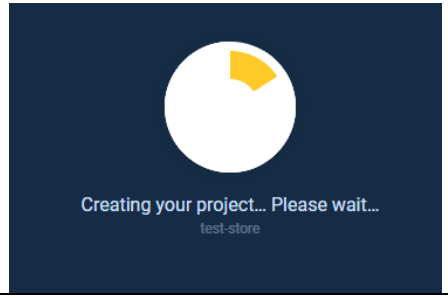
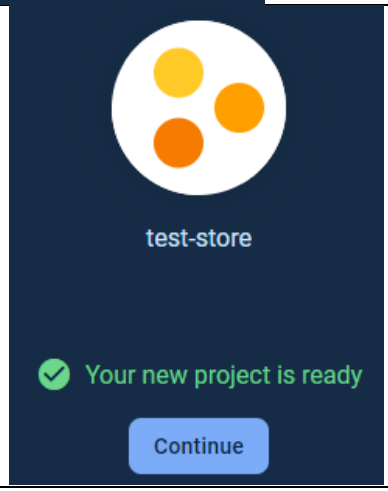
Vite. Робота з Firestore Database в Firebase

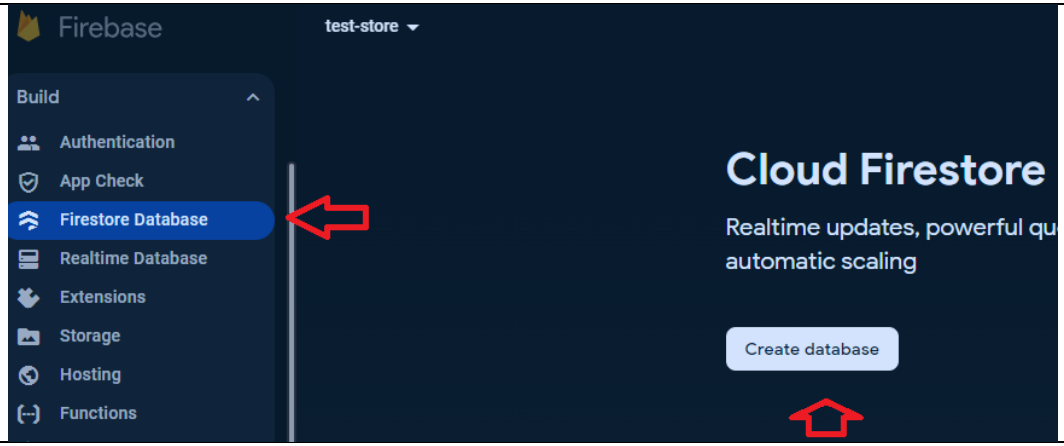
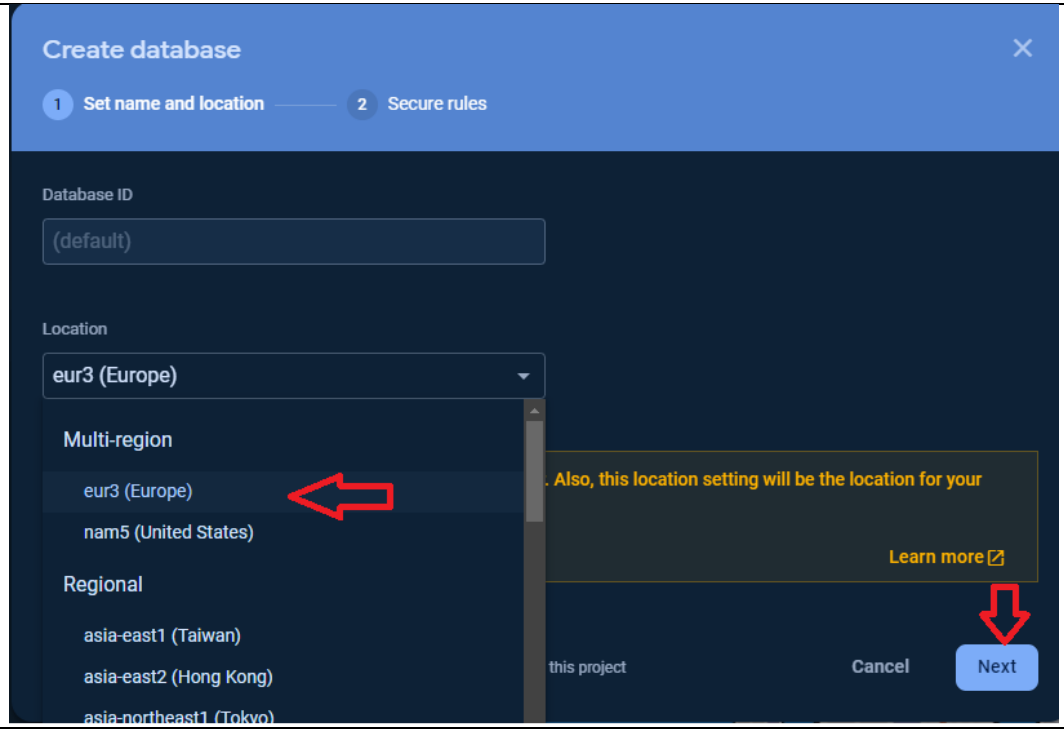
Add Firebase to your JavaScript project

