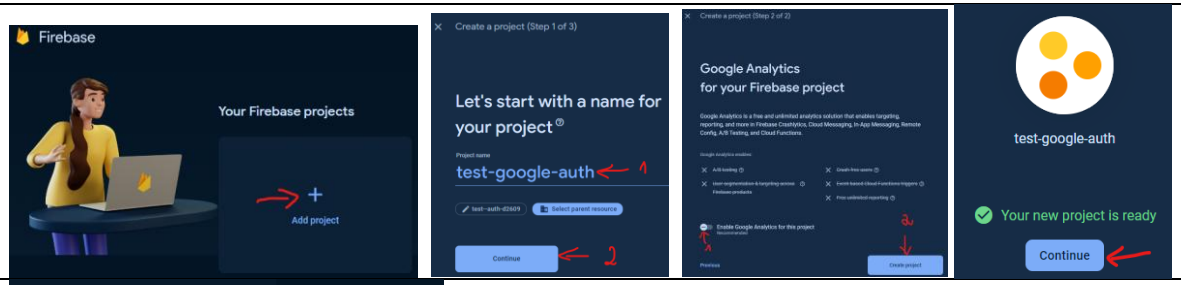
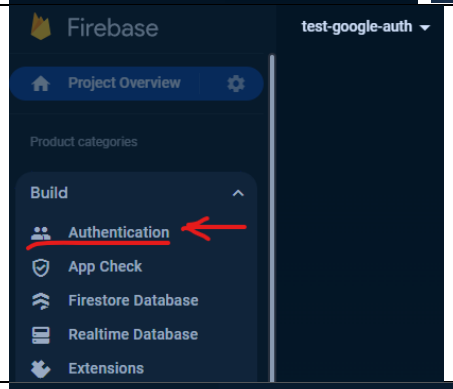
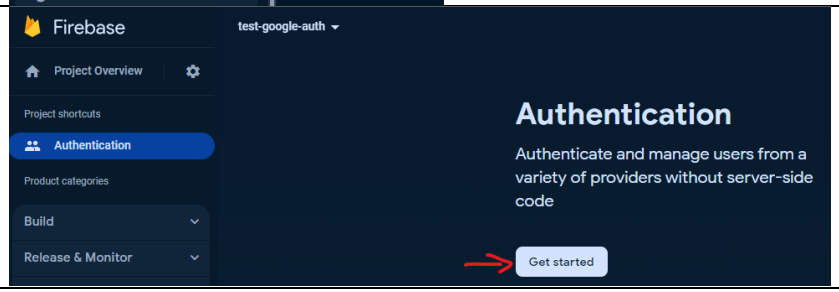
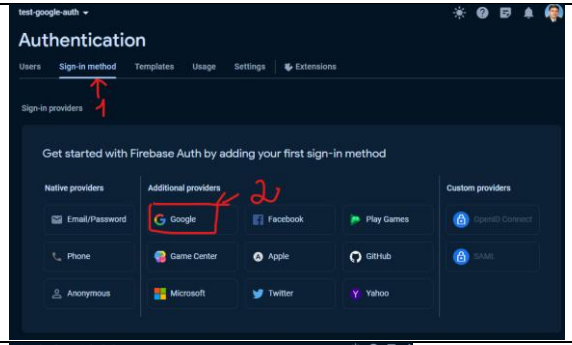
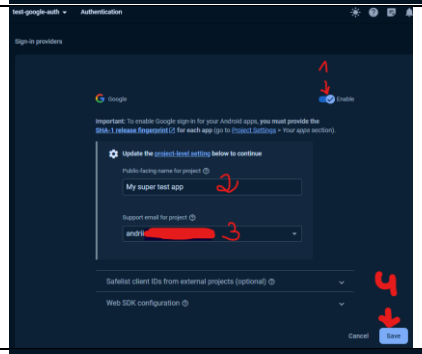
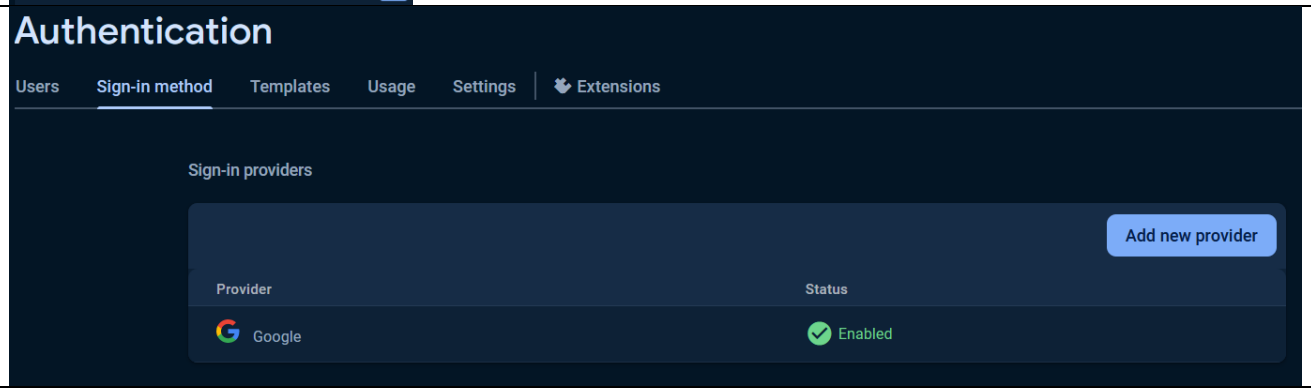
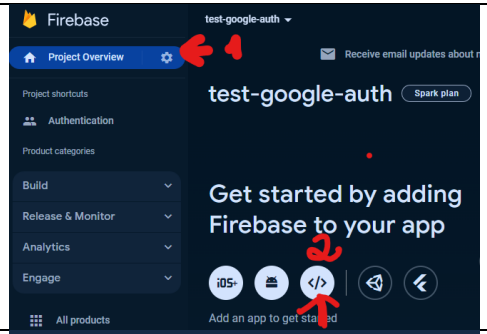
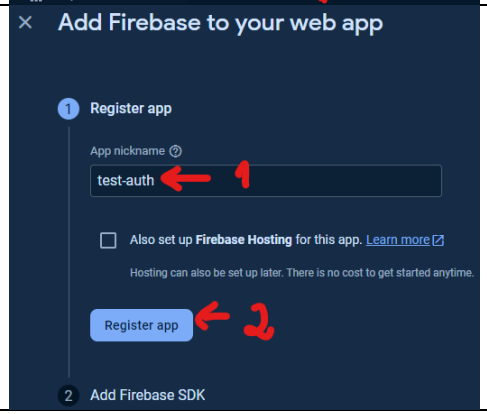
Google oAuth Firebase

