

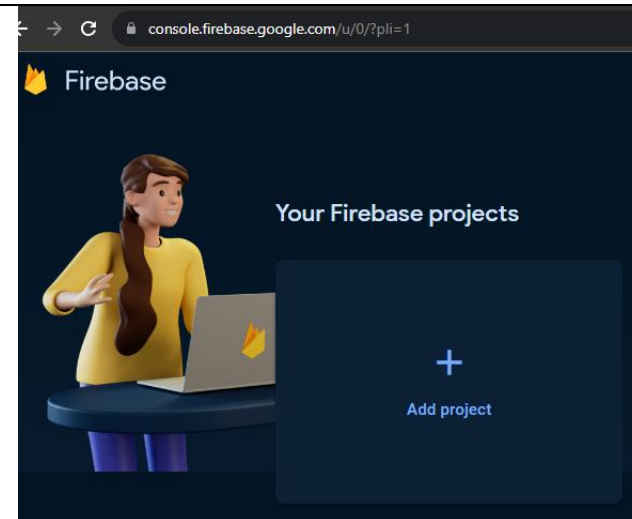
## Vite. Робота з Firestore Database в Firebase

# Add Firebase to your JavaScript project

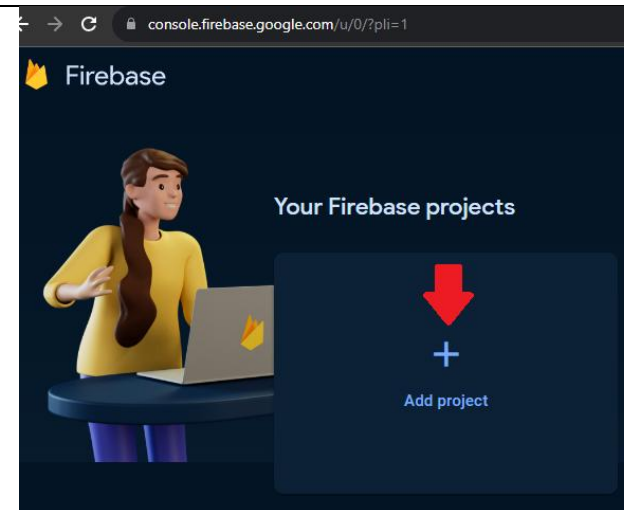
[https://firebase.google.com/docs/web/setup?hl=en&authuser=0&gl=1\\*1as5592\\*ga\\*MjA4NDYzOTg2Ny4xNzAwNjczMzM5\\*ga\\_CW55HF8NVT\\*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA](https://firebase.google.com/docs/web/setup?hl=en&authuser=0&gl=1*1as5592*ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA).

Переходимо у консоль firebase

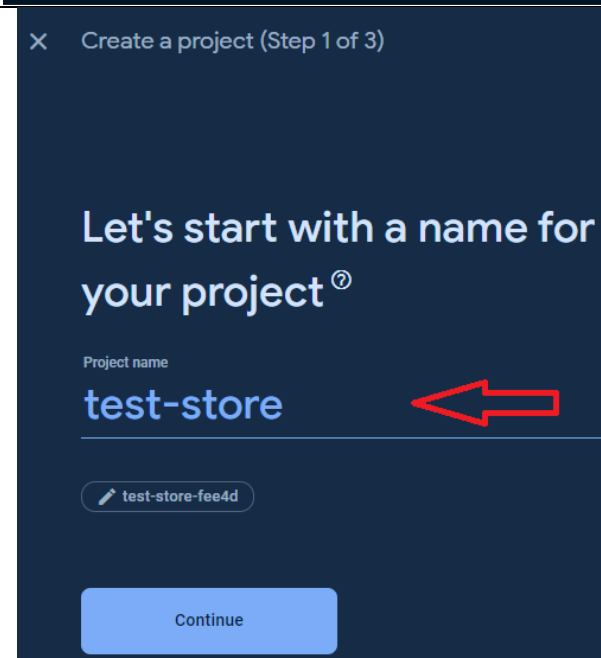
<https://console.firebase.google.com/u/0/?pli=1>



Створюємо проект в Firebase



Задаємо якусь назву проєкта



Відключаємо аналітику

×

Create a project (Step 2 of 2)