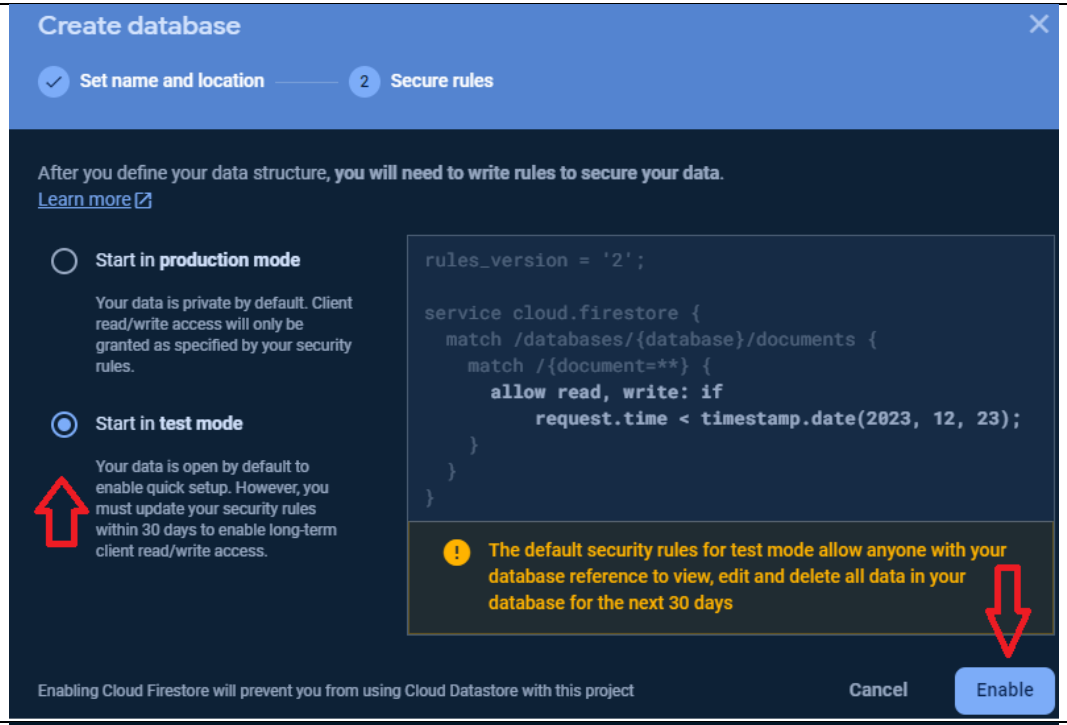
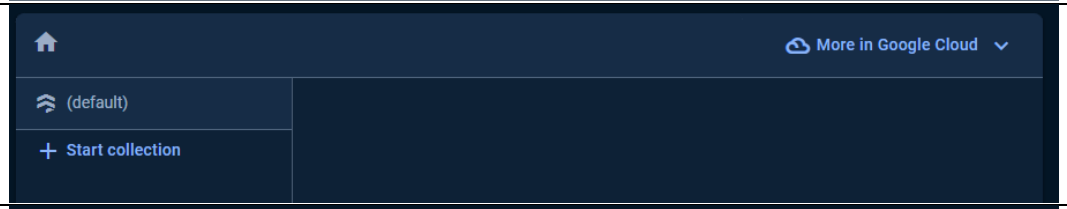
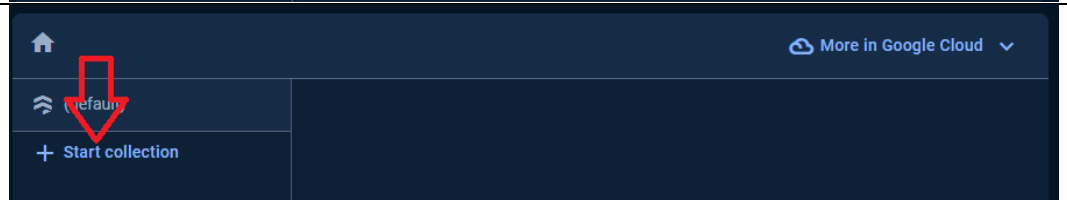
https://firebase.google.com/docs/web/setup?hl=en&authuser=0&_gl=1*1as5592*_ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*_ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA.

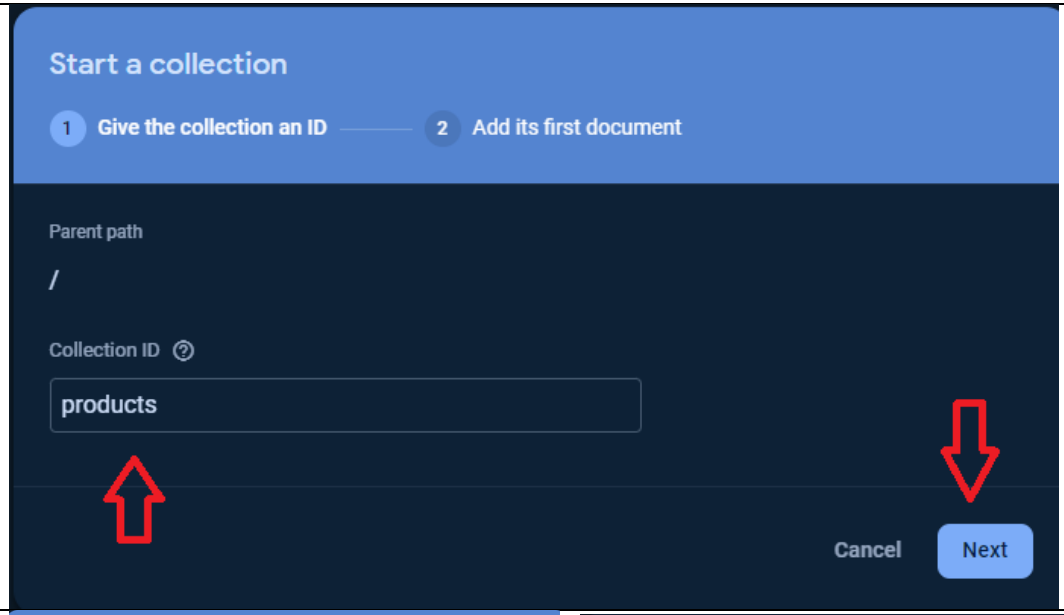
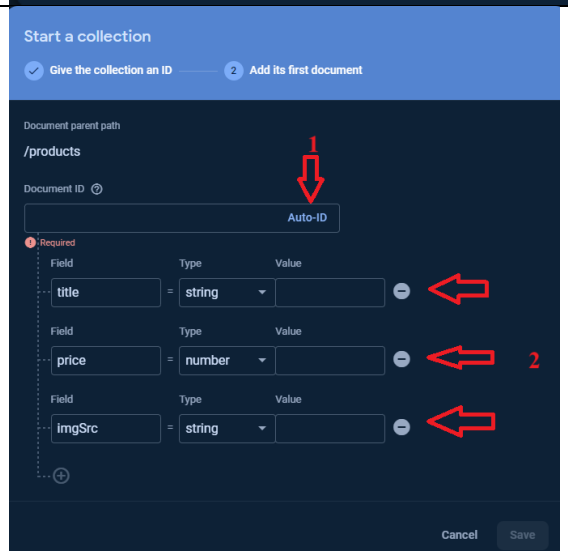
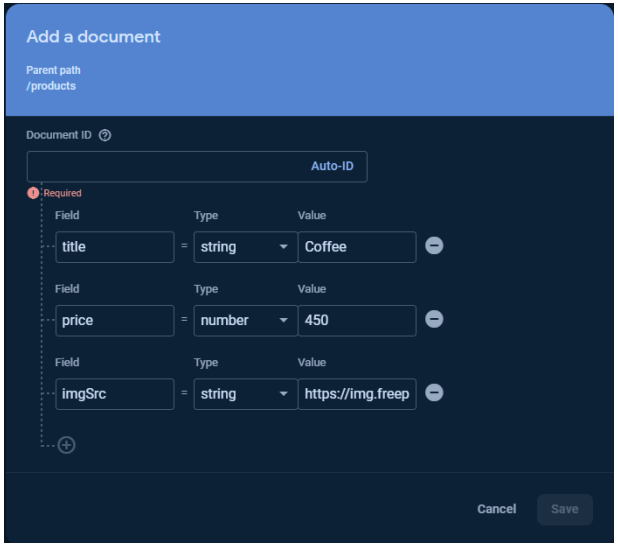
Переходимо у консоль firebase https://console.firebase.google.com/u/0/?pli=1		

