

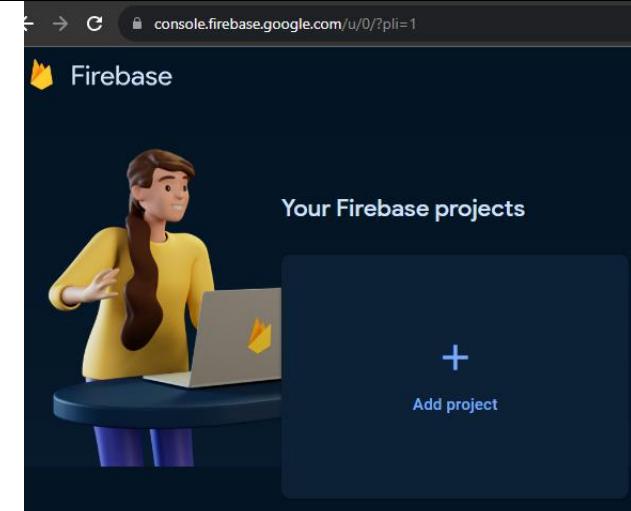
Vite. Робота з Firestore Database в Firebase

Add Firebase to your JavaScript project

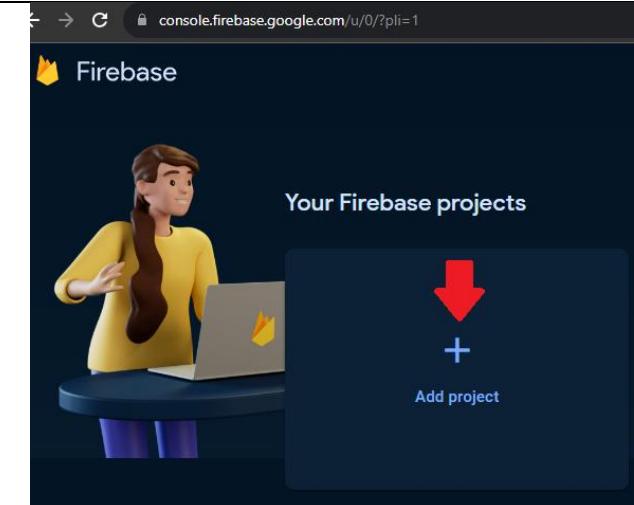
https://firebase.google.com/docs/web/setup?hl=en&authuser=0& gl=1*1as5592* ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5* ga CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA.

Переходимо у консоль firebase

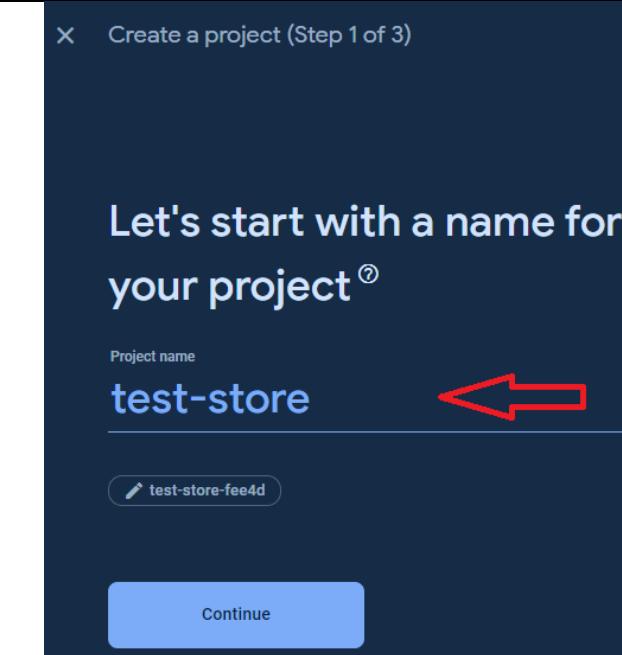
<https://console.firebaseio.google.com/u/0/?pli=1>