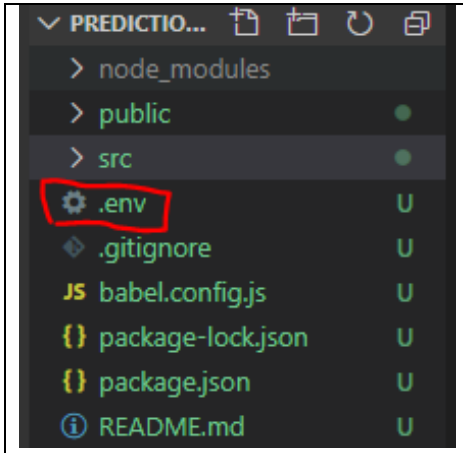
Проект повинен бути створений, ініціалізований та розміщуватися на хості.

Переходимо у Firabase Console https://console.firebase.google.com	
Створюємо проєкт test-google-auth	
Переходимо на вкладку Authentication	
Переходимо за Get Started	

У вкладці “Sign-in method” обираємо Google	
1)Обираємо Enable 2)задаємо назву додатку, яку будуть бачити користувачі після авторизації 3)задаємо електронну пошту додатку та нажимаємо Save	
У нас з’явився Sign-in provider	

Після того як у Firebase створили google провайдера, можемо його використати у нашому проекті.

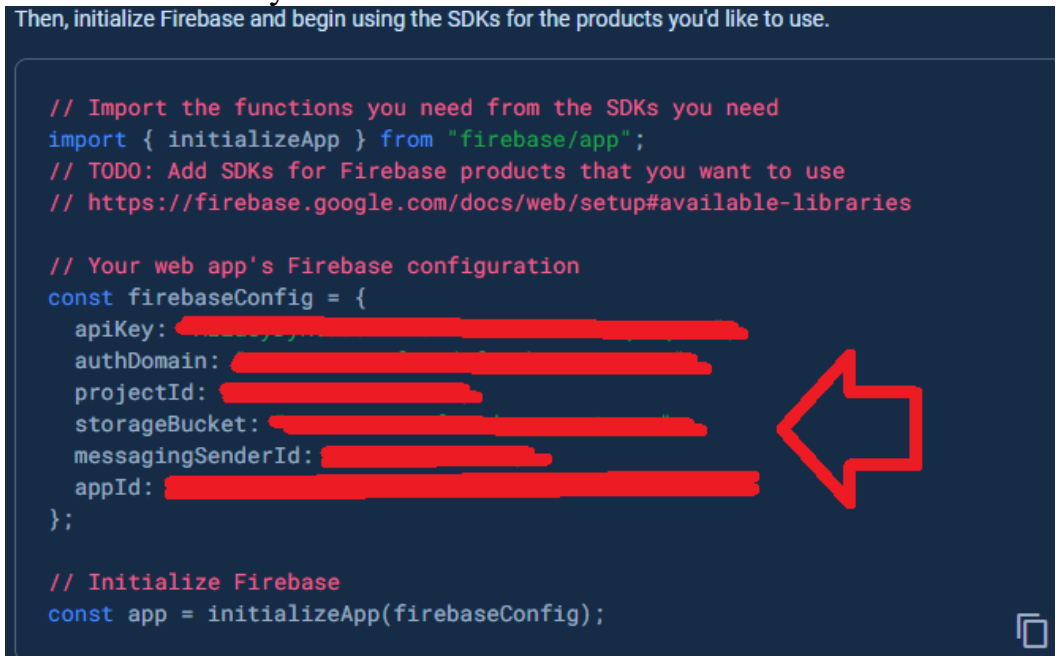
Додаємо firebase до проєкта	
Задаємо назву і реєструємо	
Встановлюємо необхідні модулі https://firebase.google.com/docs/web/setup?hl=en&authuser=0&_gl=1*1as5592*_ga*MjA4NDYzOTg2Ny4xNzAwNjczMzM5*_ga_CW55HF8NVT*MTcwMDczNDY2My40LjEuMTcwMDczNjIwOS42MC4wLjA.	`npm install firebase`
Створюємо файл .env	Додаємо наш додаток в Firebase проект //----- з **CLI** дані з Вашого проекту VUE_APP_FIREBASE_AUTH_DOMAIN= дані з Вашого проекту = дані з Вашого проекту VUE_APP_FIREBASE_PROJECT_ID= дані з Вашого проекту VUE_APP_FIREBASE_STORAGE_BUCKET= дані з Вашого проекту VUE_APP_FIREBASE_MESSAGE_SENDER_ID= дані з Вашого проекту VUE_APP_FIREBASE_APP_ID= дані з Вашого проекту //----- з **VITE**