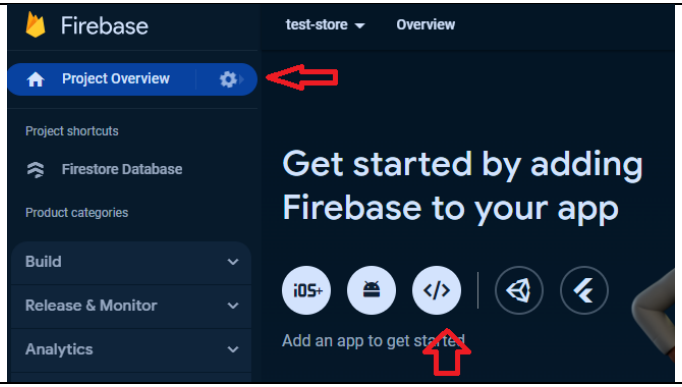
Створюємо проект в Firebase		
Задаємо якусь назву проєкта		

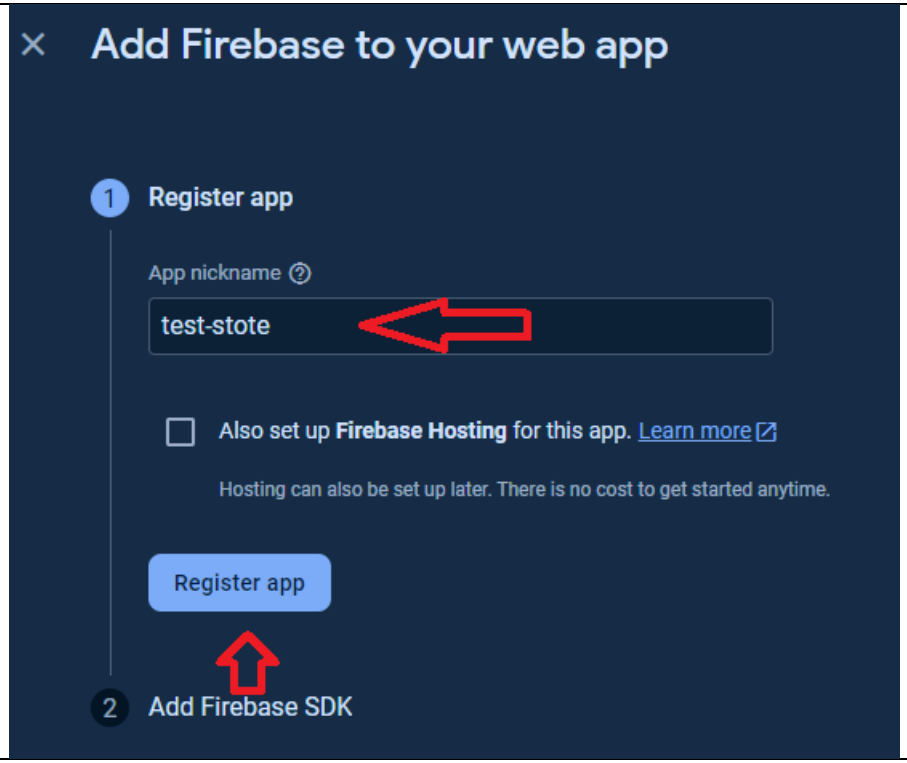
Відключаємо аналітику	
Створюємо проєкт	 

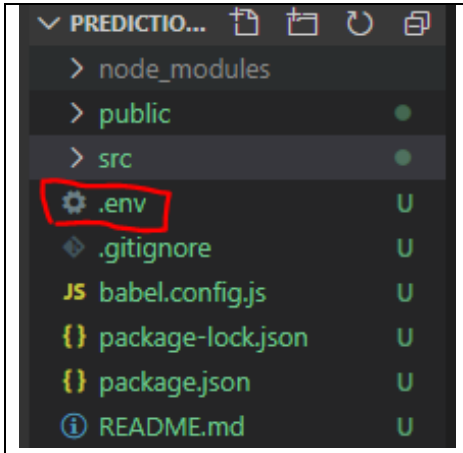
Додаємо Firestore Database	
	

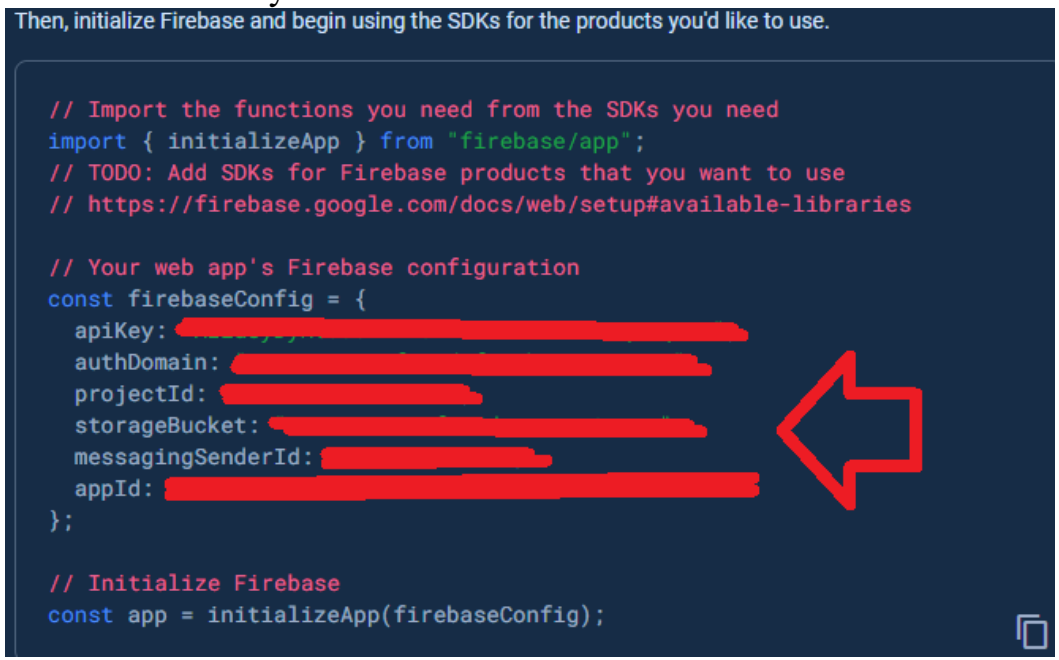
	
	
Створюємо колекцію даних	

Задаємо ім’я колекції	
1)Визначаємо спосіб задання id як автоматичний 2)Додаємо документ (задаємо поля документа(назва, тип, значення) Можемо задати вручну декілька документів	 

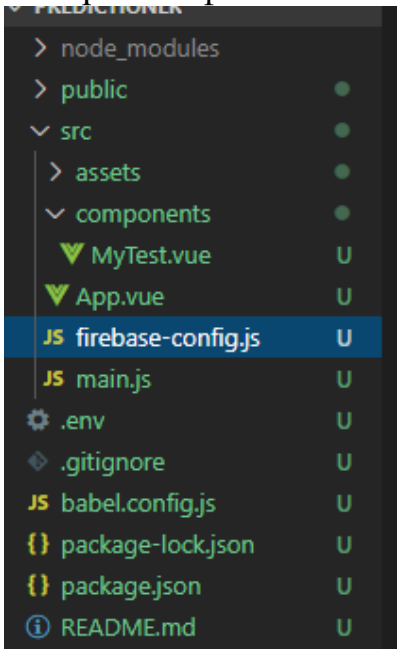
<table>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Додаємо firebase до проєкта</td><td></td><td> 
</td><td></td></tr>
</table>

Задаємо назву і реєструємо	
Встановлюємо необхідні модулі https://firebase.google.com/docs/web/setup?hl=en&authuser=0&_gl=1*1as5592*_ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*_ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA.	`npm install firebase`
Створюємо файл .env	Додаємо наш додаток в Firebase проект дані з Вашого проекту _APP_FIREBASE_AUTH_DOMAIN= дані з Вашого проекту = дані з Вашого проекту _APP_FIREBASE_PROJECT_ID= дані з Вашого проекту _APP_FIREBASE_STORAGE_BUCKET= дані з Вашого проекту _APP_FIREBASE_MESSAGE_SENDER_ID= дані з Вашого проекту _APP_FIREBASE_APP_ID= дані з Вашого проекту

<table>
<tr>
<td> 

