Eventually (MVP)

An event planner and guest list manager

Jorge Otero
SWDV-691: Capstone
Maryville University
December, 2020

1 Overview

Eventually is a simple way of planning events and managing guest lists. Eventually can be used to plan live, virtual or hybrid events, and coordinate invitations, RSVPs and seating arrangements. All of this in one intuitive and simple interface.

1.1 Minimum Viable Product (MVP)

The application's MVP is meant to deliver the following features:

- User authentication workflow; which consists of:
  - Registering an account
  - Logging in
  - Signing out
  - Resetting a user password
- Persistent data storage for:
  - Users
  - Events
  - Guests
- The ability to create, retrieve, update and delete:
  - Events
  - Guests

These features are meant to give the user the ability to manually keep track on events and their associated guest lists. This constitutes an initial capability from which the application grow and evolve its features based on user interaction and feedback.

2 Front-end: Ionic (Angular) Framework Layout

The application was built using the Ionic framework (6.10.1) and Angular (10.0.7). Ionic is a hybrid application development framework and software development kit (SDK). Angular is a TypeScript-based web application framework meant to facilitate front-end development with built-in design patterns and utilities that compile to valid JavaScript.

The application's layout consists of the following core levels (among others):

- App name: top directory with the application's entire development environment.
  - Src: main directory for the application's source code and dependencies.
    - App: the directory containing the application's source code.

2.1 Pages

A "page" is a type of module (a cohesive block of code with a closely related set of capabilities) meant to render as a complete screen in the user's display. A page's core elements are an HTML template (structure of the page), a Sass file for style (look and feel of the page) and a TypeScript file (the behavior of the page).

This application's pages are organized as follows:

2.1.1 Home

The application's landing page after a successful login. This page contains a set of cards representing event and display their respective event's information.

Route: 'home/:uid'

2.1.2 Login

The default route for the application. This page displays a form for the user to authenticate with and buttons to link to other parts of the authentication workflow.

Route: ‘/’ or ‘login’

2.1.3 Register

This page contains a form use to create a new user. Once a user is created the application routes them to their home page.

Route: ‘register’

2.1.4 Reset

This page allows the user to trigger a password reset workflow via email verification. The user will receive a reset link to the provided email, which will take them to a password reset form. After submitting the form, the user will be routed the login page to authenticate themselves.

Route: ‘reset’

2.1.5 Event Details

This page displays the details of an event when an event card is clicked (touched) in the home page. The page retrieves the details from the database and autocompletes an input form with them. This allows the users to update the details if desired from the same page.

Route: 'home/:uid/event-details/:eventid'

2.1.6 Add Event

This page is presented after the user clicks the floating action button (FAB) on the bottom right of the home page. It displays a form for the user to enter an event’s title (required), description (optional), date (required), time (required), and venue (optional). Once submitted, an event object is created in the database with the aforementioned information and the user is routed to the home page.

Route: 'home/:uid/add-event'

Guest List

This page displays the list of guests associated with an event. It is displayed when the user clicks the link in the event details page. When not empty, the guest list consists of sliding items displaying the guest’s name, RSVP response and an avatar. Sliding the item left reveals a button to edit that guest’s information, and sliding the item right reveals a button to remove that guest from the list.

Route: 'home/:uid/event-details/:eventid/guest-list'

2.1.7 Add Guest

This page displays a form after the user clicks the FAB on the bottom right of the guest list page. The form allows the user to enter guest information and once submitted, a new guest object is created in the data base. The user is then routed to the guest list page.

Route: 'home/:uid/event-details/:eventid/guest-list/add-guest'

2.1.8 Guest Details

This page displays the selected guest’s information. Each field is autocompleted from data retrieved from the database which allows the user to update it on the same page.

Route: 'home/:uid/event-details/:eventid/guest-list/:guestid/guest-details'

2.2 SERVICES

An Angular service is a class that can be injected as a dependency to another module. The properties and methods in this class are a form of reusable code available to all other modules.

2.2.1 CanLoad

The CanLoad interface provides the class that implements it, the ability to restrict a module from being loaded if a test is not passed.

