valtech_
mobility

ANGULAR

WHAT IS ANGULAR

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps.

DETAILS ON ANGULAR

- Build on TypeScript
- Component-based framework
- Includes collection of well-integrated libraries
- A suite of developer tools to help you develop, build, test, and update your code

INSTALLATION

ANGULAR CLI

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

CREATE PROJECT

Before adding features or libraries like Bootstrap

```
ng new <application-name>
```

INSTALL BOOTSTRAP

Optional step if you want to use Bootstrap with Angular

Run in your Project directory

```
npm install bootstrap
ng add @ng-bootstrap/ng-bootstrap
```

INSTALL BOOTSTRAP

Optional step if you want to use Bootstrap with Angular

Run in your Project directory

```
npm install bootstrap
ng add @ng-bootstrap/ng-bootstrap
```

CONFIG BOOTSTRAP CSS

Make sure your styles.scss contains

```
@import "~bootstrap/scss/bootstrap";
```

RUN YOUR APP

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

Run in your Project directory

```
ng serve --open
```

EMBRACE YOUR WORK AND DRINK COFFEE

GET BACK TO WORK !!

CREATE LIST PAGE

- Create a Superhuman Interface
- Create a List Component

```
ng generate interface <InterfaceName>
ng generate component <ComponentName>
```

*NGFOR

```
<h2>All Superhumans</h2>
<ul class="heroes">
  <li *ngFor="let sup of heroes">
    <span class="badge">{{sup.id}}</span> {{sup.name}}
  </li>
</ul>
```

SELECT A SUPERHUMAN

Add this to your List Component HTML

```
<h2>{{selectedSuperhuman.name}} Details</h2>
<div><span>id: </span>{{selectedSuperhuman.id}}</div>
<div>
    <label for="hero-name">Hero name: </label>
    <input id="hero-name" [(ngModel)]="selectedSuperhuman.name" p
</div>
```

SELECT A SUPERHUMAN

Add this to your List Component TS

```
selectedSuperhuman?: Superhuman;
onSelect(sup: Superhuman): void {
    this.selectedSuperhuman = sup;
  }
```

BROKEN ? WHY ?

You try to access a variable that isn’t set when you enter the page

*NGIF

Suround your Detail View with a div tag and add a ngIf

```
<div *ngIf="selectedSuperhuman">
  <h2>{{selectedSuperhuman.name | uppercase}} Details</h2>
  <div><span>id: </span>{{selectedSuperhuman.id}}</div>
  <div>
    <label for="hero-name">Hero name: </label>
    <input id="hero-name" [(ngModel)]="selectedSuperhuman.name" p
  </div>
</div>
```

EXTRACT THE DETAILS

- Create Detail Component
- Copy html from List to Detail Component
- Integrate the Detail Component and pass the selected Superhuman

PASSING DATA TO COMPONENTS

Add this to the Detail js

```
@Input() sup?:Superhuman;
```

PASSING DATA TO COMPONENTS

Add this to List html

```
<app-superhuman-edit [sup]="selectedSuperhuman"></app-superhuman-
```

SERVICES

- Create Superhuman Service
- Remove Superhuman List from List Component and add it to the service

```
ng g s <ServiceName>
```

SERVICES

- User service to get Data in the List Components Init Method

```
sups?: Superhuman[];

  constructor(private supService: SuperhumanService) {
  }

  ngOnInit(): void {
    this.sups = this.supService.getSuperhumans();
  }
```

NAVIGATION

Angular uses Routes and Router to move between Screens

ROUTES

- Open app-routing.module.ts
- Define a route to your detail component

```
-----app-routing.module.ts-------
const routes: Routes = [
  {path:'detail',component:SuperhumanDetailComponent}
];
```

ROUTES

- Add router outlet to your app.component.html
- Remove the List tag

```
-----app.component.html-------
<div class="container">
  <h1>Welcome to the SuperhumanDB</h1>
  <router-outlet></router-outlet>
</div>
```

ROUTES

- Add link to open List Component

```
-----app.component.html-------
<div class="container">
  <h1>Welcome to the SuperhumanDB</h1>
  <nav class="pb-4 font-weight-bolder">
    <a routerLink="/list">List</a>
  </nav>
  <router-outlet></router-outlet>
</div>
```

ROUTES

- Extract Detail Component to show seperatly from the list component
- Create a back button to return to the List page from the detail page

PIPES

- Add dead boolean to Superhumen interface
- Kill some superhumans (MUHAHAHAHAHA)

PIPES

- Filter the List of superhumans to show only alive superhumans
- Create Pipe to filter the list

```
ng g pipe <PipeName>
```

PIPES

- Add a button to list component to switch between dead and alive

GET DATA FROM A SERVER

- Open Shell and navigate to given jar
- Start the given jar with

```
java –jar <JarName>
```