Google Analytics

for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

×

A/B testing ⓘ

×

Crash-free users ⓘ

×

User segmentation & targeting across Firebase products ⓘ

×

Event-based Cloud Functions triggers ⓘ

×

Free unlimited reporting ⓘ

Enable Google Analytics for this project

Recommended

↑

Previous

Create project

Створюємо проєкт

Creating your project... Please wait...

test-store

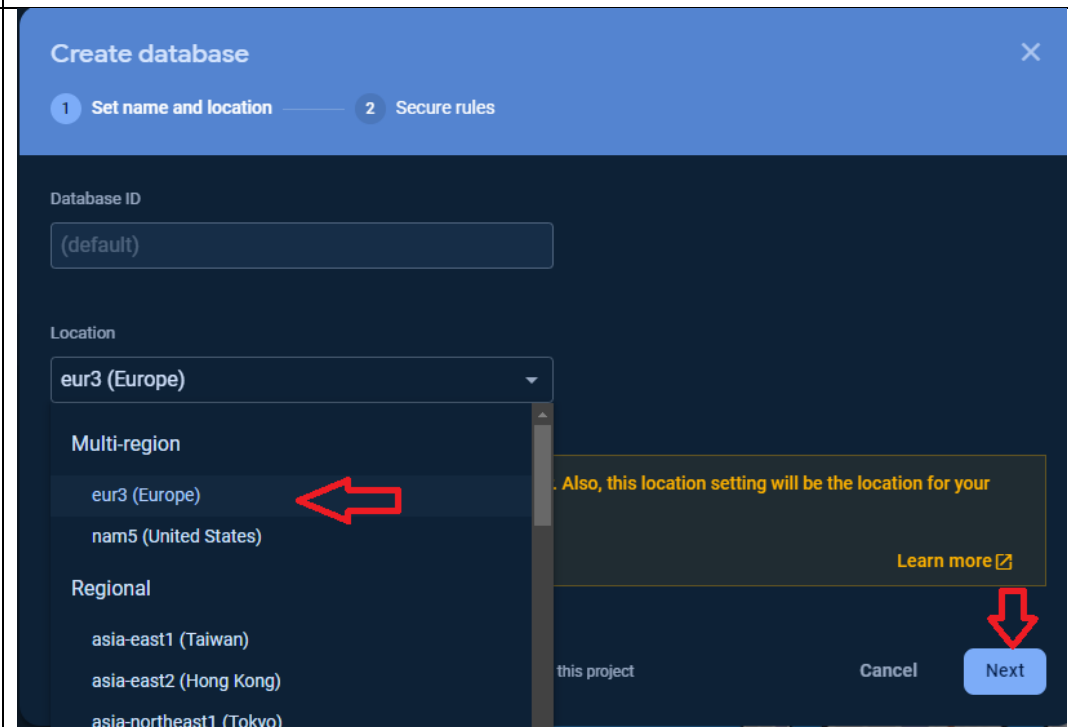
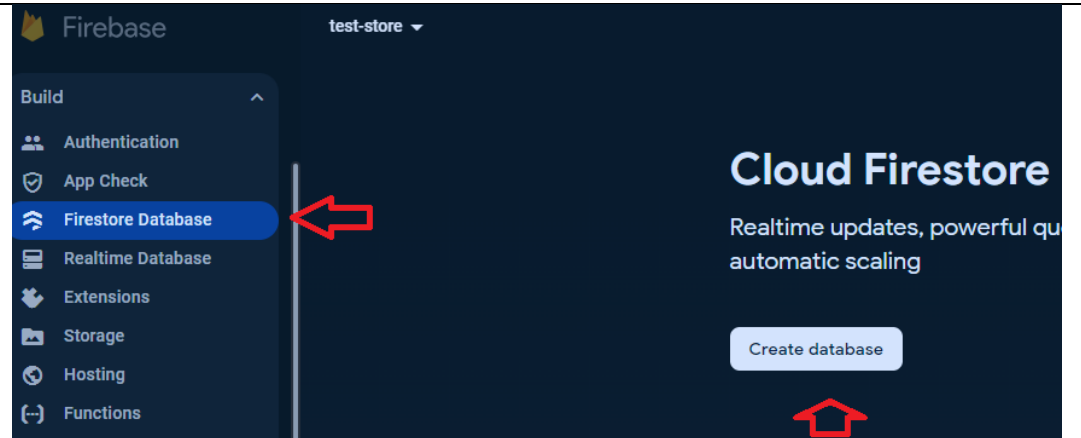
test-store

