Angular: Platform and Framework for Web Application Development

Jesús Moreno Durán
Fulda University of Applied Science
jesus.moreno-duran@informatik.hs-fulda.de

Contents

ABSTRACT

In this write-up, we will be discussing Angular, a framework and platform that enables the development of client-side single-page applications (SPA) through the use of TypeScript and HTML. Additionally, we will provide a hands-on demonstration of how to create a web application using Angular, specifically one that allows for car sharing and ride-sharing. The platform enables users to publish travel routes and fares, while others can join in. However, it's important to mention that the application's implementation is based on dummy data and can only be accessed through the frontend.

1. INTRODUCTION

Sharing cars has gained immense popularity in recent times due to its cost-effective and travel-efficient benefits. However, managing coordination between users can often be challenging. To tackle this issue, an Angular-based web application has been created that enables users to share rides and travel together seamlessly, thereby reducing traffic congestion and also promoting environmental sustainability by reducing the number of cars on the road.

1.1 What is Angular?

Angular is a powerful platform and open-source framework used for developing single-page applications (SPA) [1, 2]. It was created and is maintained by Google, and provides a complete solution for building dynamic web applications using HTML, CSS, and TypeScript.

The framework is designed to simplify the development process by providing a structured approach for building complex client-side applications. Angular utilizes declarative templates and dependency injection to separate application logic from presentation, allowing developers to focus on the core business logic of their applications.

One of the key features of Angular is its ability to handle complex data models and provide a scalable architecture for building large applications. It also includes a robust set of built-in tools and libraries that simplify common tasks such as routing, forms handling, and testing.

Angular's popularity is due in part to its flexibility and ability to integrate with other tools and technologies, making it an ideal choice for a wide range of web development projects.

Angular was initially developed as a private project by Google, with the first version being released internally in 2009. However, it wasn't until September 2016 that the framework was officially made available to the public, with the release of Angular 2. This move to open-source made Angular accessible to a wider community of developers, leading to its rapid growth and adoption as a leading framework for building single-page applications. Despite its transition to an open-source project, Angular continues to be supported and maintained by Google, ensuring that it remains a cutting-edge technology for years to come.

1.2 Features of Angular

- TypeScript: Angular is built using TypeScript, a statically typed superset of JavaScript that provides improved error checking and code maintainability.
- Components: Angular provides a powerful component based architecture that allows develop complex applications by composing simple components. This makes it easy to reuse and share code, and simplifies the development of large-scale applications.
- Dependency Injection: Angular uses dependency injection to manage object creation and simplify the development of complex applications.
- Declarative Templates: Angular utilizes declarative templates to separate application logic from presen-

tation, making it easier to maintain and scale applications.

- Routing: Angular includes a powerful routing system that enables developers to build applications with multiple views and navigate between them.
- Progressive Web Apps (PWA): Angular provides support for building Progressive Web Apps, which can be installed on users' devices and function like native apps. PWAs built with Angular are fast, responsive, and can work offline, making them a powerful tool for engaging users and improving user experience.

1.3 How to create a project

In order to create a project in Angular [1, 3, 4] you need to follow these steps:

1. Install a code editor. To edit the project, you need to have a code editor installed on your system. Visual Studio Code is a good option as it has extensions to facilitate development in Angular.
2. Install Node.js and npm. Angular uses Node.js and npm as package managers. Hence, you need to have them installed on your system.
3. Install the Angular client. To install the Angular client, use the command line tool and run this command: `npm install -g @angular/cli`. It will install the Angular client globally on your system.
4. Create a new project. To create a new project, execute: `npm new [project-name]`. This command creates a new folder with the name of your project and generates the basic structure of the project in it.
5. Run the project. Navigate to the project folder using the command `cd [project-name]` and then execute the following command: `ng serve -o`. This command compiles the project and starts it in your default browser, allowing you to see the changes in real-time as you make them.

2. STATE OF THE ART

2.1 Angular vs React

Angular and React are both popular JavaScript frameworks used for building web applications. Differences [5]:

- Flexibility: React is known for its flexibility, giving developers more freedom to structure their projects and choose their preferred syntax. It also allows the use of different complementary technologies. Angular, on the other hand, offers a stricter project structure and syntax that can be beneficial for novice developers or large teams needing to follow a defined standard.
- Origin: Google develops and maintains Angular, while Facebook is responsible for React.
- Typing: Angular uses Typescript, a typed programming language that provides greater security and ease of maintenance for large projects. React, on the other hand, only requires JavaScript and JSX for component creation.
- Templates: Angular uses HTML templates for component creation, which can be beneficial for web designers and make it easier to create reusable components. In contrast, React uses JSX, a JavaScript extension that allows for component creation with a syntax similar to HTML.
- Data binding: Angular allows for bidirectional data binding, where changes to the data model are automatically reflected in the view. React, however, uses unidirectional data binding, requiring changes in the data model to be explicitly passed to the view using props.

