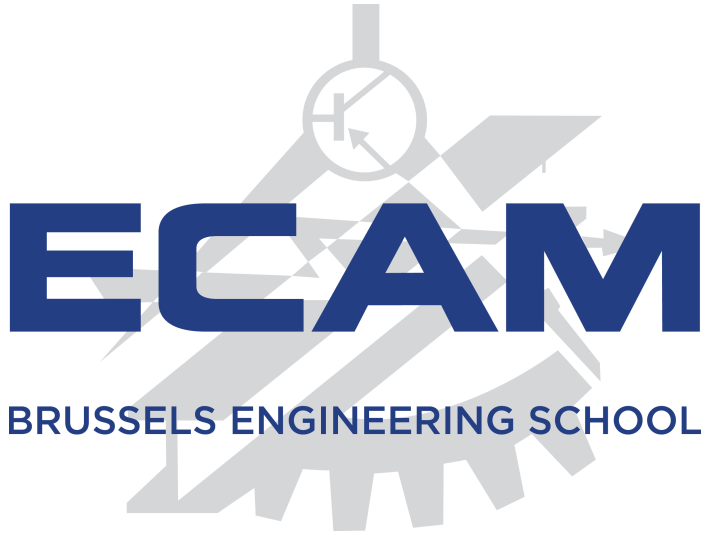

ECAM Brussels Engineering School

2022-2023

NoSQL Databases project:

Ecamlendar with Firebase

Mohamed	**Aousji**	17236
Edouard	**de Schietere de Lophem**	18072
Nicolas	**SAMELSON**	17288
Emene	**Abah**	17282

December 12, 2022

Contents

List of Figures

1 Introduction

In the course of NoSQL Database we were asked to choose a NoSQL Database and to develop a little showcase app showing off the basic features of our chosen database. We also were asked to do some research about it to be able to do a presentation in front of an audience and to explain how to implement it in a project. This report will serve as a base for this presentation.

1.1 What is Firebase?

We chose to develop an app using Firestore. Firestore is a NoSQL cloud-based database service which is developed by Google, and is part of the Firebase solution. Firebase is an app development platform that helps to build and easily store all the data from your apps or games.
This solution is proposed by Google to allow small businesses to develop and make their own apps. As of pricing, two plans are available: Spark and Blaze. The Spark plan is free, and all the features are accessible, but with limits on (e.g. the stored data, transferred data,...), while Blaze is a paid plan. It includes all the features the Spark contains, with additional storage or transferred data.

1.2 Application

The showcase app we decided to develop, is a calendar type app[1] that resembles the iconic calendar web-app known as calendar.ecam.be, but for smartphone. The objective is to make a mobile application that shows a calendar and then gives the user the possibility to display a course-schedule with each individual course displayed as another event. Furthermore, the users would be able to add or delete personal events, meaning that an authentication system will also be added. Flutter was chosen as the programming language for the application as this is also developed by Google and easy to use with the Firebase platform.

[1] Github repository of our showcase application: Github Calendar App

2 How to get started with Firebase?

2.1 Create a Firestore Database

To get started with Firestore, you need to log on to Firebase.com with a Google account. Once logged-in you are presented with the homepage (Figure 1) where you can add a new project. When you create a new project you have to follow some simple steps to set it up. But if you work with other people you can also be invited by someone to join an existing project.

The first step, obviously is to give your project a name, then you're asked if you want to activate Google Analytics for your project or not. Google Analytics is a tool that allows the developer to get free reports of how the users, behave and use the application. As this is only a showcase app, we did not activate Google Analytics. (If you do decide to use Google Analytics, the next step would be to set it up with the correct Geographic zone where you're located in, and then accept the terms of use.) Then, clicking on creating project will set up the environment and directs you to the dashboard of the newly created project.