https://cli.vuejs.org/ru/guide/mode-and-env.html#%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5-%D0%BE%D0%BA%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F</td>
<td>Копіюємо з сайту
 
</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</table>

Створюємо файл налаштувань firebase



Або для початку можна просто скопіювати firebaseConfig з сайту без створення файлу .env

```
import { initializeApp } from 'firebase/app'
import { getFirestore } from 'firebase/firestore/lite'

const firebaseConfig = {
  apiKey: import.meta.env.VITE_APP_FIREBASE_API_KEY,
  authDomain: import.meta.env.VITE_APP_FIREBASE_AUTH_DOMAIN,
  projectId: import.meta.env.VITE_APP_FIREBASE_PROJECT_ID,
  storageBucket: import.meta.env.VITE_APP_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: import.meta.env.VITE_APP_FIREBASE_MESSAGE_SENDER_ID,
  appId: import.meta.env.VITE_APP_FIREBASE_APP_ID,
}

const app = initializeApp(firebaseConfig)
const db = getFirestore(app)
export default db
```

```
---------------- ОПЕРАЦІЇ З БАЗОЮ ДАНИХ -----------------
//---- отримуємо посилання на базу даних
import firebaseDB from ``
//---- імпортуємо методи для роботи з даними
import { doc, collection, getDocs, addDoc, deleteDoc, updateDoc, query, where } from 'firebase/firestore/lite'

//---- отримуємо посилання на колекцію
this.dbCollection = collection(firebaseDB, `назва колекції`)
```