GET DATA FROM A SERVER

- Enable HTTP Services

```
------app.modules.ts--------
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  imports: [
    HttpClientModule,
  ],
})
```

GET DATA FROM A SERVER

- Modify the SuperhumanService to get the Data from the server

```
constructor(private http: HttpClient) { }
```

GET DATA FROM A SERVER

- Get data from server with HttpClient

```
private baseUrl: string = "http://localhost:8015/superhuman";

  constructor(private http: HttpClient) {
  }

  getSuperhumans(): Observable<Superhuman[]> {
    return this.http.get<Superhuman[]>(this.baseUrl + "/all")
  }
```

GET DATA FROM A SERVER

- Append Superhuman interface
- Append App logic to work with Observable
- Show all the superhuman data in the Detail Screen

SEND DATA TO SERVER

- Create a Edit Component
- Add a link to the Edit Component to the nav bar

REACTIVE FORMS

- Implement a Edit Form for superhumans
- Add a submit button

```
<form (ngSubmit)="onSubmit()" [formGroup]="supForm"> <== onSubmit
  <div class="form-group">
    <label for="name">Name</label>
    <input class="form-control" type="text" id="name" formControl
    <div [hidden]="!submitted||name?.valid"
         class="alert alert-danger">
      Name is required
    </div>
  </div>
</form>
```

REACTIVE FORMS

```
-----edit-component.ts-------
constructor(private fb: FormBuilder) {
    this.supForm = fb.group({
      name: ['']
    });
}
```

REACTIVE FORMS VALIDATION

- Make name required
- Add Input for Power and make sure the value is greater than 0

VALIDATORS

```
-----edit-component.ts-------
constructor(private fb: FormBuilder) {
    this.supForm = fb.group({
      name: ['', Validators.required],
      story: ['',Validators.maxLength(255)],
      heroType: ['', Validators.required],
      power: ['', [Validators.required, powerValidator]]
    });
}
```

SEND DATA TO SERVER

- Implement a POST Request in the superhuman service
- Send data in the onSubmit Method

CUSTOM VALIDATORS

```
-----edit-component.ts-------
function powerValidator(control: FormControl): { [key: string]: b
      console.log(control.value);
      if (control.value !== undefined && control.value >= 0) {
        return null;
      }
      return {'power': true};
    }
```

BASE IS DONE

IMPLEMENT SOME GAME LOGIC

FIGHTS

- Create a Component that shows a superhuman
- Add three buttons (Strength, Power, Intelligence)
- Create a fight component that includes two of the components above

FIGHTS

Basic Rules:

- One starts as the attacker and one as the defender
- Roundbased combat (attacker goes first)
- Round
  - 1. Attacker chooses trait (Intelligence, Power, Strength) to attack
  - 2. Defender defends with same trait

FIGHTS

Basic Rules Damage:

- 3. Damage is the difference between both trait values.
  - (Attacker Intel = 5, Defender Intel = 3, Defender gets damage of 2)
  - (Attacker Intel = 1, Defender Intel = 5, Defender blocks attack and takes no damage)

FIGHTS

Basic Rules End of Phase:

- 4. Defender is Attacker now and round starts again
- 5. If the damage taken by one contest equal or greater than his/her maximum health the oponent wins

EVENTS

- Send Attack Event to Defender

EVENTS

- Create a service that handles the attack

```
export class AttackService {

  attack: Attack = new Attack(AttackTrait.INTELLIGENCE, true);
  attackData: BehaviorSubject<Attack>;

  constructor() {
    this.attackData = new BehaviorSubject(this.attack);
  }

   performAttack(trait: AttackTrait, isAttacker: boolean) {
    this.attack = new Attack(trait, isAttacker);
    this.attackData.next(this.attack);
  }
}
```

LOCALSTORAGE

- Create a service that saves every attack in the localstorage
- Show the content of the localstorage on a page

```
localStorage.setItem('dataSource', this.dataSource.length);
console.log(localStorage.getItem('dataSource'));
```

(Hint: Use a service to handle localstorage calls)

MORE POSSIBLE LOGIC

EXTENDED FIGHT

- HeroTypes add lose condition
  - If a hero loses against a hero there is a 20% chance that the loser needs a hospital visit
  - If a hero loses against a villain (or the other way around) the loser has a 60% chance to die and a 40% Chance to land in the hospital

EXTENDED FIGHT WEAPONS

- Superhumans can use Weapons
  - Weapons trait get added to the superhuman traits
  - Weapons can only be used as often as their max use value

HOSPITAL STAY LOGIC

- If a superhuman is in Hospital they can’t fight
- Show all superhumans that are in hospital

FIGHTS ++

- Add randomness to fights (e.g. Multipliers for traits that get rolled every attack)