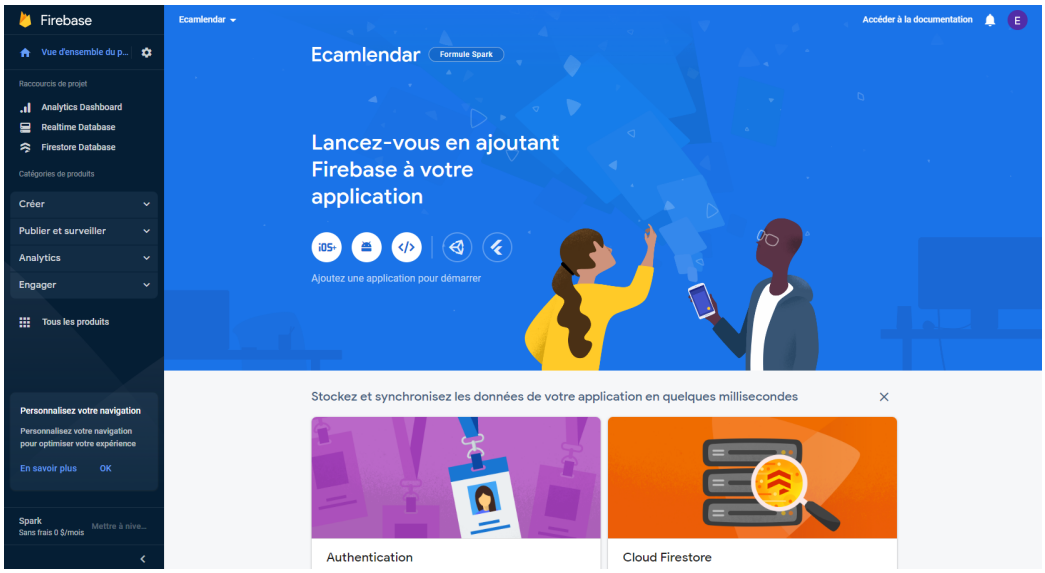

Figure 1: Dashboard of Firebase

As shown on the Figure 1, the dashboard contains all the controls of the project, which are on the left-side of the screen. In the middle, you are presented with quick access to integrate an application as well as recommended services (i.e. Authentication and Cloud Firestore).
To create a new Firestore Database, select "Cloud Firestore" in the "Build" roll-up menu in the left banner (or from the quick access). Then, follow those three simple steps:

1. Start your database in production mode.
2. Choose a geographic zone where the it will be hosted (eur3) for Western-Europe.
3. Once clicked on the Enable Button, your database is being created. And you will be redirected to the Firestore Database window as shown in Figure 2.

2.2 Access Firestore Database

The Firestore Database allows you to access and view the data stored in the chosen NoSQL database. In our calendar application, we can see all the events saved in the database. The data is saved as individual files, which are similar to JSON files but can handle more data types and are limited to a maximum size of 1MB. These files can be considered as lightweight JSON files.

Firestore saves the data entered in the calendar application as individual files (each file corresponds to an event), which are similar to JSON files. These files can be accessed in the "events" collection, as shown in Figure 2. This collection contains information about each event, such as the calendar name, end time, location, and description. This data can then be extracted from the database and displayed on the calendar view in the application.