2.2 Angular information sources

Learn Angular is not difficult at all. There are several sources where you can learn.

YouTube is a great resource for finding channels that offer tutorials for beginners. For example, 'Amigos Code' [6] provides straightforward tutorials that are easy to follow. 'FreeCodeCamp.org' [7] and 'Traversy Media' [8] are also excellent channels to check out. Additionally, using the search bar can help you find exactly what you need to learn.

Always is a good option take courses in platforms like Udemy, Coursera or others similar.

There are several resources available to developers looking for help with Angular. The official Angular documentation is a recommended source, as it provides comprehensive information about the framework. In addition, there are several forums such as 'Stack Overflow', 'Reddit' and 'Angular Discussions', where developers can ask specific questions and get answers from the community. These resources can be useful for troubleshooting, gathering information and learning best practices related to Angular development.

3. METHODOLOGY

An example of how Angular is applied in web application development is demonstrated through a carpooling application [9]. This practical application enables users to share rides and travel together, offering a platform for posting routes and prices for others to join. The application is based on dummy data and is only implemented on the frontend.

In the subsequent paragraphs, we will explore various sections of the Angular code written for the development of this carpooling application.

```
const appRoutes: Routes=[
  {path:'', component: IndexPageComponent},
  {path:'login', component: LoginPageComponent},
  {path:'register', component: RegisterPageComponent},
  {path:'newtrip', component: NewTripPageComponent},
  {path:'trips', component: TripsPageComponent},
  {path:'mytrips', component: MyTripsPageComponent}
]

@NgModule({
  declarations: [
    AppComponent,
    IndexPageComponent,
    NavarComponent,
    FooterComponent,
    LoginPageComponent,
    RegisterPageComponent,
    NewTripPageComponent,
    TripsPageComponent,
    TripCardComponent,
    MyTripsPageComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    RouterModule.forRoot(appRoutes),
    HttpClientModule,
    GoogleMapsModule
  ],
  providers: [UsersServiceService,
  TripServiceService],
  bootstrap: [AppComponent]
})
```

Figure 1. Declaration of appRoutes and NgModule with components and their corresponding routes.

app.module.ts

app.module.ts [9] is used to define and configure the application's components, services, and dependencies. It is the root module of an Angular application, and it sets up the dependency injection system, which is used to provide instances of services and other dependencies to components and other parts of the application.

In the Figure 1, the appRoutes array defines the routes for the application and maps each path to a corresponding component. When a user navigates to a path in the application, Angular uses the defined routes to determine which component to render. For example, the IndexPageComponent is associated with the root path (''), while the LoginPageComponent is associated with the 'login' path.

The @NgModule decorator is used to define the metadata for the root module of the application. The declarations property is an array of components that belong to the module. The imports property is an array of other modules used in the application. The providers property is an array of services that will be at the application.

App Component

In the Figure 2 we see template of the root component [9]. It has a navbar and a footer and between that two it exist a component called 'router-outlet', it shows the specific component depending on the route defined on app.module.ts in the Figure 1

```
<app-navar></app-navar>
<br>
<router-outlet></router-outlet>
<br>
<app-footer></app-footer>
```

Figure 2. app.component.html file of Carida project.

Models

```
import { Trip } from "./trip.module";

export class User{

  name?:string;
  email?:string;
  password?:string;
  trips?:Trip[];

  constructor(email?:string, password?:string,
    name?:string, trips?:Trip[]){
    this.email = email;
    this.password = password;
    this.name = name;
    this.trips = trips;
  }
}
```

Figure 3. User class created in Angular.

In Angular we use models to define objects and we use this models in the aplication [9]. In the Figure 3 we can see how we define a user in Carida. The class has a constructor method tho initialize the properties of the objects.

In Carida we also define the model Trip which has the properties: id, start, finish, price, date, driver, freeSeats, occupiedSeats, startLatitude, startLongitude, finishLatitude, and finishLongitude. The definition of this model is pretty similar to the model user.