✓

Your new project is ready

Continue

Додаємо Firestore Database



	<div><div>Create database</div><div><div>✓ Set name and location</div><div>2 Secure rules</div></div><div>After you define your data structure, you will need to write rules to secure your data. <a href="#">Learn more</a></div><div><div><div><div></div><div>Start in production mode</div></div><div>Your data is private by default. Client read/write access will only be granted as specified by your security rules.</div></div><div><div><div></div><div>Start in test mode</div></div><div>Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.</div></div></div><div><pre>rules_version = '2';  service cloud.firestore {   match /databases/{database}/documents {     match /{document=**} {       allow read, write: if         request.time &lt; timestamp.date(2023, 12, 23);     }   } }</pre><div><div>!</div><div>The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days</div></div></div><div><div>Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project</div><div>Cancel</div><div>Enable</div></div></div>
	<div><div><div>Home</div><div>More in Google Cloud</div></div><div><div>(default)</div><div>+ Start collection</div></div></div>
Створюємо колекцію даних	<div><div><div>Home</div><div>More in Google Cloud</div></div><div><div>(default)</div><div>+ Start collection</div></div></div>

Задаємо ім'я колекції

Start a collection

1 Give the collection an ID — 2 Add its first document

Parent path

/

Collection ID ?

products

Cancel Next

1)Визначаємо спосіб задання id як автоматичний  
2)Додаємо документ (задаємо поля документа(назва, тип, значення)

Можемо задати вручну декілька документів

Start a collection

✓ Give the collection an ID — 2 Add its first document

Document parent path

/products

Document ID ?

Auto-ID

1 Required

Field	Type	Value
title	string	
price	number	
imgSrc	string	

Cancel Save

Add a document

Parent path

/products

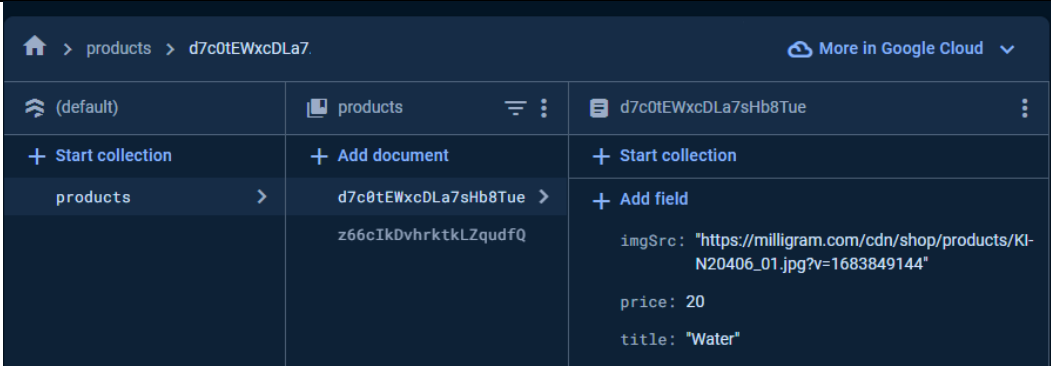
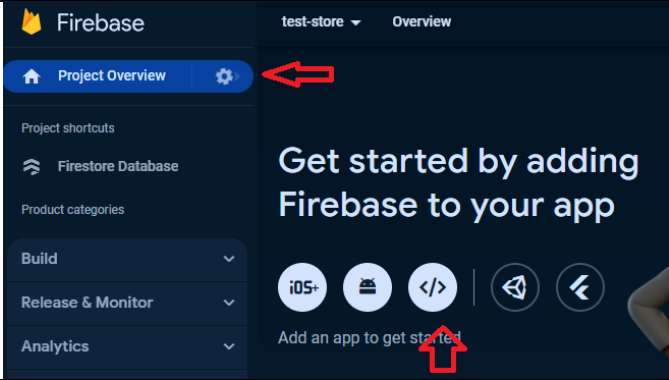
Document ID ?