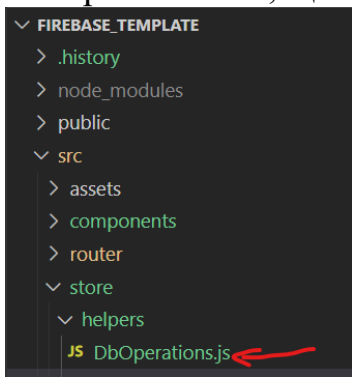
Зчитуваня даних
https://firebase.google.com/docs/web/setup?hl=en&authuser=0&_gl=1*1as5592*_ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*_ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA.

```
            getDocs(this.dbCollection)
                .then((querySnapshot) => {
                    const list = []
                    querySnapshot.docs.forEach((doc) => {
                        list.push({
                            id: doc.id,
```

https://firebase.google.com/docs/reference/js/firestore_lite.md?hl=ru#getdocs	```...doc.data(), }) }) resolve(list) }) .catch((error) => { reject(error) })```
Додавання даних https://firebase.google.com/docs/reference/js/firestore_lite.md?hl=ru#adddoc	```addDoc(this.dbCollection, item) .then(() => { resolve(true) }) .catch((error) => { reject(error) })```
Видалення даних https://firebase.google.com/docs/reference/js/firestore_lite.md?hl=ru#deletedoc	```deleteDoc(doc(this.dbCollection, id)) .then(() => { resolve(true) }) .catch((error) => { reject(error) })```
Модифікація даних https://firebase.google.com/docs/reference/js/firestore_lite.md?hl=ru#updatedoc_2	```updateDoc(oldDocRef, data) .then(() => { resolve(true) }) .catch((error) => { reject(error) })```
Фільтрація **https://cloud.google.com/firestore/docs/query-data/queries**	```loadFilteredData(fieldTitle, compareOperator, valueToCompare) { const q = query(this.dbCollection, where(fieldTitle, compareOperator, valueToCompare)) return new Promise((resolve, reject) => {```

```
            getDocs(q)
                .then((querySnapshot) => {
                    const list = []
                    querySnapshot.docs.forEach((doc) => {
                        list.push({
                            id: doc.id,
                            ...doc.data(),
                        })
                    })
                    resolve(list)
                })
                .catch((error) => {
                    reject(error)
                })
        })
    }
```

Створюємо клас, що містить основні операції з базою




```
import firebaseDB from '@/firebase-config'
import {
  doc,
  collection,
  getDocs,
  getDoc,
  addDoc,
  deleteDoc,
  updateDoc,
  query,
  where,
  startAfter,
  limit,
  orderBy,
} from 'firebase/firestore/lite'

class DbOperations {
  constructor(collectionTitle) {
    this.dbCollection = collection(firebaseDB, `/${collectionTitle}`)
  }
```

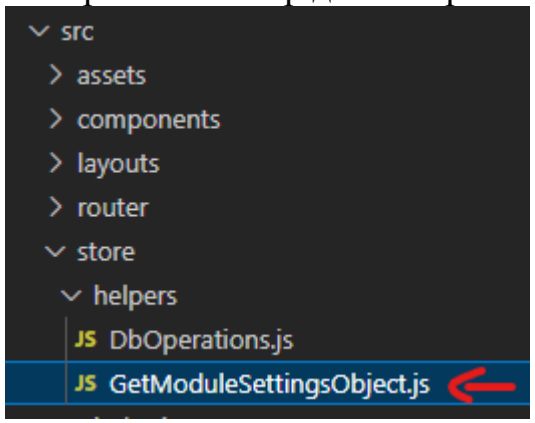
```
getItemFromSnap(docSnap) {
  return {
    id: docSnap.id,
    ...docSnap.data(),
  }
}

getListFromSnapshot(snapshot) {
  const list = []
  snapshot.docs.forEach((doc) => {
    // list.push({
    //   id: doc.id,
    //   ...doc.data(),
    // })
    list.push(this.getItemFromSnap(doc))
  })
  return list
}
// loadItemsList() {
//   return new Promise((resolve, reject) => {
//     getDocs(this.dbCollection)
//       .then((querySnapshot) => {
//         resolve(this.getListFromSnapshot(querySnapshot))
//       })
//       .catch((error) => {
//         reject(error)
//       })
//   })
// }
getQueryOptions(options, filter) {
  const queryOpt = []
  if (filter) queryOpt.push(filter)

  if (options?.orderBy)
    queryOpt.push(
      orderBy(
```

```
        options.orderBy,

        options.orderType ?? 'asc'
      )
    )

  if (options?.page) {
    const limitNum = options?.limit ?? 5
    queryOpt.push(startAfter(options.page * limitNum))
    queryOpt.push(limit(limitNum))
  } else {
    if (options?.limit) queryOpt.push(limit(options?.limit))
  }
  return queryOpt
}
loadItemsList(options) {
  const queryOpt = this.getQueryOptions(options)
  return new Promise((resolve, reject) => {
    getDocs(query(this.dbCollection, ...queryOpt))
      .then((querySnapshot) => {
        resolve(this.getListFromSnapshot(querySnapshot))
      })
      .catch((error) => {
        reject(error)
      })
  })
}
//--------------
loadItemById(id) {
  return new Promise((resolve, reject) => {
    getDoc(doc(this.dbCollection, id))
      .then((querySnapshot) => {
        if (querySnapshot.exists()) {
          resolve(this.getItemFromSnap(querySnapshot))
        } else throw new Error('Items not exists')
      })
      .catch((error) => {
```

```
        reject(error)
      })
  })
}

addItem(item) {
  return new Promise((resolve, reject) => {
    addDoc(this.dbCollection, item)
      .then(() => {
        resolve(true)
      })
      .catch((error) => {
        reject(error)
      })
  })
}

deleteItem(id) {
  return new Promise((resolve, reject) => {
    deleteDoc(doc(this.dbCollection, id))
      .then(() => {
        resolve(true)
      })
      .catch((error) => {
        reject(error)
      })
  })
}

updateItem(itemId, data) {
  return new Promise((resolve, reject) => {
    const oldDocRef = doc(this.dbCollection, itemId)
    updateDoc(oldDocRef, data)
      .then(() => {
        resolve(true)
      })
      .catch((error) => {
```

```
          reject(error)
        })
    })
  }
  loadFilteredData({ fieldTitle, compareOperator, valueToCompare }, options)
{
    const filter = where(fieldTitle, compareOperator, valueToCompare)
    const queryOpt = this.getQueryOptions(options, filter)

    const q = query(this.dbCollection, ...queryOpt)

    return new Promise((resolve, reject) => {
      getDocs(q)
        .then((querySnapshot) => {
          resolve(this.getListFromSnapshot(querySnapshot))
        })
        .catch((error) => {
          reject(error)
        })
    })
  }
}

export default DbOperations
```