2.2.2 AuthGuard Service

The AuthGuard service was created to prevent unauthorized access to routes that require user authentication to access.

The service checks with the backend service has the user authenticated, if so, it allows the requested route to be loaded and displayed. Otherwise, it redirects the user to the login page.

2.3 PROGRESSIVE WEB APP

The Ionic Angular SDK facilitates the addition of progressive web app boilerplate code. This code was included and customized to allow the user to add the application to their desktop or mobile device home screen without the need to install it or deploy it via an app store.

2.4 AUTOMATED DEPLOYMENT

The application is automatically deployed using GitHub Actions. Using a YAML file to define the deployment process the Action triggers on pull requests to the main (default) branch.

The workflow file defines the trigger. Then creates an Ubuntu container and copies the repository files onto it. The dependencies are installed first, then the ionic project is built. Finally, using a secret Firebase token, the workflow deploys to the appropriate Firebase hosting account.

3 BACK-END: FIREBASE

Eventually’s backend was build on Google’s Firebase. Firebase is a managed backend solution with an intuitive API for quick application deployment. The following sections describe the Firebase functionality implemented in this application. All backend API code was implemented using the @angular/fire library and the firebase-tools CLI.

3.1 AUTHENTICATION

The AngularFireAuth module was used to facilitate the authentication workflow. Firebase’ signInWithEmailAndPassword() method takes a user’s email and password strings respectively and returns a promise that, when resolved, contains metadata about the user; which includes a user ID – a hexadecimal string that uniquely identifies a registered user.

The createUserWithEmailAndPassword() method takes a similar input as above and it is used to create a user in the firebase authentication service. Likewise, it returns a promise the provides user metadata when resolved.

The sendPasswordResetEmail() method a string with a user email and an object with a ‘url’ property. An email, which may be customized from the Firebase console, will be sent with a link to reset the current password. The object take as an argument is passed to Firebase to redirect the user to the desired URL when the password has been reset.

3.2 DATABASE

Firebase provides a persistent data storage service called Firestore. This is a noSQL database structure as collections which hold documents. A document is a set of key-value pairs which do not follow any particular schema.

The AngularFirestore module was used to facilitate interaction with the database.

The collection() method returns a reference to the Firestore collection name passed as a string argument. This reference can be used to query the collection either by getting the current state of the database via the valueChanges() method or subscribing to a stream of the state of the database with snapshotChanges().

When retrieving the documents in the collection, the returned data needs to be unpacked. The events in the home page, for example, are mapped into an array of objects with 'eventid' and 'eventdata' properties. These properties are used to identify the document containing the event information and the details of the event.

The Firestore doc() method gets a string argument in the form "collection/document ID" and returns a reference to a specific document. This reference can be used to update the entire document with the update() method or remove it from the data base with the delete() method.

3.3 Hosting

The Firebase hosting service was used to deploy Eventually. To deploy, the application was compiled with the following ionic CLI command:

```
ionic build --engine=browser --prod
```

The build command is the equivalent of running the 'ng build' npm command. The --engine=browser option tells the compiler to target a web browser rather than a mobile application. Finally, the --prod option tells the compiler to use the production environment variables. The output of the build process is saved in the 'www' directory.

Using the firebase-tools CLI the output of the build process was launched on Firebase with the command:

```
firebase deploy --only hosting
```

On the initialized project folder, this command deploys the contents of the 'www' directory onto firebase.

In addition, firebase provides preview channels. These are temporary deployments that can be used for testing. These were used throughout the development to test the application and to provide temporary links for peer reviews.

4 Stretch Goals

Beyond the MVP deployment, the application will evolve with the following features:

- Invitations:
  - Eventually will give the user the ability to send invitation and automatically keep track of responses.
- Social media 3rd party authentication:
  - Eventually will allow users to log in user social media account credentials.
- Social media integration:
  - Eventually will allow users the ability to post events on their social media platforms.
- Messaging:
  - Eventually will allow users and guests to communicate via instant messaging.
- Collaboration:
  - Eventually will allow multiple users to collaborate in the planning of an event.
- Ticketing:
  - Eventually will allow users to sell and purchase tickets to events.