Створюємо проект в Firebase



Задаємо якусь назву проекта



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 ⓘ

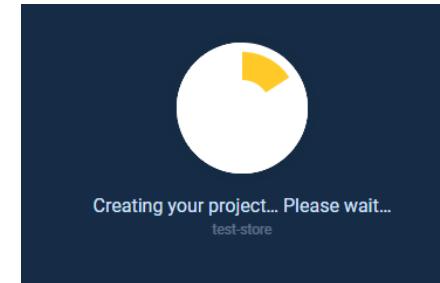
Enable Google Analytics for this project
Recommended



Previous

Create project

Створюємо проект



test-store

Your new project is ready

Continue

Додаємо Firestore Database

The screenshot shows the Firebase console interface. On the left, there's a sidebar with a tree view under the 'Build' category. The 'Firestore Database' item is highlighted with a blue background and has a red arrow pointing to it from the top right. To the right of the sidebar is a dark header bar with the text 'test-store' and a dropdown arrow. Below the header is a banner for 'Cloud Firestore' with the text 'Realtime updates, powerful query language, automatic scaling'. A 'Create database' button and a house icon are also visible. The main content area is titled 'Create database' and shows two steps: '1 Set name and location' and '2 Secure rules'. Under 'Database ID', '(default)' is selected. Under 'Location', 'eur3 (Europe)' is selected, with a red arrow pointing to it from the bottom left. A note says 'Also, this location setting will be the location for your data in this project.' Below the location dropdown, there are sections for 'Multi-region' and 'Regional' locations. In the bottom right corner of the modal, there are 'Cancel' and 'Next' buttons, with a red arrow pointing to the 'Next' button.

Firebase

test-store ▾

Build

- Authentication
- App Check
- Firestore Database**
- Realtime Database
- Extensions
- Storage
- Hosting
- Functions

Cloud Firestore

Realtime updates, powerful query language, automatic scaling

Create database

eur3 (Europe)

Multi-region

- eur3 (Europe)
- nam5 (United States)

Regional

- asia-east1 (Taiwan)
- asia-east2 (Hong Kong)
- asia-northeast1 (Tokyo)

Cancel

Next

Create database

Set name and location

Secure rules

After you define your data structure, **you will need to write rules to secure your data.**

[Learn more](#)

Start in **production mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

Start in **test mode**

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.

```
rules_version = '2';

service cloud.firestore {
    match /databases/{database}/documents {
        match /{document=**} {
            allow read, write: if
                request.time < timestamp.date(2023, 12, 23);
        }
    }
}
```

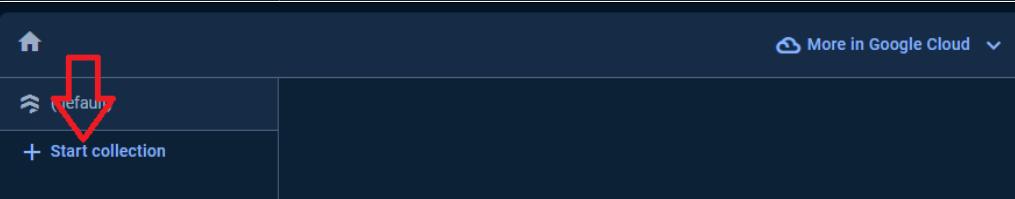
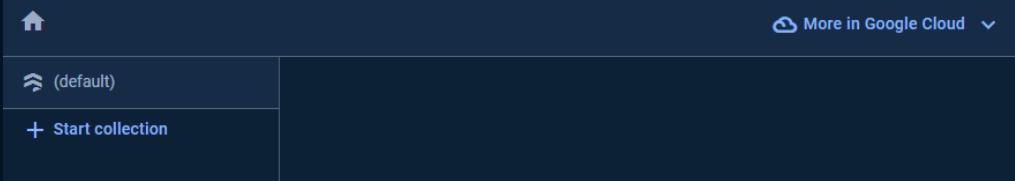
! The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project