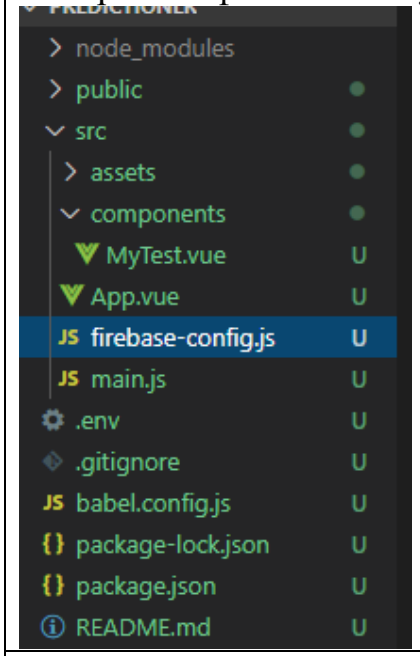
https://cli.vuejs.org/ru/guide/mode-and-env.html#%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5-%D0%BE%D0%BA%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F

```
VITE_FIREBASE_API_KEY = дані з Вашого проекту
VITE_FIREBASE_AUTH_DOMAIN=  дані з Вашого проекту
VITE_FIREBASE_DATABASE_URL= дані з Вашого проекту
VITE_FIREBASE_PROJECT_ID= дані з Вашого проекту
VITE_FIREBASE_STORAGE_BUCKET= дані з Вашого проекту
VITE_FIREBASE_MESSAGE_SENDER_ID= дані з Вашого проекту
VITE_FIREBASE_APP_ID= дані з Вашого проекту
```

Копіюємо з сайту



Створюємо файл налаштувань firebase




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

  const firebaseConfig = {
    apiKey: process.env.VUE_APP_FIREBASE_API_KEY,
    authDomain: process.env.VUE_APP_FIREBASE_AUTH_DOMAIN,
    projectId: process.env.VUE_APP_FIREBASE_PROJECT_ID,
    storageBucket: process.env.VUE_APP_FIREBASE_STORAGE_BUCKET,
    messagingSenderId: process.env.VUE_APP_FIREBASE_MESSAGE_SENDER_ID,
    appId: process.env.VUE_APP_FIREBASE_APP_ID
  };

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

У файлі конфігурації додаємо ініціалізацію засобів авторизації:

```
import {getAuth} from 'firebase/auth';
```

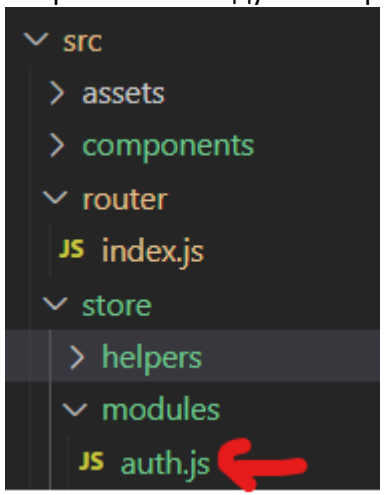
```
export const auth = getAuth(app);
```

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

const firebaseConfig = {
    apiKey: process.env.VUE_APP_FIREBASE_API_KEY,
    authDomain: process.env.VUE_APP_FIREBASE_AUTH_DOMAIN,
    projectId: process.env.VUE_APP_FIREBASE_PROJECT_ID,
    storageBucket: process.env.VUE_APP_FIREBASE_STORAGE_BUCKET,
    messagingSenderId: process.env.VUE_APP_FIREBASE_MESSAGE_SENDER_ID,
    appId: process.env.VUE_APP_FIREBASE_APP_ID,
}

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

---------- для VITE

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

const firebaseConfig = {
  apiKey: import.meta.env.VUE_APP_FIREBASE_API_KEY,
  authDomain: import.meta.env.VUE_APP_FIREBASE_AUTH_DOMAIN,
  projectId: import.meta.env.VUE_APP_FIREBASE_PROJECT_ID,
  storageBucket: import.meta.env.VUE_APP_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: import.meta.env.VUE_APP_FIREBASE_MESSAGE_SENDER_ID,
  appId: import.meta.env.VUE_APP_FIREBASE_APP_ID,
}

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

Розробляємо модуль авторизації

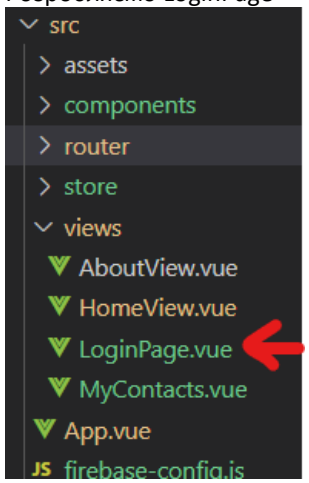



```
import { auth } from '@/firebase-config.js'
import { GoogleAuthProvider, signInWithPopup, signInWithCredential, signOut } from
'firebase/auth'

export default {
    namespaced: true,
    state: {
        user: null,

        loading: false,
        error: null,
    },
    getters: {
        getUser: (state) => state.user,
    },
    mutations: {
        setUser(state, user) {
            state.user = user
        },
```

```
        setLoading(state, loading) {
            state.loading = loading
        },
        setError(state, error) {
            state.error = error
        },
    },
    actions: {
        async saveLoginUserData({ commit }, loginResult) {
            console.log('loginResult')
            console.log(loginResult)
            //--------- user data -------
            const user = loginResult?.user // об'єкт користувача
            commit('setUser', user)
            //user.displayName - ім'я акаунту
            //user.photoURL - аватар акаунту
            //user.email - електронна адреса акаунту

            //--------- user access token -------
            // This gives you a Google Access Token. You can use it to access the
Google API.
            let credential = GoogleAuthProvider.credentialFromResult(loginResult)

            localStorage.setItem('authCredential', JSON.stringify(credential))
        },
        loginWithGoogle({ commit, dispatch }) {
            const provider = new GoogleAuthProvider()
            signInWithPopup(auth, provider)
                .then((loginResult) => {
                    dispatch('saveLoginUserData', loginResult)
                })
                .catch((error) => {
                    commit('setError', error)
                })
        },
        async loginWithCredential({ commit, dispatch }) {
            return new Promise((resolve, reject) => {
```

```
                let credential = localStorage.getItem('authCredential')

                if (credential) {
                    credential = JSON.parse(credential)

                    credential = GoogleAuthProvider.credential(credential.idToken)

                    signInWithCredential(auth, credential)
                        .then((loginResult) => {
                            dispatch('saveLoginUserData', loginResult)
                            resolve(loginResult)
                        })
                        .catch((error) => {
                            console.log(error)
                            commit('setError', error)
                            // Handle Errors here.
                            //var errorCode = error.code
                            //var errorMessage = error.message
                            // The email of the user's account used.
                            //var email = error.email
                            reject(false)
                        })
                } else resolve(false)
            })
        },
        logout({ commit }) {
            signOut(auth)
                .then(() => {
                    localStorage.removeItem('authCredential')
                    commit('setUser', null)
                })
                .catch((error) => {
                    commit('setError', error)
                })
        },
    },
}
```