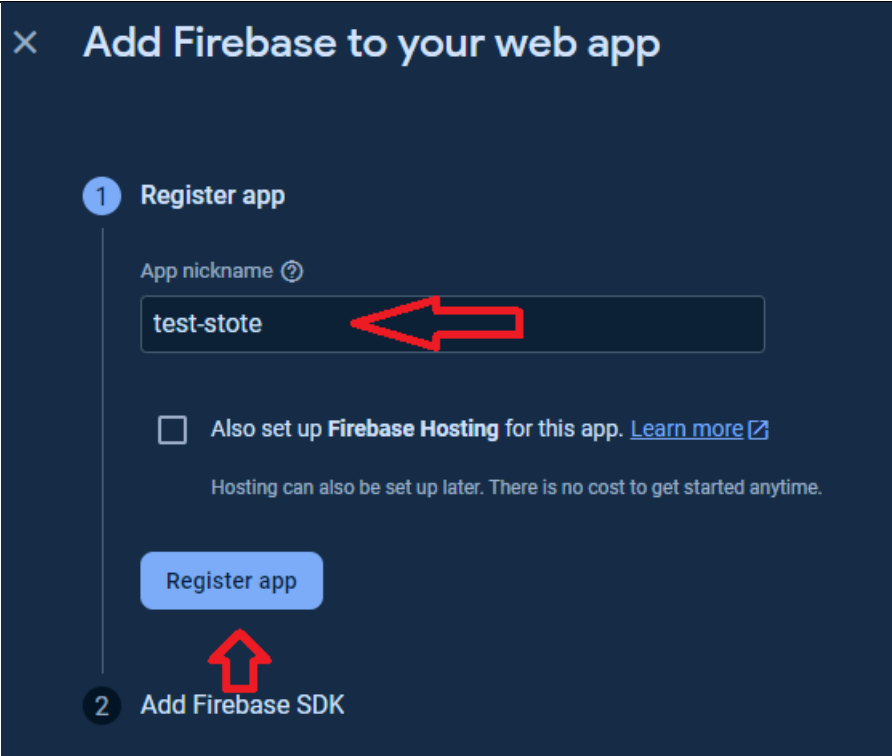
Auto-ID

1 Required

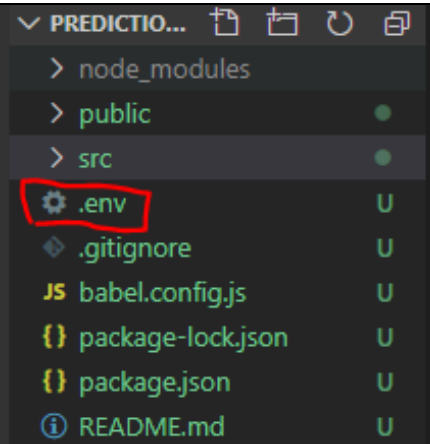
Field	Type	Value
title	string	Coffee
price	number	450
imgSrc	string	https://img.freep

Cancel Save

		
Додаємо firebase до проєкта		

<p>Задаємо назву і реєструємо</p>	
<p>Встановлюємо необхідні модулі</p> <p><a href="https://firebase.google.com/docs/web/setup?hl=en&amp;authuser=0&amp;_gl=1*_as5592*_ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*_ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA">https://firebase.google.com/docs/web/setup?hl=en&amp;authuser=0&amp;_gl=1*_as5592*_ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*_ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA</a>.</p>	<p><code>npm install firebase</code></p>
<p>Створюємо файл .env</p>	<p>Додаємо наш додаток в Firebase проект</p> <p>дані з Вашого проекту</p> <p><code>_APP_FIREBASE_AUTH_DOMAIN=</code> дані з Вашого проекту</p> <p><code>=</code> дані з Вашого проекту</p> <p><code>_APP_FIREBASE_PROJECT_ID=</code> дані з Вашого проекту</p> <p><code>_APP_FIREBASE_STORAGE_BUCKET=</code> дані з Вашого проекту</p> <p><code>_APP_FIREBASE_MESSAGE_SENDER_ID=</code> дані з Вашого проекту</p> <p><code>_APP_FIREBASE_APP_ID=</code> дані з Вашого проекту</p>