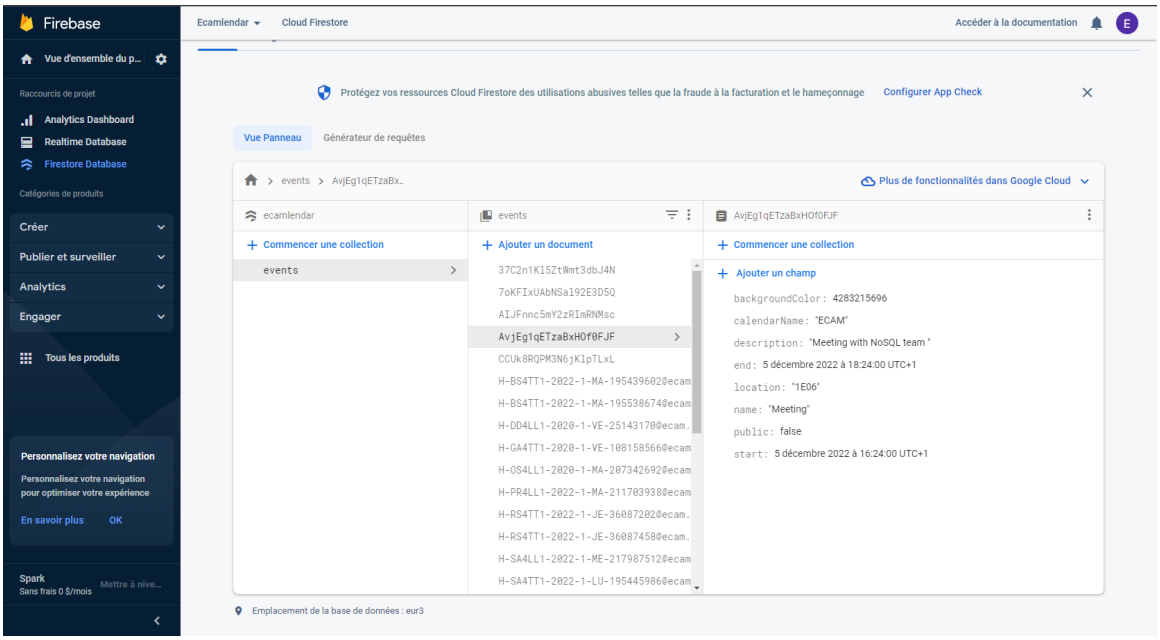

Figure 2: The Firestore Database window

2.3 Connect the App with the Database

To populate the database with Data from our application, we first need to add an application on Firebase. This is done by going to the "Project Overview" window, and then, by clicking on Add App, you will be presented with the steps to follow to connect your app to the database. It mainly consists of selecting the language of the app (e.g. Flutter, JavaScript,...), then downloading the `google-services.json` file and finally implementing it into your project.

2.4 Implementation of Firebase in the code

In the case of a flutter application, the `google-services.json` file needs to be put in the project under the `android/app/` folder. Then, in the `lib/main.dart` file, the following lines are to be added:

```
import 'package:firebase_core/firebase_core.dart';

void main() async {
  // initializer to talk with firebase
  await Firebase.initializeApp();

  runApp(MyApp());
}
```

Finally, the methods used to call the Firestore service can be added in a new dart file under the `lib/` folder:

```
import 'package:cloud_firestore/cloud_firestore.dart';

class DatabaseService {
  // collection reference
    final FirebaseFirestore db = FirebaseFirestore.instance;
    final CollectionReference eventCollection =
        FirebaseFirestore.instance.collection("events");

    // Get a single event searching by id (doc name in Firestore)
    Future<Event> readSingleEvent(String id) async {
        late Event event;
        event = (await eventsRef.doc(id).get()).data()!;
        return event;
    }
}
```

This code snippet can be called in another file of the project and will get the details of the event the user entered (in our case with the id).

3 How does Firestore work?

3.1 Firestore Database vs Realtime Database

Firestore is part of the Firebase cloud solution, which offers two cloud-based, client-accessible database solutions:

1. **RealTime Database**: Firebase's original database.
    - It uses a JSON data model;
    - It allows the user to perform simple, shallow queries at the advantage of being a low-latency solution;
    - It is preferred for the use case of sending a stream of tiny updates;
    - It offers an offline access, allowing the app to continue working even without any connection;
    - It can automatically scale up or down, based on the needs of the application.
2. **Cloud Firestore**: Firebase's database which is build on the Realtime Database with a new and more intuitive data model.
    - It uses a document-oriented data model;
    - It provides more powerful querying capabilities, with complex and deep queries on the data;
    - It does not offer built-in offline support, but data persistence can be used;
    - It can only automatically scale up, by adding more storage and/or servers.

In the case of a small application (like our showcase app), the differences between the two services are not significant. We thus chose Cloud Firestore, as we are more familiar with this database.

3.2 Data Model

First of all, Firestore is a NoSQL document-oriented database. This means that instead of a normal NoSQL database, there are no tables or rows in the data we save. We instead store the data directly in documents which are organized into collections. Each document, then contains a set of key-value pairs (that are called fields). Firestore is made to store a large collection of smaller documents. These documents can contain subcollections and nested objects, both of which can contain primitive fields like strings or lists.

All in all, the data-model is composed of:

- **Collections** in which the documents live. These can be viewed as simple containers for our documents. In our case these are the different events.
- **Documents** corresponding to the individual events and possess each a unique id.
- **Fields** that are key-value pairs. In our project's case, the fields correspond to the name, description, start-time, end-time, etc....

