

Home > Programming Blog > Angular

Angular 9 Tutorial: Learn to Build a CRUD Angular App Quickly

by Didin J. on Feb 12, 2020



A comprehensive step by step Angular 9 tutorial on learning to build a CRUD (create, read, update, delete) Angular app quickly

Programming Blog

Vue.js

In this Angular 9 tutorial, we will create an Angular app of Coronavirus cases which basically implementing CRUD operation. This app will use REST API that gets the required data, post the case data, put case changes, and delete case data. So, we will learn how to use HTTPClient to access REST API, using Angular Form, using Routing and Navigation, and create a simple Calendar.

This tutorial divided into several steps:

- [Step #1. Create a New Angular 9 App](#)
- [Step #2. Add the Angular 9 Routing and Navigation](#)
- [Step #3. Add the Angular 9 Service](#)
- [Step #4. Display List of Data using Angular Material](#)
- [Step #5. Show and Delete Data Details using Angular Material](#)
- [Step #6. Show Statistic using Ng2Charts and Chart.js](#)
- [Step #7. Add a Data using Angular Material Form](#)
- [Step #8. Edit a Data using Angular Material Form](#)
- [Step #9. Run and Test the Angular 9 Coronavirus Cases App](#)

In the end, the final Angular 9 Coronavirus Cases app will look like this.

[HTML 5 Tutorial](#)

[Flutter Tutorial](#)

[Javascript](#)

[Node.js](#)

[ASP.NET Core](#)

[Groovy and Grails](#)

[React Native](#)

[React.js](#)

[Java](#)

[MongoDB](#)

[CSS 3](#)

[Ionic Framework](#)

[Angular](#)

[All Articles](#)

Popular Articles:

- [Angular 8 Tutorial: REST API and HttpClient Examples \(49558\)](#)
- [Angular 8 Tutorial: Routing & Navigation Example \(28304\)](#)
- [Angular 8 Tutorial: Observable and RXJS](#)

Examples (24724)

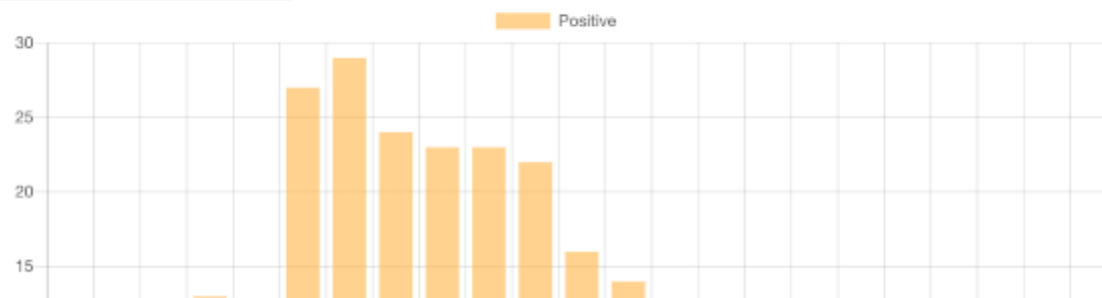
- [Angular 7 Tutorial: Building CRUD Web Application \(13321\)](#)
- [Angular 6 HttpClient: Consume RESTful API Example \(11968\)](#)
- [Angular 8 Tutorial: Learn to Build Angular 8 CRUD Web App \(9622\)](#)
- [Angular 8 Tutorial: How to Create an Angular Web App Quickly \(6571\)](#)
- [Angular Material Form Controls, Form Field and Input Examples \(6456\)](#)
- [Push Notification using Ionic 4 and Firebase Cloud Messaging \(6048\)](#)
- [React Native Firebase Cloud Messaging \(FCM\) Push Notification \(5853\)](#)
- [React Native Tutorial: SQLite Offline Android/iOS Mobile App \(5823\)](#)
- [Ionic 4 Angular 8 Tutorial: Learn to Build CRUD Mobile Apps \(5788\)](#)
- [Ionic 4, Angular 7 and Cordova Tutorial: Build CRUD Mobile Apps \(5709\)](#)
- [Angular 8 Tutorial: Facebook Login \(4527\)](#)
- [Building Web App using ASP.NET Web API Angular 7 and SQL Server \(4282\)](#)