<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>

## Копіюємо з сайту

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

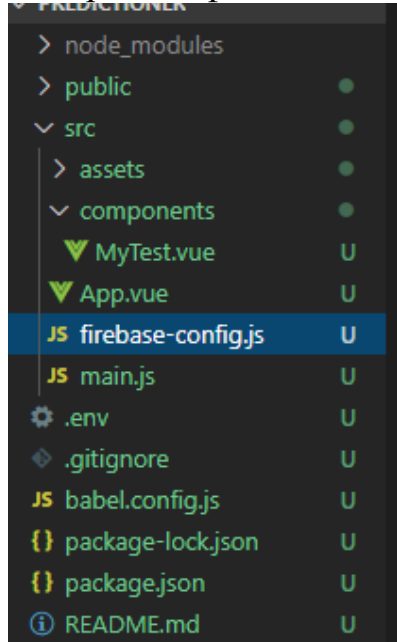
```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyA...",
  authDomain: "...",
  projectId: "...",
  storageBucket: "...",
  messagingSenderId: "...",
  appId: "..."
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```



## 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`)
```

### Зчитування даних

[https://firebase.google.com/docs/web/setup?hl=en&authuser=0&\\_gl=1\\*1as5592\\*\\_ga\\*MjA4NDYzOTg2Ny4xNzAwNjczMzM5\\*\\_ga\\_CW55HF8NVT\\*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA](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,
```

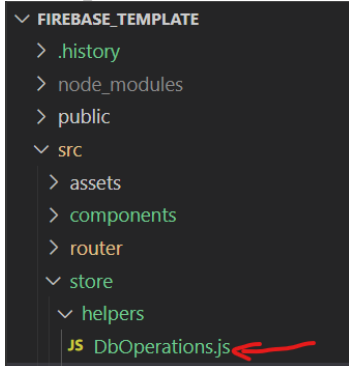
<a href="https://firebase.google.com/docs/reference/js/firestore_lite.md?hl=ru#getdocs">https://firebase.google.com/docs/reference/js/firestore_lite.md?hl=ru#getdocs</a>	<pre>         ...doc.data(),     })     })     resolve(list)   })   .catch((error) =&gt; {     reject(error)   }) </pre>
<p>Додавання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

```

        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

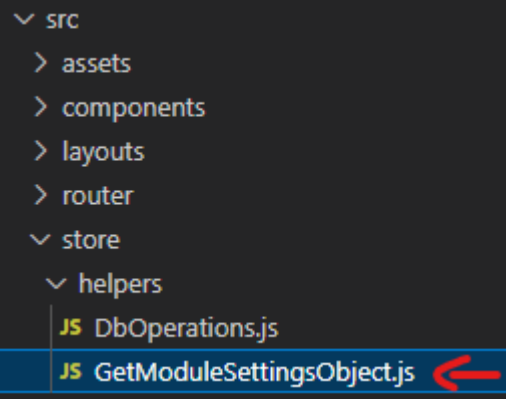
```

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

```
└─ store
  └─ helpers
    └─ JS DbOperations.js
    └─ JS GetModuleSettingsObject.js
    └─ JS index.js ←
```

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

## Основний layout

```
└─ src
  └─ > assets
  └─ > components
  └─ layouts
    └─ ▼ MainLayout.vue
    └─ ▼ ProductsLayout.vue
```

```
<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>
```

	<pre>&lt;style lang="scss" scoped&gt;&lt;/style&gt;</pre>
<p>Для сторінок products</p> <div data-bbox="87 228 472 507"> <ul style="list-style-type: none"> <li>src <ul style="list-style-type: none"> <li>assets</li> <li>components</li> <li>layouts <ul style="list-style-type: none"> <li>MainLayout.vue</li> <li>ProductsLayout.vue</li> </ul> </li> </ul> </li> </ul> </div>	<pre> &lt;template&gt;   &lt;main-layout :is-loading="isLoading" :has-error="hasError"&gt;     &lt;slot&gt;&lt;/slot&gt;   &lt;/main-layout&gt; &lt;/template&gt;  &lt;script&gt; import { mapGetters } from 'vuex'  export default {   name: 'ProductsLayout',   computed: {     ...mapGetters('products', ['getItemsList', 'isLoading', 'hasError']),   }, } &lt;/script&gt;  &lt;style lang="scss" scoped&gt;&lt;/style&gt; </pre>
<div data-bbox="87 962 414 1273"> <ul style="list-style-type: none"> <li>views <ul style="list-style-type: none"> <li>products <ul style="list-style-type: none"> <li>DetailProduct.vue</li> <li>EditProduct.vue</li> <li>ProductsView.vue</li> <li>AboutView.vue</li> <li>HomeView.vue</li> </ul> </li> </ul> </li> </ul> </div>	<pre> &lt;template&gt;   &lt;div class="d-flex"&gt;     &lt;div&gt;       &lt;label&gt;         Title         &lt;input type="text" v-model.lazy="queryOptData.productTitle" /&gt;       &lt;/label&gt;     &lt;/div&gt;     &lt;div&gt;       &lt;label&gt;         Order         &lt;select v-model="queryOptData.orderType"&gt;           &lt;option value="asc"&gt;За зростанням&lt;/option&gt;           &lt;option value="desc"&gt;За спаданням&lt;/option&gt;         &lt;/select&gt;       &lt;/label&gt;     &lt;/div&gt;   &lt;/div&gt; </pre>