3.3 Data Types

Firestore supports different types of data to be stored in our database. In the table below, you can find all the different forms of data types that you can use to store data on Firestore:

Data type	Sort order	Notes
Array	By element values	An array cannot contain another array value as one of its elements. Within an array, elements maintain the position assigned to them. When sorting two or more arrays, arrays are ordered based on their element values. When comparing two arrays, the first elements of each array are compared. If the first elements are equal, then the second elements are compared and so on until a difference is found. If an array runs out of elements to compare but is equal up to that point, then the shorter array is ordered before the longer array. For example, `[1, 2, 3] < [1, 2, 3, 1] < [2]`. The array `[2]` has the greatest first element value. The array `[1, 2, 3]` has elements equal to the first three elements of `[1, 2, 3, 1]` but is shorter in length.
Boolean	`false < true`	—
Bytes	Byte order	Up to 1,048,487 bytes (1 MiB - 89 bytes). Only the first 1,500 bytes are considered by queries.
Date and time	Chronological	When stored in Cloud Firestore, precise only to microseconds; any additional precision is rounded down.
Floating-point number	Numeric	64-bit double precision, IEEE 754.
Geographical point	By latitude, then longitude	At this time we do not recommend using this data type due to querying limitations. It is generally better to store latitude and longitude as separate numeric fields. If your app needs simple distance-based geoqueries, see Geo queries
Integer	Numeric	64-bit, signed

Map	By keys, then by value	Represents an object embedded within a document. When indexed, you can query on subfields. If you exclude this value from indexing, then all subfields are also excluded from indexing.
		
		Key ordering is always sorted. For example, if you write `{c: "foo", a: "bar", b: "qux"}` the map is sorted by key and saved as `{a: "bar", b: "qux", c: "foo"}`.
		
		Map fields are sorted by key and compared by key-value pairs, first comparing the keys and then the values. If the first key-value pairs are equal, the next key-value pairs are compared, and so on. If two maps start with the same key-value pairs, then map length is considered. For example, the following maps are in ascending order:
		
		`{a: "aaa", b: "baz"}`
		`{a: "foo", b: "bar"}`
		`{a: "foo", b: "bar", c: "qux"}`
		`{a: "foo", b: "baz"}`
		`{b: "aaa", c: "baz"}`
		`{c: "aaa"}`

Null	None	—
Reference	By path elements (collection, document ID, collection, document ID...)	For example, `projects/[PROJECT_ID]/databases/[DATABASE_ID]/documents/[DOCUMENT_PATH]`.
Text string	UTF-8 encoded byte order	Up to 1,048,487 bytes (1 MiB - 89 bytes). Only the first 1,500 bytes of the UTF-8 representation are considered by queries.

Figure 3: Supported Data-Types

3.4 Firestore Cloud locations

When using Firestore, you will need to choose a location for your database. As said before, when starting up a project, you have to choose a region of your choice. This is an important step to avoid latency and increase the availability of the data you want to store, as it needs to be as close as possible to the users and chosen services. As our app is a small school project, we chose the "eur3" region name as this is the special code for Western-Europe (where we are located).

4 Features of Firestore

4.1 Storage Feature

One great feature of Firestore is the storage option. In the "create" tab, you can find this feature. This option allows you to store, for example, an image in the database and then use it anywhere you want. This is great because from past experience with SQL databases, we all know that storing and then using images can be difficult.

4.2 Authentication

The main feature of Firestore, which is game-changing, is it's Authentication feature. This allows us to easily implement user authentication to our application. This is useful for any kind of app. In our case, this means that we can give each user its own calendar. Our users can login with their Google Account or any email-password combination and it will be stored in the Firestore database. Another feature in Authentication, is the ability to create customized sign-in confirmation emails. Those features are easily implemented, and above all, Firebase makes sure to encrypt the passwords and any sensible information. This means that the developers (us) will not have any access to the user's passwords.

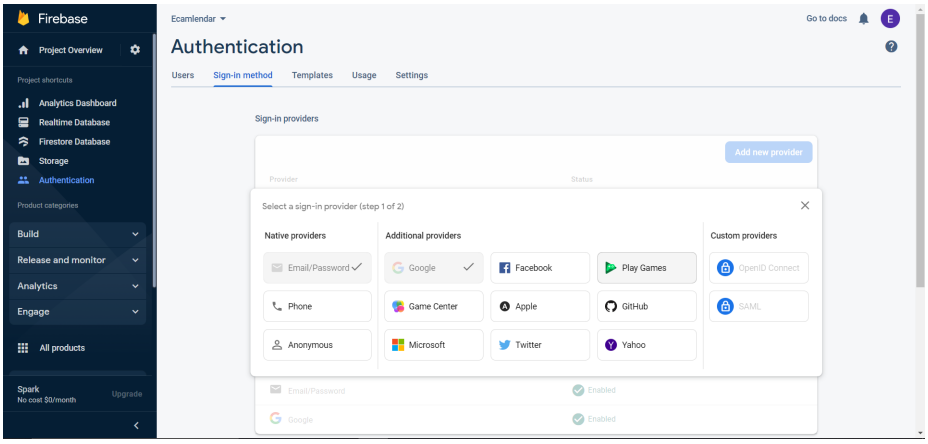

Figure 4: Possible Log-In methods

4.3 Extensions

Firestore also allows us to add Extensions for the project, this can be useful to deploy for example payment solutions or text search of your database. There is a big marketplace where you can find a multitude of extensions that can suit the needs of your application.