Створюємо гелпер для створення модулів




```
import DbOperations from './DbOperations'

function getModuleSettingsObject(collectionTitle) {
  const collectionDB = new DbOperations(collectionTitle)
  return {
    namespaced: true,

    state: () => ({
      [collectionTitle]: [],
      currentItem: null,
      loading: false,
      error: null,
    }),
```

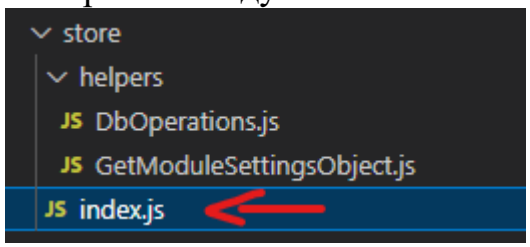
```
  getters: {
    isLoading: (state) => state.loading,
    hasError: (state) => state.error,

    getItemsList: (state) => state[collectionTitle],
    getItemById: (state) => (itemId) =>
      state[collectionTitle].find((item) => item.id == itemId),
    getCurrentItem: ({ currentItem }) => currentItem,
  },

  mutations: {
    setItemsList(state, itemsList) {
      state[collectionTitle] = itemsList
    },
    setCurrentItem(state, itemData) {
      state.currentItem = itemData
    },
    addItem(state, item) {
      state[collectionTitle].push(item)
    },
    deleteItem(state, deleteItemId) {
      state[collectionTitle] = state[collectionTitle].filter(
        (item) => item.id !== deleteItemId
      )
    },
    setLoading(state, value) {
      state.loading = value
    },
    setError(state, error) {
      state.error = error
    },
  },

  actions: {
    loadList({ commit }, options) {
      commit('setError', null)
```

```
    commit('setLoading', true)

    collectionDB
      .loadItemsList(options)
      .then((list) => {
        commit('setItemsList', list)
      })
      .catch((error) => {
        commit('setError', error)
      })
      .finally(() => {
        commit('setLoading', false)
      })
  },
  addItem({ commit, dispatch }, item) {
    commit('setError', null)
    commit('setLoading', true)
    collectionDB
      .addItem(item)
      .then(() => {
        dispatch('loadList')
      })
      .catch((error) => {
        commit('setError', error)
      })
      .finally(() => {
        commit('setLoading', false)
      })
  },
  deleteItem({ commit, dispatch }, itemId) {
    commit('setError', null)
    commit('setLoading', true)

    collectionDB
      .deleteItem(itemId)
      .then(() => {
        dispatch('loadList')
```

```
      })
      .catch((error) => {
        commit('setError', error)
      })
      .finally(() => {
        commit('setLoading', false)
      })
  },
  updateItem({ commit, dispatch }, { itemId, data }) {
    commit('setError', null)
    commit('setLoading', true)

    collectionDB
      .updateItem(itemId, data)
      .then(() => {
        dispatch('loadList')
      })
      .catch((error) => {
        commit('setError', error)
      })
      .finally(() => {
        commit('setLoading', false)
      })
  },
  loadItemDataById({ commit }, itemId) {
    commit('setError', null)
    commit('setLoading', true)
    commit('setCurrentItem', null)
    collectionDB
      .loadItemById(itemId)
      .then((itemData) => {
        commit('setCurrentItem', itemData)
      })
      .catch((error) => {
        commit('setError', error)
      })
      .finally(() => {
```

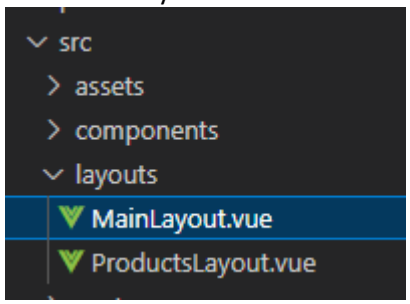
```
            commit('setLoading', false)
          })
      },
      loadFilteredData(
        { commit },
        { fieldTitle, compareOperator, valueToCompare, options }
      ) {
        commit('setError', null)
        commit('setLoading', true)
        collectionDB
          .loadFilteredData(
            { fieldTitle, compareOperator, valueToCompare },
            options
          )
          .then((list) => {
            commit('setItemsList', list)
          })
          .catch((error) => {
            commit('setError', error)
          })
          .finally(() => {
            commit('setLoading', false)
          })
      },
    },
  }
}

export default getModuleSettingsObject
```

Створюємо модуль




```
import { createStore } from 'vuex'
import getModuleSettingsObject from './helpers/GetModuleSettingsObject'
export default createStore({
  namespaced: true,
  modules: {
    products: getModuleSettingsObject('products'),
  },
})
```