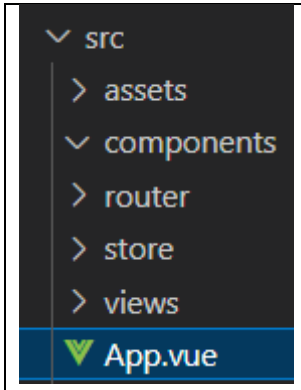
Розробляємо LoginPage




```vue
<template>
    <div>
        <h1>Увійдіть для користування сервісами</h1>
        <button @click="onLogin">Увійти</button>
    </div>
</template>

<script>
import { mapActions } from 'vuex'

export default {
    name: 'LoginPage',

    methods: {
        ...mapActions('auth', ['loginWithGoogle']),
        async onLogin() {
            try {
                await this.loginWithGoogle()
                this.$router.push({
                    name: 'home',
                })
            } catch (error) {
                alert(error.message)
            }
        },
    },
}
</script>

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

Аналізуємо чи авторизований користувач і відображаємо

```vue
<template>
  <div class="container">
```



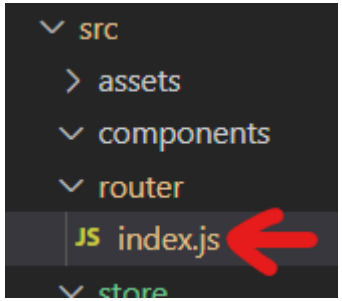
. . . . . .

. . . . . .

```
    <nav>
      <router-link to="/"> Home </router-link> |
      <router-link to="/about"> About </router-link>
      <router-link to="/contacts"> My Contacts </router-link>
    </nav>
    <div>
      <span v-if="getUser">
        <span>{{ getUser.displayName }}</span>
        <img :src="getUser.photoURL" />
        <button @click="onLogout">Logout</button>
      </span>
      <button v-else @click="onLogin">Login</button>
    </div>
  </div>
  <router-view />
</template>
<script>
import { mapGetters, mapActions } from 'vuex'

export default {
  computed: {
    ...mapGetters('auth', ['getUser']),
  },

  methods: {
    ...mapActions('auth', ['loginWithGoogle', 'logout']),
    onLogout() {
      this.logout()
      this.$router.push({
        name: 'login',
      })
    },
    async onLogin() {
      try {
        await this.loginWithGoogle()
        this.$router.push({
          name: 'home',
```

```
        })
      } catch {
        alert('Щось пішло не так')
      }
    },
  },
}
</script>

<style lang="scss">
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  // text-align: center;
  color: #2c3e50;
}

nav {
  padding: 30px;

  a {
    font-weight: bold;
    color: #2c3e50;

    &.router-link-exact-active {
      color: #42b983;
    }
  }
}
</style>
```

Додаємо захист роутів

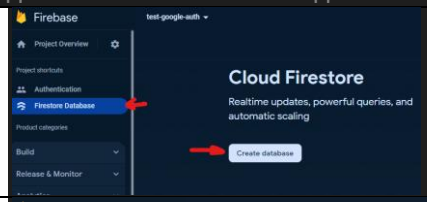


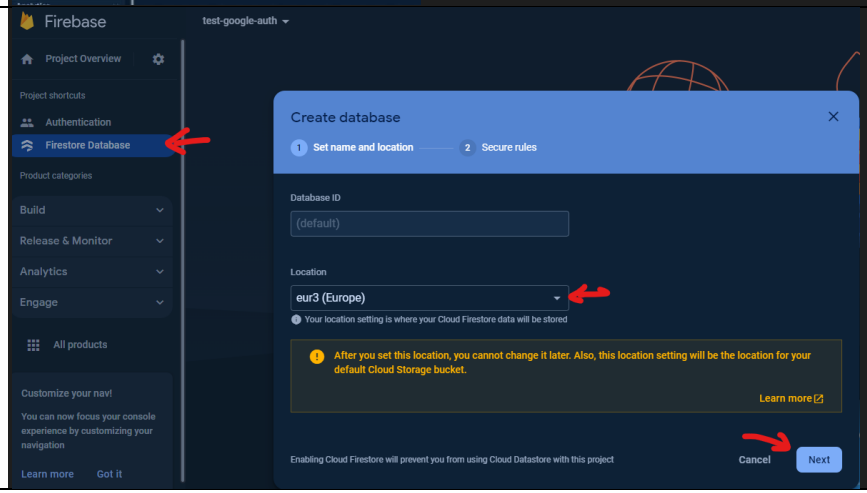