4.4 Release and Monitor

This tab allows us to monitor performance and stability of our app. This can help to improve our application by detecting any issues through tests.

4.5 Analytics

Firebase includes the Google Analytics tool, which is a free and unlimited way to track up to 500 distinct events within your app. This tool provides detailed reporting and insights into how your app is being used. The Analytics data can help the developers understand the behavior of your users and make informed decisions about how to improve the application.

4.6 Consistency Model

A consistency model refers to the rules and requirements a database transaction must follow to be accurate and reliable.
In the case of Firestore database, the service implements a strong consistency model. In other words, any transaction to change the data will be predictable and of an expected manner. If multiple users are accessing and updating the same document, they should all see the same updates and changes in real time which is the case for Firestore.

4.7 Language Support and how to implement?

Firestore is supported by multiple different languages. Here are a few of them:

1. JavaScript
2. Flutter
3. C++
4. Python
5. ...

To integrate the Firebase solution into a new application, we need to go to the general settings of the Firebase console (as explained before) and there we can find an "Add APP" button with all the languages supported by Firestore.

4.8 Access Documentation

Even if Firebase is easy to use, there is still plenty of documentation and online support. It has its own documentation website[2] and contains tutorials on how to get started, as well as tutorials for each of their service and each supported language.

[2]Firebase Website with all the needed documentation: firebase.google.docs

5 Presentation of the app

5.1 Use cases of the application

The application developed as a showcase is a calendar app for students. It was developed in Flutter, which is a mobile application development framework created by Google and has the advantage to be easily ported to different devices (iOS, Android) and desktop platforms (Windows, MacOs, Linux), as well as providing support for web applications. This is a great advantage in this specific use case, as students would be able to access anywhere, anytime and on whichever device they choose to use.

The user stories for this project are the following:

1. As a user, I want to create an account;
2. As a user, I want to log in with my account in order to access to my calendar;
3. As a user, I want to create a private event and customize it;
4. As a user, I want to see the public events;
5. As a user, I want to see the private events I created and see the the details;
6. As a user, I want to be able to modify the details of my private events.

Now, let's present the application:

When the application is launched, it opens up on the Log-In/Sign-up screen where users can log-in to their calendar App with their account, or they can create a new account if they do not have one. This is put in place with the Authentication feature of Firestore we talked before. We could even have put in place an automatic emailing system to reinitialize the password if the user forgot its password.