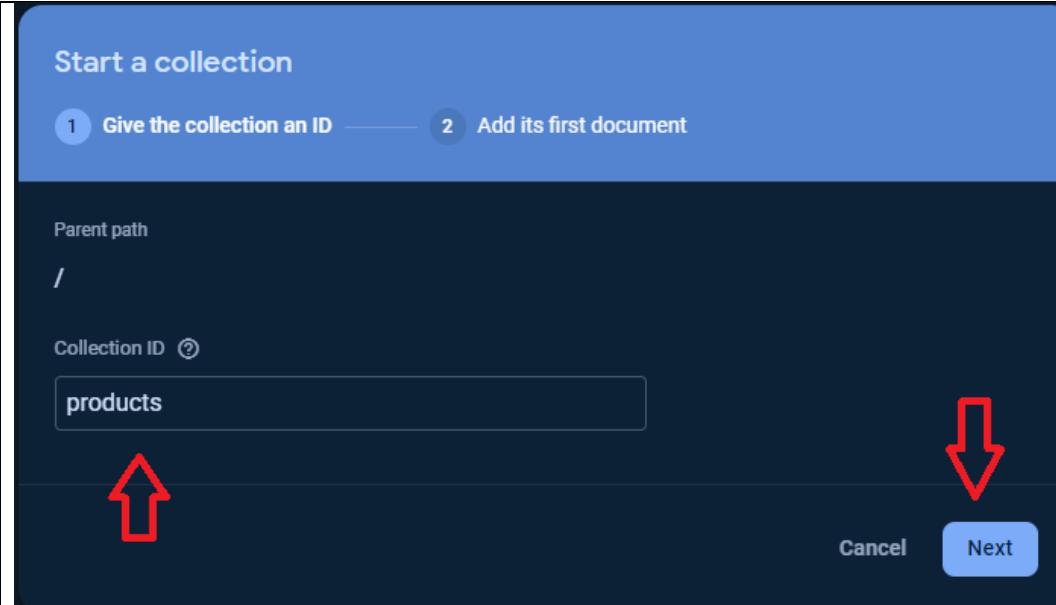
[Cancel](#)

[Enable](#)

Створюємо колекцію даних



Задаємо ім'я колекції

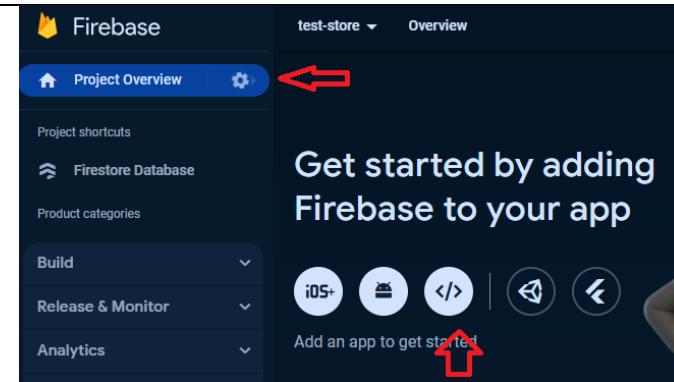


- 1) Визначаємо спосіб задання id як автоматичний
- 2) Додаємо документ (задаємо поля документа(назва, тип, значення))

Можемо задати вручну декілька документів

More in Google Cloud
(default)
+ Start collection
products
products
d7c0tEWxcDLa7sHb8Tue
+ Add document
d7c0tEWxcDLa7sHb8Tue >
z66cIkDvhrtkLZqudfQ
+ Start collection
+ Add field
imgSrc: "https://milligram.com/cdn/shop/products/KI-N20406_01.jpg?v=1683849144"
price: 20
title: "Water"

Додаємо firebase до проекта



Задаємо назву і реєструємо

X Add Firebase to your web app

1 Register app

App nickname ?

test-stote



Also set up **Firebase Hosting** for this app. [Learn more ↗](#)

Hosting can also be set up later. There is no cost to get started anytime.