```

        </select>
      </label>
    </div>
  </div>
  <hr />
  <h2>Список продуктів</h2>
  <products-layout>
    <div v-for="item in getItemList" :key="item.id" class="d-flex">
      <div>
        <router-link
          :to="{
            name: 'product-detail',
            params: {
              id: item.id,
            },
          }"
        >{{ item.title }}</router-link>
        <br />
        <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>
  </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>
```

```
▼ views
  ▼ products
    ▼ DetailProduct.vue
    ▼ EditProduct.vue
    ▼ ProductsView.vue
  ▼ AboutView.vue
  ▼ HomeView.vue
```

```
<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']),
  },
}
```

	<pre>     },      created() {       this.loadItemDataById(this.id)     },   } &lt;/script&gt;  &lt;style lang="scss" scoped&gt;&lt;/style&gt; </pre>
<div data-bbox="89 496 414 807"> <ul style="list-style-type: none"> <li>views           <ul style="list-style-type: none"> <li>products               <ul style="list-style-type: none"> <li>DetailProduct.vue</li> <li><b>EditProduct.vue</b></li> <li>ProductsView.vue</li> <li>AboutView.vue</li> <li>HomeView.vue</li> </ul> </li> </ul> </li> </ul> </div> <p>Необхідно додатково встановити</p> <p>npm install vee-validate --save</p> <p>npm install yup --save</p>	<pre> &lt;template&gt;   &lt;products-layout&gt;     &lt;Form       v-slot="{ handleReset }"       :validation-schema="schema"       :initial-values="product"       @submit="onSubmit"     &gt;       &lt;div&gt;         &lt;label for="title"&gt;Title:&lt;/label&gt;         &lt;Field name="title" type="text" /&gt;         &lt;ErrorMessage name="title" /&gt;       &lt;/div&gt;       &lt;div&gt;         &lt;label for="price"&gt;Price:&lt;/label&gt;         &lt;Field name="price" type="number" /&gt;         &lt;ErrorMessage name="price" /&gt;       &lt;/div&gt;       &lt;button type="submit"&gt;Submit&lt;/button&gt;       &lt;button type="button" @click="handleReset"&gt;Reset&lt;/button&gt;     &lt;/Form&gt;   &lt;/products-layout&gt; &lt;/template&gt;  &lt;script&gt; import { Form, Field, ErrorMessage } from 'vee-validate' </pre>

```
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)
      }
    }

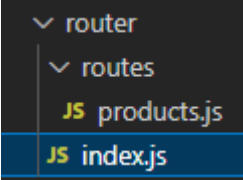
    resetForm()
    this.$router.push({
      name: 'product-list',
    })
  } catch {
    alert('Щось пішло не так!')
  }
},
},
created() {
  this.product = { ...initialProductData }
}
```

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

#### Роути products

```
└─ router  
  └─ routes  
    └─ JS products.js  
    └─ JS index.js  
  > store  
  └─ views  
    └─ products  
      └─ ▼ DetailProduct.vue  
      └─ ▼ EditProduct copy 2.vue  
      └─ ▼ EditProduct copy.vue  
      └─ ▼ EditProduct.vue  
      └─ ▼ ProductsView.vue  
      └─ ▼ AboutView.vue  
      └─ ▼ HomeView.vue  
      └─ ▼ App.vue
```

```
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,  
      },  
    ],  
  },  
],  
},
```

	<pre>]  export default routes</pre>
<p>Основний файл з налаштуваннями роутера</p> 	<pre>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: () =&gt; import('../views/AboutView.vue'),     },   ], })  export default router</pre>

- views
  - products
    - DetailProduct.vue
    - EditProduct.vue
    - ProductsView.vue
  - AboutView.vue
  - HomeView.vue
  - App.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>
```