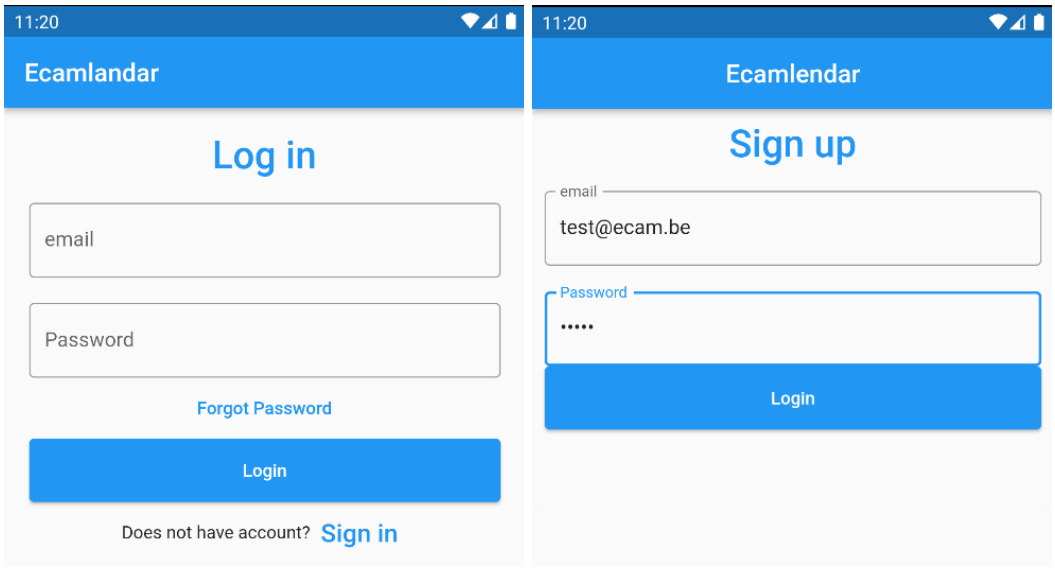

Figure 5: Login & Signup Page

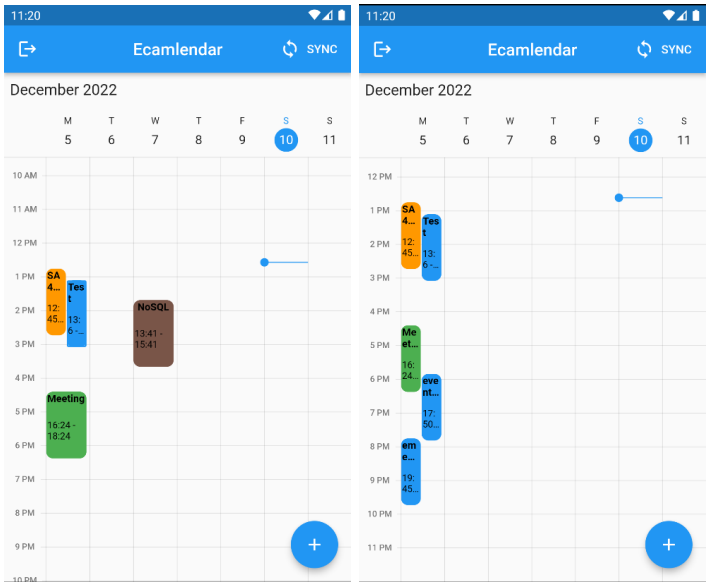

Figure 6: Main Calendar

The calendar views shown here are from two different persons, you can see the "course" type of events which are common for both calendars from persons in the same year. And "personal" type of events like meetings that are specific to each person.

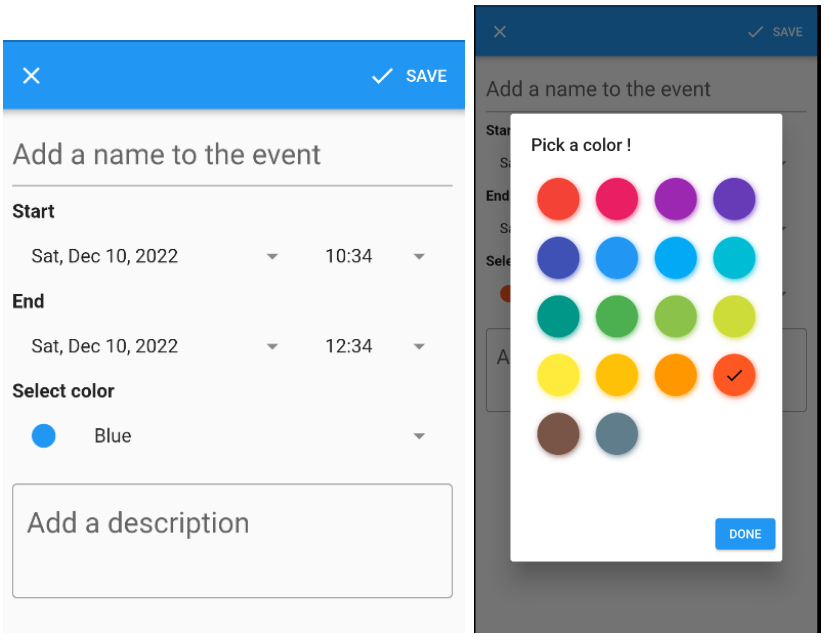

Figure 7: Adding an event and changing it's color.

The Figure 7 shows the page to create or modify an event. We can give the new event a name, a starting time and an ending time, as well as giving it a special color that will show in our calendar view, we can also give it a more detailed description if needed.