Services

In the Figure 4 is shown the service that is used to manage the information related to the created trips [9].

The annotation '@Injectable(providedIn: 'root')' indicates that only one instance of this service will be created and shared by the whole application.

The dummy data we have in the application is created and stored in the trips list. As there are 4 trips created at the beginning, the ID control starts at 5.

In the service we have also defined 3 methods that have been necessary to manage the information:

- 'addTrip(trip:Trip)': this method creates a trip by assigning it an id and adding it to the list. In a real situation this method would be in charge of sending the Trip to the backend and nothing else.
- 'findTrip(trip:Trip): Trip — undefined': this method searches for a specific trip based on the departure

```
@Injectable({
  providedIn: 'root'
})
export class TripServiceService {

  //Dummy data trips
  trips: Trip[] = [...];
  //Id of the data (dummy)
  singleID: number = 5;

constructor() { }

addTrip(trip:Trip){
  trip.id = this.singleID;
  this.singleID++;
  this.trips.push(trip);
}

findTrip(trip:Trip): Trip | undefined{
  return this.trips.find(TripToFind => TripToFind.start == trip.start
    && TripToFind.finish == trip.finish);
}

getTrips(): Trip[]{
  return this.trips;
}

}
```

Figure 4. Trip service with the methods that are implemented.

and arrival points and returns the object or undefined if it does not exist.

- 'getTrips(): Trip[]': all trips are returned.

In the project we use two more services, one dedicated to manage the users and the other to manage the map shown in the application.

Components

```
<div class="container bg-palid-green rounded shadow">
  <p class="text-center fs-1 p-4 fw-bold">Trips</p>
</div>

<div class="container" id="main-container">

  <ng-container *ngFor="let trip of trips">
    <app-tripCard [trip]="trip" [booked]="false"></app-tripCard>
  </ng-container>

</div>
```

Figure 5. HTML code of the trips page component.

In the Figure 5 we see the code of the page where we can see all the trips avaliable to book [9], the page resulting from this code can be seen in the Figure 7.

The first section of the code corresponds to the title of the page. Further down, we utilize the Angular syntax to implement a foreach loop that iterates over each trip in the trips list.

The trips list, which contains all the available trips for booking, is created in the tripsPage.component.ts file and is populated by the 'getTrips()' method of the service, as seen in Figure 4.

The foreach loop generates cards for each trip, as shown in Figure 7. We pass the trip data to the 'tripCard' component and set the 'booked' variable to false. If the variable were set to true, the button 'Book a seat' would appear in red with the text 'Cancel Reservation'.

4. RESULTS

Although Carida is not a functional application at the moment, the result of this project is a simple and intuitive website that makes car sharing easy for anyone.

Every user, trip or booking created during a sesion is deleted once the page is reloaded.

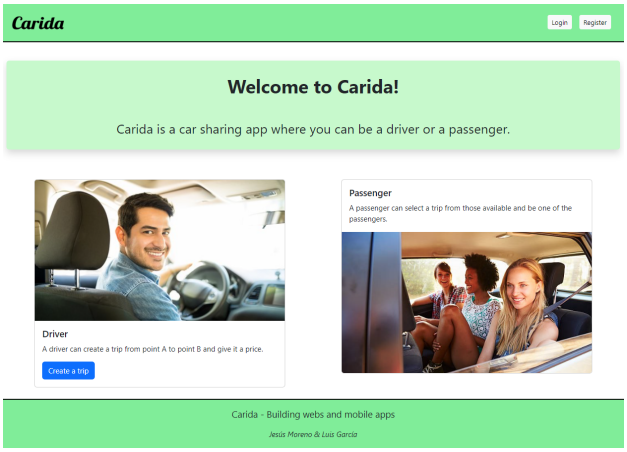

Figure 6. Carida's main page.

Figure 6 displays the main page, which features a navbar that offers the options to Login or Register. A brief description of the page is also provided through a short text, along with two cards that showcase the various roles available to users.

The login and register sections have no special features.

Once a user is logged the buttons in the navbar change and allows you to go to avaliable trips sections, to create a trip and let you see your trips.