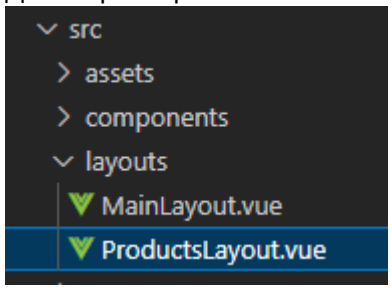
Основний layout



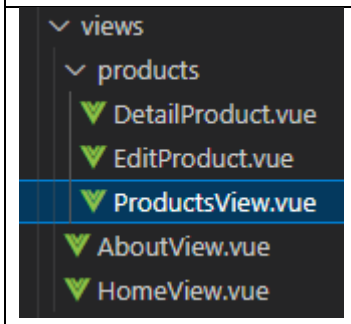

```
<template>
  <div>
    <div v-if="isLoading">Loading ......</div>
    <div v-else-if="hasError">Вибачте. Сталась помилка</div>
    <div v-else>
      <slot></slot>
    </div>
  </div>
</template>

<script>
export default {
  name: 'MainLayout',
  props: {
    isLoading: {
      type: Boolean,
      default: false,
    },
    hasError: {
      type: Boolean,
      default: false,
    },
  },
}
</script>
```

```vue
<style lang="scss" scoped></style>
```

Для сторінок products




```vue
<template>
  <main-layout :is-loading="isLoading" :has-error="hasError">
    <slot></slot>
  </main-layout>
</template>

<script>
import { mapGetters } from 'vuex'

export default {
  name: 'ProductsLayout',
  computed: {
    ...mapGetters('products', ['getItemsList', 'isLoading', 'hasError']),
  },
}
</script>

<style lang="scss" scoped></style>
```



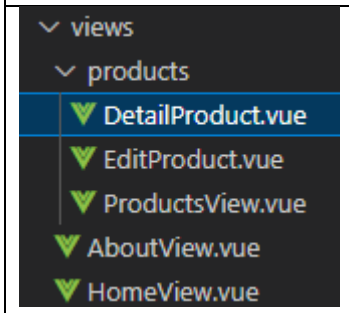

```vue
<template>
  <div class="d-flex">
    <div>
      <label>
        Title
        <input type="text" v-model.lazy="queryOptData.productTitle" />
      </label>
    </div>
    <div>
      <label>
        Order
        <select v-model="queryOptData.orderType">
          <option value="asc">За зростанням</option>
          <option value="desc">За спаданянм</option>
```

```
      </select>
    </label>
  </div>
</div>
<hr />
<h2>Список продуктів</h2>
<products-layout>
  <div v-for="item in getItemsList" :key="item.id" class="d-flex">
    <div>
      <router-link
        :to="{
          name: 'product-detail',
          params: {
            id: item.id,
          },
        }"
        >{{ item.title }}</router-link
      >
      -
      <span>{{ item.price }}</span>
    </div>
    <div>
      <router-link
        :to="{
          name: 'product-edit',
          params: {
            id: item.id,
          },
        }"
      >
        Edit
      </router-link>
      <button @click="deleteItem(item.id)">Delete</button>
    </div>
  </div>
  <hr />
  <router-link
```

```
      :to="{
        name: 'product-edit',
      }"
      >Додати товар</router-link
    >
  </products-layout>
</template>

<script>
import MainLayout from '@/layouts/MainLayout.vue'
import { mapGetters, mapActions } from 'vuex'
import ProductsLayout from '@/layouts/ProductsLayout.vue'

export default {
  components: { MainLayout, ProductsLayout },
  name: 'ProductsView',
  computed: {
    ...mapGetters('products', ['getItemsList', 'isLoading', 'hasError']),
    queryOptions() {
      return {
        orderBy: 'price',
        orderType: this.queryOptData.orderType,
      }
    },
  },

  data() {
    return {
      queryOptData: {
        orderType: null,
        productTitle: null,
      },
    }
  },

  watch: {
    queryOptData: {
```

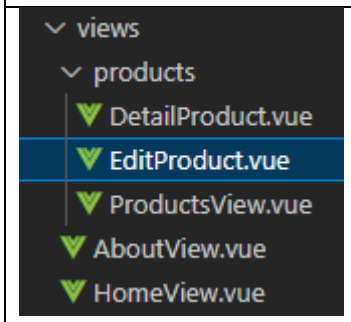
```
      handler() {
        this.loadData()
      },
      deep: true,
    },
  },

  methods: {
    ...mapActions('products', [
      'loadList',
      'addItem',
      'deleteItem',
      'updateItem',
      'loadFilteredData',
    ]),

    loadData() {
      if (this.queryOptData.productTitle)
        this.loadFilteredData({
          fieldTitle: 'title',
          compareOperator: '==',
          valueToCompare: this.queryOptData.productTitle,
          options: this.queryOptions,
        })
      else this.loadList(this.queryOptions)
    },
  },

  created() {
    this.loadList()
  },
}
</script>

<style lang="scss" scoped>
.d-flex {
  display: flex;
```

```
  justify-content: space-between;
}
</style>
```




```
<template>
  <products-layout>
    {{ getCurrentItem }}
  </products-layout>
</template>

<script>
import MainLayout from '@/layouts/MainLayout.vue'
import { mapGetters, mapActions } from 'vuex'
import ProductsLayout from '@/layouts/ProductsLayout.vue'
export default {
  components: { MainLayout, ProductsLayout },
  name: 'DetailProduct',
  props: {
    id: {
      type: [Number, String],
      required: true,
    },
  },

  data() {
    return {
      itemData: {},
    }
  },

  computed: {
    ...mapGetters('products', ['getCurrentItem']),
  },

  methods: {
    ...mapActions('products', ['loadItemDataById']),
```

```
  },

  created() {
    this.loadItemDataById(this.id)
  },
}
</script>

<style lang="scss" scoped></style>
```