```
router.beforeEach(async (to) => {
 if (to.meta?.requiredAuth) {
   let isAuthenticated = store.state.auth.user

   if (!isAuthenticated) isAuthenticated =
await store.dispatch('auth/loginWithCredential')

   if (!isAuthenticated)
      return {
         name: 'login',
      }
 }
})
```

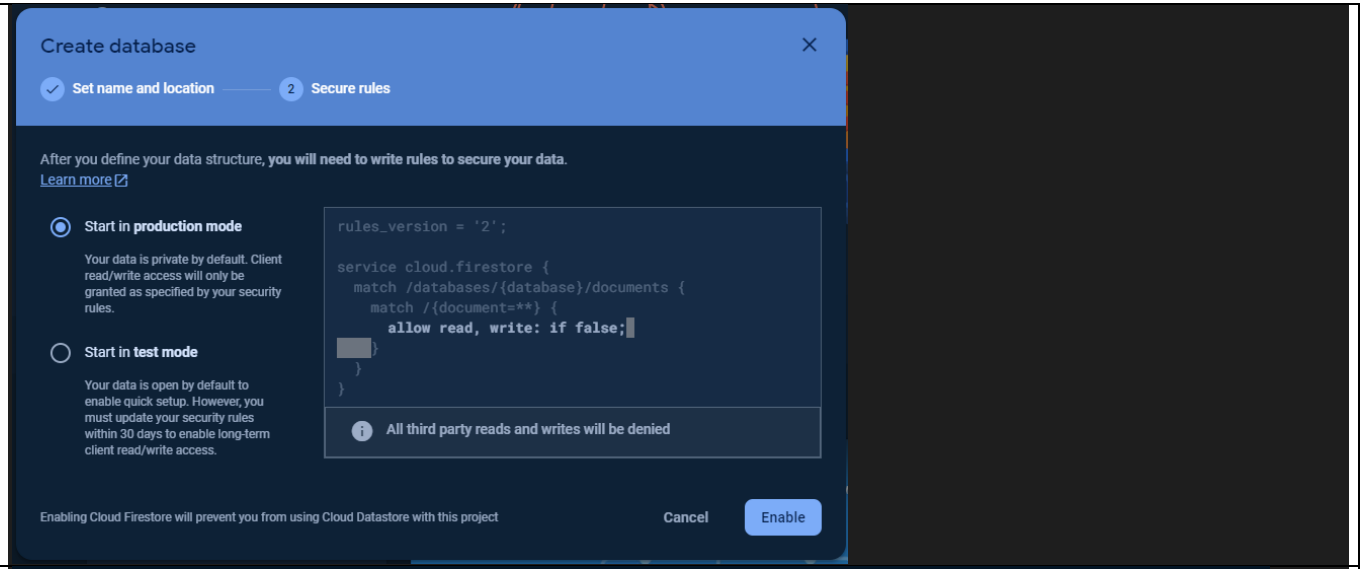
```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import AboutView from '../views/AboutView.vue'
import MyContacts from '../views/MyContacts.vue'
import LoginPage from '../views/LoginPage.vue'
import store from '@/store'
const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },
    {
        path: '/about',
        name: 'about',
        meta: {
            requiredAuth: true,
        },
        component: AboutView,
    },
    {
        path: '/contacts',
        name: 'contacts',
        component: MyContacts,
    },
    {
        path: '/login',
        name: 'login',
        component: LoginPage,
    },
]

const router = createRouter({
    history: createWebHistory(),
    mode: 'history',
    routes,
```

```
})

router.beforeEach(async (to) => {
    if (to.meta?.requiredAuth) {
        let isAuthenticated = store.state.auth.user

        if (!isAuthenticated) isAuthenticated = await
store.dispatch('auth/loginWithCredential')

        if (!isAuthenticated)
            return {
                name: 'login',
            }
    }
})

