Stack Documentation

Contents

1 Executive Summary

The purpose of this document is to provide information on the full stack of the system and application. It does into depth on the development language, Angular that was utilized and a brief overview of how to get started with it. It briefly describes Node.js, PostgreSQL, and CentOS. These were all platforms used for the creation of the project.

2 Angular

2.1 Getting Started

For the purposes of this project, Angular 4.0.0 is being utilized.

Angular is a fairly new and dynamic development framework that makes it easy to build applications for the web, a mobile device, or desktop. Angular uses a variety of templates, components, and injections to seamlessly solve development problems, and build applications.

In order to develop in Angular, the development environment must include Node.js and an npm package.

Step 1: Make sure environment includes Node.js and npm manager
Angular requires Node.js version 8.x or 10.x. In order to check the development systems version, run the command below:

```
# node -v
```

If node.js is not downloaded on the system, go to `www.nodejs.org` to download. Step-by-step instructions on how to do so are in the **Deployment Plan** section of this documentation.

Step 2: Install Angular CLI
In order to start developing in Angular, one must download the **Angular CLI**. **Angular CLI** is a command line interface that is used by Angular to scaffold and build projects and applications. It is important to download the Angular CLI globally. To do so, enter the following command in the terminal:

```
# npm install -g @angular/cli
```

Step 3: Create workspace and inital application
It is necessary to create an Angular workspace to hold the new project. In order to create a new project, enter the following command:

```
# ng new my-project
```

"My-project" can be any name that the developer wants to name their project.

The Angular CLI creates all of the necessary files, folders, npm packages and dependencies for a new Angular project. It creates a new workspace and the following starter files:

- A new workspace, with a root folder named my-project
- An initial skeleton app project, also called my-app (in the src subfolder)
- An end-to-end test project (in the e2e subfolder)
- Related configuration files

Step 4: 'Serve' or 'run' the application
To start up the starter application that Angular CLI created, one must follow these steps: 1. Go to the workspace folder "my-project". In order to do this, use the commands:

```
# ls
```

(To look at all of the files and folders that can be chosen)

```
# cd folderName
```

(To enter one of those folders.)

Alternate between these two commands until in the folder **"my-project"**.

Once in the folder, run the following command to open the starter application:

```
# ng serve --open
```

(**ng serve** is the command that launches the server, watches your files, and rebuilds the app as you make changes to those files.)

The –open (or just -o) option automatically opens your browser to **http://localhost:4200/**.

Step 5: Edit your first Angular component
Components are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and take action based on that input.
As part of the initial app, the CLI created the first Angular component for you. It is the root component, and it is named app-root.

1. Open: ./src/app/app.component.ts.

2. Change the title property from **'my-app'** to **'My First Angular App'**.

src/app/app.component.ts

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'My First Angular App!';
}
```

3. Open: ./src/app/app.component.css and give the component some style.

src/app/app.component.css

```
h1 {
  color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
}
```

2.2 Architecture

The architecture of the application is written in Angular which is a platform and framework for building client applications in HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

- The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.
- The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).

An app's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

2.3 NgModules

NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities. They can contain components, service providers, and other code files whose scope is defined by the containing NgModule. They can import functionality that is exported from other NgModules, and export selected functionality for use by other NgModules.

Every Angular app has at least one NgModule class, the root module, which is conventionally named AppModule and resides in a file named app.module.ts. You launch your app by bootstrapping the root NgModule.

While a small application might have only one NgModule, most apps have many more feature modules. The root NgModule for an app is so named because it can include child NgModules in a hierarchy of any depth.

2.4 Angular libraries

Angular loads as a collection of JavaScript modules. You can think of them as library modules. Each Angular library name begins with the @angular prefix. Install them with the npm package manager and import parts of them with JavaScript import statements.

For example, import Angular's Component decorator from the @angular/core library like this.

```
import { Component } from '@angular/core';
```

2.5 Components:

A component controls a patch of screen called a 'view'.

You define a component's application logic—what it does to support the view—inside a class. The class interacts with the view through an API of properties and methods.

src/app/hero-list.component.ts (class)

```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

Angular creates, updates, and destroys components as the user moves through the application. Your app can take action at each moment in this lifecycle through optional lifecycle hooks, like ngOnInit().

3 Node.js

What is Node.js?

- Node.js is an open source server environment.
- Node.js is free of cost. For the purposes of this project, all software is required to be free, as it is for a non-profit organization.
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.).
- Node.js uses JavaScript on the server.

What can Node.js do?

- Node.js can generate dynamic page content.
- Node.js can create, open, read, write, delete, and close files on the server.
- Node.js can collect form data.
- Node.js can add, delete, modify data in your database.

Details on how to install Node.js are in the ***Deployment Plan*** *section of this guide.*

4 PostgreSQL

What is PostgreSQL?
PostgreSQL is a general purpose and object-relational database management system, the most advanced open source database system.

PostgreSQL has many advanced features that other enterprise database management systems offer, such as:

- User-defined types
- Table inheritance
- Sophisticated locking mechanism
- Foreign key referential integrity
- Views, rules, subquery
- Nested transactions (savepoints)
- Multi-version concurrency control (MVCC)
- Asynchronous replication

5 CentOS7

What is CentOS7?
The CentOS Linux distribution is a stable, predictable, manageable and reproducible platform derived from the sources of Red Hat Enterprise Linux (RHEL).

The CentOS Project is a community-driven free software effort focused around the goal of providing a rich base platform for open source communities to build upon.

6 Stack