Необхідно додатково встановити

npm install vee-validate –save

npm install yup --save

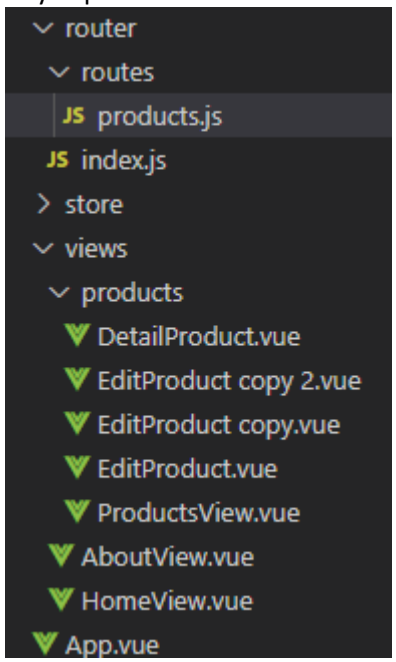
```
<template>
  <products-layout>
    <Form
      v-slot="{ handleReset }"
      :validation-schema="schema"
      :initial-values="product"
      @submit="onSubmit"
    >
      <div>
        <label for="title">Title:</label>
        <Field name="title" type="text" />
        <ErrorMessage name="title" />
      </div>
      <div>
        <label for="price">Price:</label>
        <Field name="price" type="number" />
        <ErrorMessage name="price" />
      </div>
      <button type="submit">Submit</button>
      <button type="button" @click="handleReset">Reset</button>
    </Form>
  </products-layout>
</template>

<script>
import { Form, Field, ErrorMessage } from 'vee-validate'
```

```
import * as yup from 'yup'
import { mapActions, mapGetters } from 'vuex'

const initialProductData = {
  title: '',
  price: null,
}

export default {
  components: {
    Form,
    Field,
    ErrorMessage,
  },

  props: {
    id: {
      type: String,
      required: false,
    },
  },
  data() {
    return {
      product: { ...initialProductData },
      schema: yup.object({
        title: yup.string().required(),
        price: yup.number().required().min(1),
      }),
    }
  },
  computed: {
    ...mapGetters('products', ['getCurrentItem', 'isLoading', 'hasError']),
  },

  watch: {
    getCurrentItem: {
      handler(newValue) {
```

```
      if (newValue) {
        this.product = {
          title: newValue.title,
          price: newValue.price,
        }
        this.updateKey++
      }
    },
    deep: true,
    immediate: true,
  },
},

methods: {
  ...mapActions('products', ['loadItemDataById', 'addItem', 'updateItem']),
  async onSubmit(values, { resetForm }) {
    try {
      if (this.id) {
        await this.updateItem({ itemId: this.id, data: values })
        console.log('Product updated:', values)
      } else {
        await this.addItem(values)
        console.log('Product created:', values)
      }

      resetForm()
      this.$router.push({
        name: 'product-list',
      })
    } catch {
      alert('Щось пішло не так!')
    }
  },
},
created() {
  this.product = { ...initialProductData }
```

```
    if (this.id) {
      this.loadItemDataById(this.id)
    }
  },
}
</script>
```

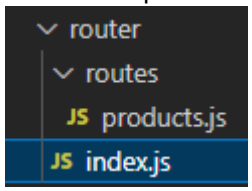
Роути products



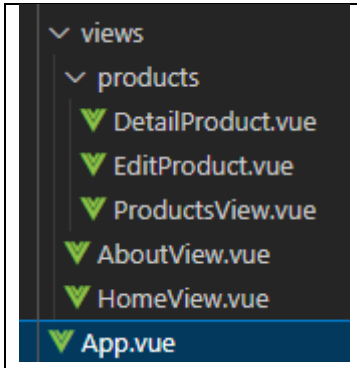

```
import ProductsView from '../../views/products/ProductsView.vue'
import EditProduct from '../../views/products/EditProduct.vue'
import DetailProduct from '../../views/products/DetailProduct.vue'

const routes = [
  {
    path: '/products',
    name: 'products-list',
    redirect: '/products/list',
    children: [
      {
        path: 'list',
        name: 'product-list',
        component: ProductsView,
      },
      {
        path: 'edit/:id?',
        name: 'product-edit',
        component: EditProduct,
        props: true,
      },
      {
        path: ':id',
        name: 'product-detail',
        component: DetailProduct,
        props: true,
      },
    ],
  },
```

```
]

export default routes
```

Основний файл з налаштувнанями роутера




```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import products from './routes/products'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      component: HomeView,
    },
    ...products,
    {
      path: '/about',
      name: 'about',
      // route level code-splitting
      // this generates a separate chunk (About.[hash].js) for this route
      // which is lazy-loaded when the route is visited.
      component: () => import('../views/AboutView.vue'),
    },
  ],
})

export default router
```




```vue
<script setup>
import { RouterLink, RouterView } from 'vue-router'
import HelloWorld from './components/HelloWorld.vue'
</script>

<template>
  <header>
    <div class="wrapper">
      <nav>
        <RouterLink to="/">Home</RouterLink>
        <RouterLink to="/products">Products</RouterLink>
        <RouterLink to="/about">About</RouterLink>
      </nav>
    </div>
  </header>

  <RouterView />
</template>

<style scoped>
header {
  line-height: 1.5;
  max-height: 100vh;
}

.logo {
  display: block;
  margin: 0 auto 2rem;
}

nav {
  width: 100%;
  font-size: 12px;
  text-align: center;
  margin-top: 2rem;
}
```

```
nav a.router-link-exact-active {
  color: var(--color-text);
}

nav a.router-link-exact-active:hover {
  background-color: transparent;
}

nav a {
  display: inline-block;
  padding: 0 1rem;
  border-left: 1px solid var(--color-border);
}

nav a:first-of-type {
  border: 0;
}

@media (min-width: 1024px) {
  header {
    display: flex;
    place-items: center;
    padding-right: calc(var(--section-gap) / 2);
  }

  .logo {
    margin: 0 2rem 0 0;
  }

  header .wrapper {
    display: flex;
    place-items: flex-start;
    flex-wrap: wrap;
  }

  nav {
    text-align: left;
```

	`` margin-left: -1rem; font-size: 1rem; padding: 1rem 0; margin-top: 1rem; } } </style>```