export default router
```

---------- ОБМЕЖЕННЯ ДОСТУПУ ДЛЯ ДЕЯКИХ КОРИСТУВАЧІВ ДО БАЗИ ДАНИХ ----------

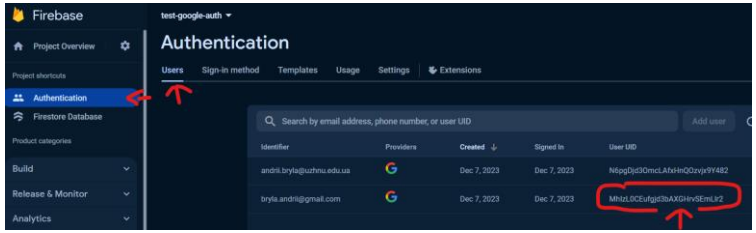




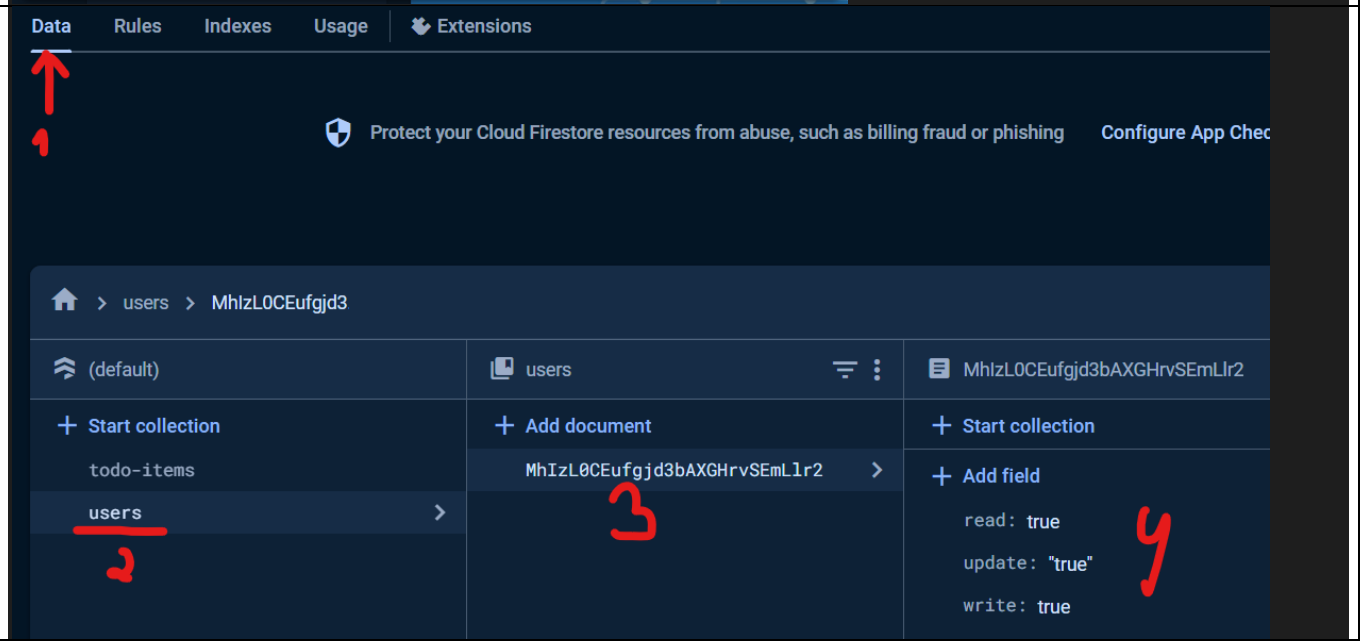


Задаємо колекцію “users”

1)переходимо на вкладку “Data”
2)створюємо колекцію “users”
3)ідентифікатором документа є ідентифікатор користувача

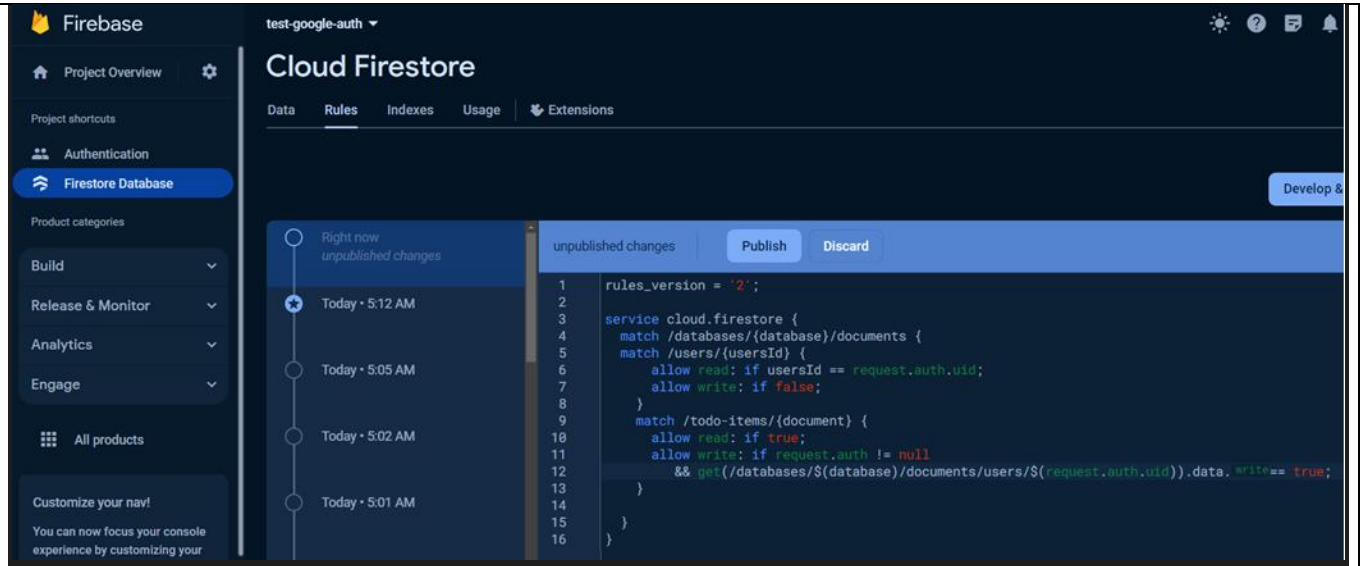


4)дозволи юзера



Задаємо правила (обмеження) використання колекцій

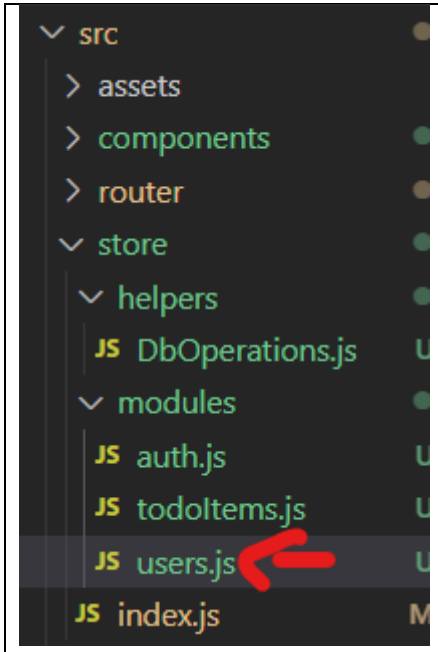
https://firebase.google.com/docs/firestore/security/get-started?hl=ru




```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
  match /users/{usersId} {
      allow read: if usersId == request.auth.uid;
      allow write: if false;
    }
    match /todo-items/{document} {
      allow read: if true;
      allow write: if request.auth != null
         && get(/databases/$(database)/documents/users/$(request.auth.uid)).data.write
== true;
    }