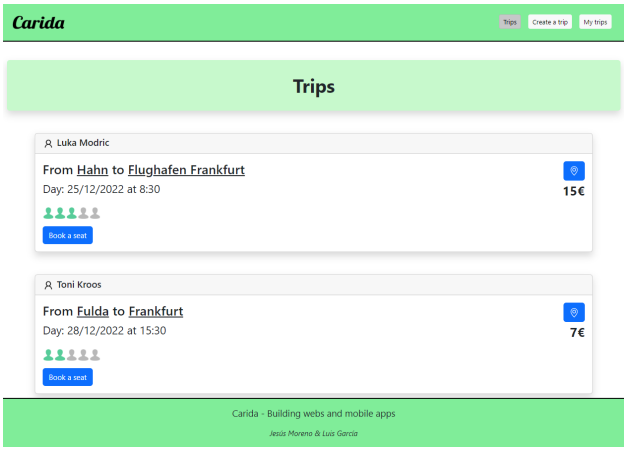

Figure 7. Page of Carida where you can see all the trips avaliable to book.

Figure 7 displays all available trips that can be booked by the user. These trips are the ones that have not been

fully booked or booked by the user already. The available trip card provides information about the route, date, price, time, and available and booked seats. Furthermore, if the user clicks on the location button, a map will open as hte one on the Figure 8, allowing them to view the exact point of the route.

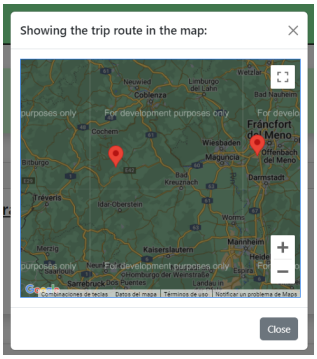

Figure 8. Screen opened where you can see the route of a trip.

The 'Create a trip' section is a form that requires the user to input all necessary details about the trip they wish to create. Additionally, the starting and ending points can be manually specified on the map. The information given is validated using Angular validators.

The map that is used in Carida is providad by Google and let the user navigate throght it.

the 'My Trips' sections let the user check all the Trips he has booked and let him cancel them if necessary. The page is so similar to tha page seen in Figure 7 but the button 'Book a seat' is red with the text 'Cancel reservation'.

Carida offers users a convenient and efficient way to discover and join trips, as well as cancel their reservation if necessary. Integration with Google Maps enables users to preview the trip route and evaluate if it meets their requirements before reserving a seat. This functionalities improves the overall user experience and expands the accessibility of car sharing.

5. DISCUSSION

We present an application [9] that addresses a significant challenge in car sharing. Our system uses dummy data to provide an efficient user experience for travel coordination.

Although our app is only a first version, we acknowledge that there is a lot of room for improvement. For example, adding a backend would allow users to manage real data, and incorporating online payments would be a valuable feature. Additionally, developing iOS and Android applications would increase the app's accessibility and user base, as people tend to prefer mobile apps over websites.

Our experience with the Angular framework has been very positive. We found it to be a powerful tool for web application development, with its organized structure making it easier to maintain and scale our code. While we don't have sufficient experience with React to make a direct comparison, we believe that Angular is a great choice for web app development. The framework can be confusing at first, but once you become familiar with the workflow, it can be a powerful and efficient tool.

However, it is worth noting that Angular is not designed for native mobile app development. For this purpose, other options like Flutter or React Native would be more appropriate.

While our paper focuses on the technical aspects of our application, we recognize that car sharing has important social and ethical implications. As such, we believe that it is important for future research to consider how applications like ours can address these issues.

6. REFERENCES

[1] "Angular official site," https://angular.io/docs, accessed: 2023-03-29.

[2] "Angular Wikipedia site," https://en.wikipedia.org/wiki/Angular_(web_framework), accessed: 2023-03-29.

[3] "Setting up the local environment and workspace for Angular," https://angular.io/guide/setup-local, accessed: 2023-03-29.

[4] "Create a new Angular project," https://angular.io/tutorial/tour-of-heroes/toh-pt0, accessed: 2023-03-29.

[5] B. Gauca, "Angular vs. React - A comparison," http://work.haufegroup.io/Angular-VS-React/, 2017, [Online; accessed 29-March-2023].

[6] "Amigoscode Youtube channel," https://www.youtube.com/@amigoscode, accessed: 2023-03-29.

[7] "freeCodeCamp.org Youtube channel," https://www.youtube.com/@freecodecamp, accessed: 2023-03-29.

[8] "TraversyMedia Youtube channel," https://www.youtube.com/@TraversyMedia, accessed: 2023-03-29.

[9] L. M. G. Marín and J. M. Durán, "Car Sharing App (BWMA-Carida)," https://github.com/jesu-m0/BWMA-Carida, 2022, [Online; accessed 29-March-2023].