Register app

2 Add Firebase SDK



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

✓ PREDICTIO...	File	Folder	Open	Save
> node_modules				
> public				
> src				
⚙️ .env	U			
↳ .gitignore	U			
JS babel.config.js	U			
{ } package-lock.json	U			
{ } package.json	U			
ⓘ README.md	U			

[#%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%BD%D8%D1%8F](https://cli.vuejs.org/ru/guide/mode-and-env.html)

Копіюємо з сайту

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

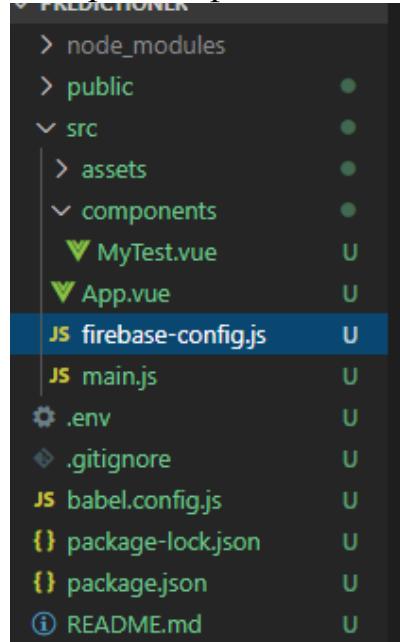
```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: [REDACTED],
  authDomain: [REDACTED],
  projectId: [REDACTED],
  storageBucket: [REDACTED],
  messagingSenderId: [REDACTED],
  appId: [REDACTED]
};

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



Створюємо файл налаштувань firebase



```
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
```

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

----- 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.

```
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(),
      })
    })
  .then(list =>
    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

```
▽ FIREBASE TEMPLATE
  > .history
  > node_modules
  > public
  ▽ src
    > assets
    > components
    > router
    ▽ store
      > helpers
        ▄ DbOperations.js ←
```

```
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.fieldPath,
        options.orderBy.direction
      )
    )
  return queryOpt
}
```

```
        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) => {
```

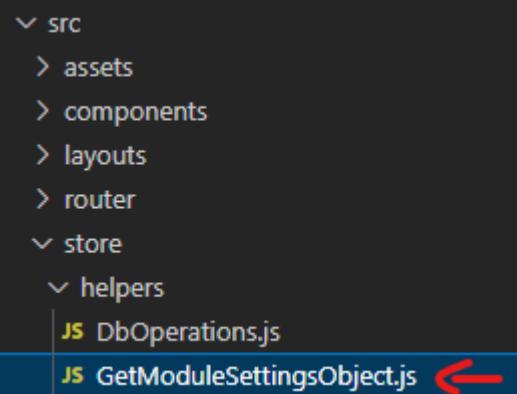
```
        reject(error)
    })
})
}
loadFilteredData({ fieldTitle, compareOperator, valueToCompare }, options)
{
    const filter = where(fieldTitle, compareOperator, valueToCompare)
    const queryOpt = this.getQueryOptions(options, filter)

    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 {
        namespaces: 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('setItemList', 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
    .loadItemId(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><style lang="scss" scoped></style> <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></pre>
<p>Для сторінок products</p> <ul style="list-style-type: none"> src <ul style="list-style-type: none"> > assets > components layouts <ul style="list-style-type: none"> >MainLayout.vue ProductsLayout.vue 	<pre><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 спаданням</option> </select> </label> </div> </div> <table border="1"> <thead> <tr> <th>Product Name</th> <th>Category</th> <th>Price (USD)</th> </tr> </thead> <tbody> <tr> <td>Laptop X12 Pro</td> <td>Electronics</td> <td>1200</td> </tr> <tr> <td>Smartphone S10</td> <td>Electronics</td> <td>800</td> </tr> <tr> <td>Monitor M24</td> <td>Electronics</td> <td>400</td> </tr> <tr> <td>Headphones H500</td> <td>Electronics</td> <td>150</td> </tr> </tbody> </table> <button @click="fetchProductsList(queryOptData)">Search</button> <div> <div> Cart (0) </div> <div> User Profile </div> </div> <Footer> <div> TechMart </div> <div> Home About Products Cart Profile </div> </Footer> </pre>

```
        </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;
}
```

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

```
        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']),
  }
}
</script>
```

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

Необхідно додатково встановити

npm install vee-validate --save

npm install yup --save

```
},
created() {
  this.loadItemDataById(this.id)
},
</script>
<style lang="scss" scoped></style>
<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>
```

Роуты products

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

```
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 налаштуваннями роутера

```
router
  routes
    products.js
index.js
```

```
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
```

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>
```