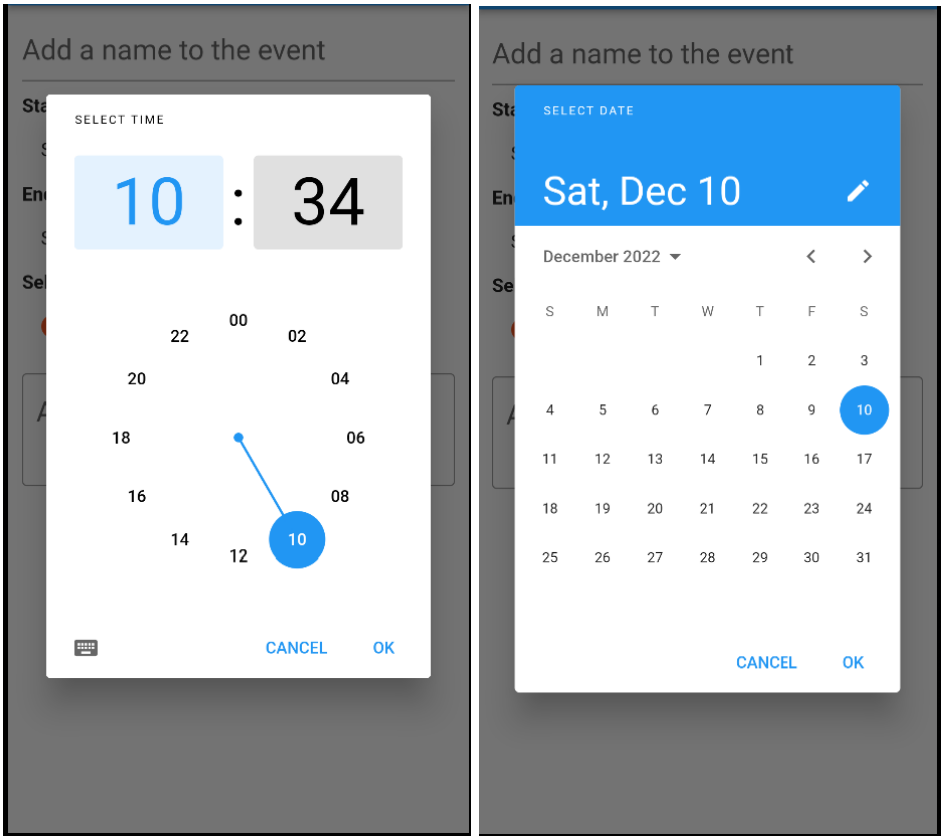

Figure 8: Changing the time and Date

Here you can see the change time and change date options when you're creating an event (these windows are the same if you're editing the time or date of an event).

6 Conclusion and what we learned

The project of a calendar in Flutter using Firestore database has been successful in its implementation. The real-time synchronization of events in the database makes it a useful tool for managing schedules, and the use of flutter allows for a visually appealing and user-friendly interface. The project showcases the potential of using these technologies for creating engaging and useful applications.
Firestore and other Firebase services are easy to implement in any project. This solution also benefits from a lot of documentation and information about it (it has its own "help desk") and the fact that it is used by a lot of people, there are a lot of tutorials on the internet on how to use it. Some of its features are also very interesting and we strongly recommend the Authentication feature for anyone that needs a secure Log-In system.

References

[1] Cloud Firestore Data model. Firebase. URL: https://firebase.google.com/docs/firestore/data-model (visited on 12/11/2022).

[2] Firebase Authentication. Firebase. URL: https://firebase.google.com/docs/auth (visited on 12/11/2022).

[3] Firestore. Firebase. URL: https://firebase.google.com/docs/firestore (visited on 12/11/2022).

[4] flutter_colorpicker — Flutter Package. Dart packages. URL: https://pub.dev/packages/flutter_colorpicker (visited on 11/21/2022).

[5] Github. URL: https://github.com/AmineAousji/Calendar_App (visited on 12/11/2022).

[6] HSL course material. URL: https://kenn7.github.io/nosql/project/ (visited on 12/11/2022).

[7] Install. URL: https://docs.flutter.dev/get-started/install (visited on 11/10/2022).

[8] Supported data types — Firestore. Firebase. URL: https://firebase.google.com/docs/firestore/manage-data/data-types (visited on 12/11/2022).

[9] syncfusion_flutter_calendar — Flutter Package. URL: https://pub.dev/packages/syncfusion_flutter_calendar (visited on 11/09/2022).