  }
}
```

Створюємо модуль для зчитування даних про дозволи користувачів

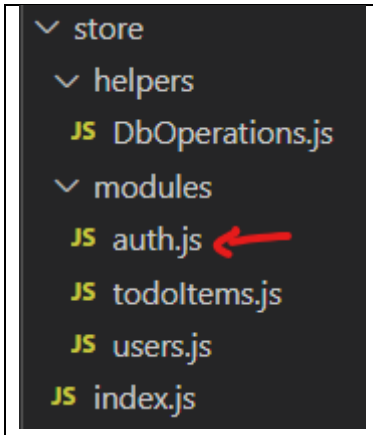
```
import DbOperations from '../helpers/DbOperations'
const collectionDB = new DbOperations('users')
export default {
    namespaced: true,
    state: () => ({
        permissions: {},
```




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

        userPermissions: (state) => state.permissions,
    },
    mutations: {
        setPermissions(state, permissions) {
            state.permissions = permissions
        },

        setLoading(state, value) {
            state.loading = value
        },
        setError(state, error) {
            state.error = error
        },
    },
    actions: {
        loadUserPermissions({ commit, rootGetters }, userId) {
            userId ??= rootGetters['bar/someGetter'].getUser.uid
            collectionDB.getItemById(userId).then((permissions) => {
                commit('setPermissions', permissions)
            })
        },
        clearPermissions({ commit }) {
            commit('setPermissions', {})
        },
    },
}
```

Додаємо у модуль auth завантаження дозволів при авторизації

```
import { auth } from '@/firebase-config.js'
```



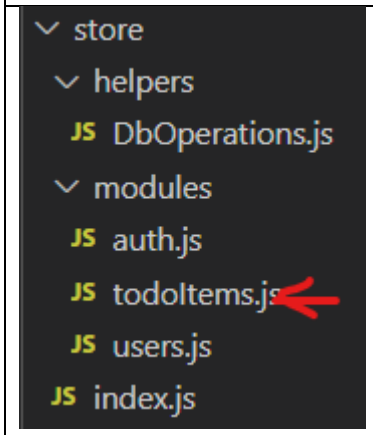

```
import { GoogleAuthProvider, signInWithPopup, signInWithCredential, signOut } from
'firebase/auth'

export default {
    namespaced: true,
    state: {
        user: null,

        loading: false,
        error: null,
    },
    getters: {
        getUser: (state) => state.user,
    },
    mutations: {
        setUser(state, user) {
            state.user = user
        },
        setLoading(state, loading) {
            state.loading = loading
        },
        setError(state, error) {
            state.error = error
        },
    },
    actions: {
        async saveLoginUserData({ commit, dispatch }, loginResult) {
            // async saveLoginUserData({ commit }, loginResult) {
            console.log('loginResult')
            console.log(loginResult)
            //--------- user data -------
            const user = loginResult?.user // об'єкт користувача
            commit('setUser', user)
            //user.displayName - ім'я акаунту
            //user.photoURL - аватар акаунту
            //user.email - електронна адреса акаунту
```

```
dispatch('users/loadUserPermissions', user.uid,
{ root: true })
```

```
            //--------- user access token -------
            // This gives you a Google Access Token. You can use it to access the
Google API.
            let credential = GoogleAuthProvider.credentialFromResult(loginResult)

            localStorage.setItem('authCredential', JSON.stringify(credential))

            dispatch('users/loadUserPermissions', user.uid, { root: true })
        },
        loginWithGoogle({ commit, dispatch }) {
            const provider = new GoogleAuthProvider()
            signInWithPopup(auth, provider)
                .then((loginResult) => {
                    dispatch('saveLoginUserData', loginResult)
                })
                .catch((error) => {
                    commit('setError', error)
                })
        },
        async loginWithCredential({ commit, dispatch }) {
            return new Promise((resolve, reject) => {
                let credential = localStorage.getItem('authCredential')

                if (credential) {
                    credential = JSON.parse(credential)

                    credential = GoogleAuthProvider.credential(credential.idToken)

                    signInWithCredential(auth, credential)
                        .then((loginResult) => {
                            dispatch('saveLoginUserData', loginResult)
                            resolve(loginResult)
                        })
                        .catch((error) => {
                            console.log(error)
                            commit('setError', error)
                            // Handle Errors here.
```

dispatch('users/clearPermissions', null, { root: true })

```
                            //var errorCode = error.code
                            //var errorMessage = error.message
                            // The email of the user's account used.
                            //var email = error.email
                            reject(false)
                        })
                } else resolve(false)
            })
        },
        logout({ commit, dispatch }) {
            signOut(auth)
                .then(() => {
                    localStorage.removeItem('authCredential')
                    commit('setUser', null)
                    dispatch('users/clearPermissions', null, { root: true })
                })
                .catch((error) => {
                    commit('setError', error)
                })
        },
    },
}
```

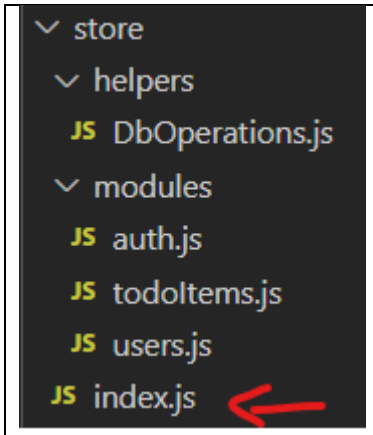



```
import DbOperations from '../helpers/DbOperations'
const collectionDB = new DbOperations('todo-items')
export default {
    namespaced: true,
    state: () => ({
        todoList: [],

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

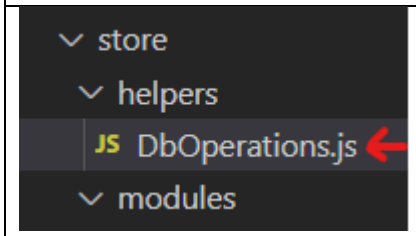
```
        getItemsList: (state) => state.todoList,
        getItemById: (state) => (itemId) => state.todoList.find((item) => item.id ==
itemId),
    },
    mutations: {
        setItemsList(state, itemsList) {
            state.todoList = itemsList
        },
        deleteItem(state, itemId) {
            state.todoList = state.todoList.filter((item) => item.id !== itemId)
        },

        setLoading(state, value) {
            state.loading = value
        },
        setError(state, error) {
            state.error = error
        },
    },
    actions: {
        loadList({ commit }) {
            commit('setError', null)
            commit('setLoading', true)

            collectionDB
                .loadItemsList()
                .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 }, itemId) {
            commit('setError', null)
            commit('setLoading', true)

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




```
import { createStore } from 'vuex'
import todoItems from './modules/todoItems'
import auth from './modules/auth'
import users from './modules/users'

export default createStore({
    namespaced: true,
    state: {},
    getters: {},
    mutations: {},
    actions: {},
    modules: {
        auth,
        users,
        todoItems,
    },
})
```



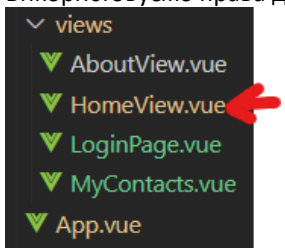

```
import firebaseDB from '@/firebase-config'
import { doc, collection, getDocs, getDoc, addDoc, deleteDoc, updateDoc, query, where
} from 'firebase/firestore/lite'

class DbOperations {
    constructor(collectionTitle) {
        this.dbCollection = collection(firebaseDB, `/${collectionTitle}`)
    }
    getListFromSnapshot(snapshot) {
        const list = []
        snapshot.docs.forEach((doc) => {
            list.push({
                id: doc.id,
                ...doc.data(),
            })
        })
        return list
    }
```

```
loadItemsList() {
    return new Promise((resolve, reject) => {
        getDocs(this.dbCollection)
            .then((querySnapshot) => {
                resolve(this.getListFromSnapshot(querySnapshot))
            })
            .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)
                })
        })
    }
    getItemById(itemId) {
        return new Promise((resolve, reject) => {
            const docRef = doc(this.dbCollection, itemId)
            getDoc(docRef)
                .then((docSnap) => {
                    if (docSnap.exists()) resolve(docSnap.data())
                    else resolve({})
                })
                .catch((error) => {
                    reject(error)
                })
        })
    }
    loadFilteredData(fieldTitle, compareOperator, valueToCompare) {
        const q = query(this.dbCollection, where(fieldTitle, compareOperator,
valueToCompare))
        return new Promise((resolve, reject) => {
            getDocs(q)
                .then((querySnapshot) => {
                    resolve(this.getListFromSnapshot(querySnapshot))
                })
                .catch((error) => {
                    reject(error)
                })
        })
    }
```

```
}

export default DbOperations
```

Використовуємо права доступу




```
<template v-else>
  <div v-for="item in getItemsList"  >
          . . . .
    <button v-if="userPermissions.delete"
      @click="deleteItem(item.id)">
        Delete
    </button>
          . . . .
    <button v-if="userPermissions.update"
      @click="onEdit(item)">
        Edit
     </button>
    </div>
</template>
```

```
<template>
  <div class="home">
    <div v-if="isLoading">Loading ....</div>
    <div v-else-if="hasError">Error</div>
    <template v-else>
      <div v-for="item in getItemsList" :key="item.id">
        {{ item.title }} - {{ item.price }}
        <button v-if="userPermissions.delete" @click="deleteItem(item.id)">
          Delete
        </button>
        <button v-if="userPermissions.update" @click="onEdit(item)">
          Edit
        </button>
      </div>
    </template>
    <hr />
    <div v-if="userPermissions.write">
      <div>
        <label>Title </label>
        <input v-model="task.title" type="text" />
      </div>
      <div>
        <label>Price </label>
        <input v-model="task.price" type="number" />
      </div>
      <button @click="onTaskAction">{{ buttonTitle }}</button>
    </div>
    <hr />
    <h3>Filters</h3>
    <div>
      <label>Price </label>
      <input v-model="searchPrice" type="number" />
```

```
    </div>
    <button
      @click="
        loadFilteredData({
          fieldTitle: 'price',
          compareOperator: '==',
          valueToCompare: searchPrice,
        })
      "
    >
      Search
    </button>
    <button @click="onClearFilter">Clear filter</button>
  </div>
</template>

<script>
import { mapActions, mapGetters } from 'vuex'

export default {
  name: 'HomeView',

  data() {
    return {
      task: {},
      taskCopy: null,
      searchPrice: null,
    }
  },

  computed: {
    ...mapGetters('todoItems', ['getItemsList', 'isLoading', 'hasError']),
    ...mapGetters('users', ['userPermissions']),
    ...mapGetters('auth', ['getUser']),
    isEditing() {
      return this.task.id
    },
```

```
  buttonTitle() {
    return this.isEditing ? 'Save' : 'Add task'
  },
},

created() {
  this.loadList()
},

methods: {
  ...mapActions('todoItems', [
    'loadList',
    'addItem',
    'deleteItem',
    'updateItem',
    'loadFilteredData',
  ]),
  onClearFilter() {
    this.loadList()
    this.searchPrice = null
  },
  onTaskAction() {
    if (this.isEditing) {
      const data = {}

      if (this.task.title !== this.taskCopy.title)
        data.title = this.task.title
      if (this.task.price !== this.taskCopy.price)
        data.price = this.task.price

      this.updateItem({
        itemId: this.task.id,
        data,
      })
    } else this.addItem(this.task)
    this.task = {}
  },
```

	``
	    onEdit(item) {
	      this.task = { ...item }
	      this.taskCopy = item
	    },
	  },
	}
	</script>
	``

https://modularfirebase.web.app/common-use-cases/authentication/

allow write: if request.auth != null && get(/databases/$(database)/documents/users/$(request.auth.uid)).read == true;