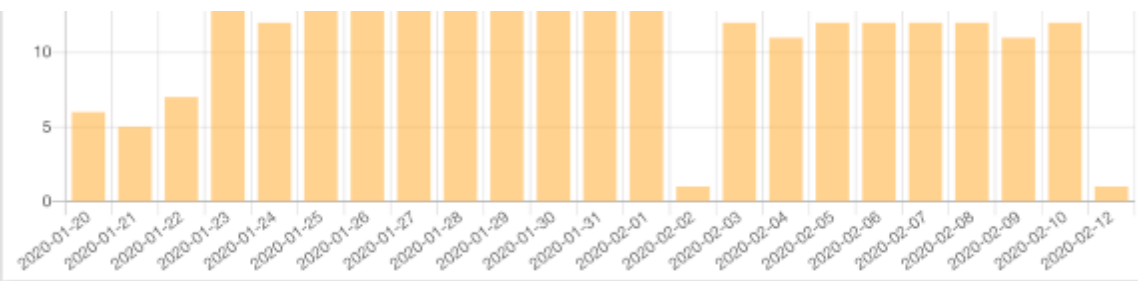
Corona Virus Cases List

[+ Cases](#)[Statistic](#)

Cases Name	Age	Status
An Kang	56	Dead
Benjamin Zhengmin Pan	55	Dead
Bill Liu	78	Recovered
Cai Dongchen	43	Positive
Cai Hongbin	64	Positive
Cai Kui	68	Positive
Cao Ji	39	Positive
Cao Longxiang	35	Positive
Cen Junda	34	Positive
Chan Laiwa	41	Positive
Chanchai Ruayrungruang	49	Positive

Corona Virus Cases Statistic

[Positive](#)[Dead](#)[Recovered](#)



Corona Virus Cases Details



Jack Ma

56 year old

Gender:

Male

Address:

Unknown

City:

Wuhan

Country:

China

Status:

Recovered



Cases



Cases

Coronavirus Add Cases



Name

Gender



Age

Address

City

Country

Status



Coronavirus Edit Cases



Name

Cai Dongchen

Gender

Male

Age

43

Address

Unknown

City


Wuhan

Country

China

Status

Positive



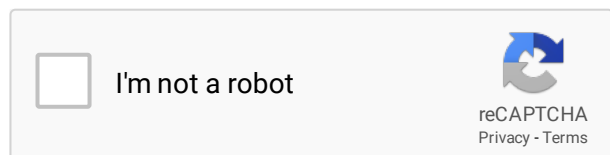
The following tools, frameworks, and modules are required for this tutorial:

1. [Node.js \(recommended version\)](#)
2. [Angular 9](#)
3. [Coronavirus REST API](#)
4. Terminal (Mac/Linux) or Node Command Line (Windows)
5. IDE or Text Editor (We are using Visual Studio Code)

We already provided the REST API for the Coronavirus cases, you can just clone, install NPM, run MongoDB server, and Run the Express/MongoDB REST API server. We assume that you have installed Node.js. Now, we need to check the Node.js and NPM versions. Open the terminal or Node command line then type these commands.

```
node -v
v10.15.1
npm -v
6.13.6
```

You can watch the video tutorial about this on our YouTube channel.



About this page

Our systems have detected unusual traffic from your computer network. This page checks to see if it's really you sending the requests, and not a robot. [Why did this happen?](#)

IP address: 2600:3c02::f03c:92ff:fea1:3ea0
Time: 2020-02-15T09:42:57Z
URL: <https://www.youtube.com/embed/R4224MsyckA>

Step #1. Create a New Angular 9 App

We will create a new Angular 9 App using Angular CLI. For that, we need to install or update the @angular/cli first to the latest version.

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


Next, create a new Angular 9 app by running this command.

```
ng new coronavirus-cases
```

If you get the question like below, choose `Yes` and `SCSS` (or whatever you like to choose).

```
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS [ https://sass-lang.
com/
documentation/syntax#scss ]
```

Next, go to the newly created Angular 9 project folder.

```
cd coronavirus-cases
```

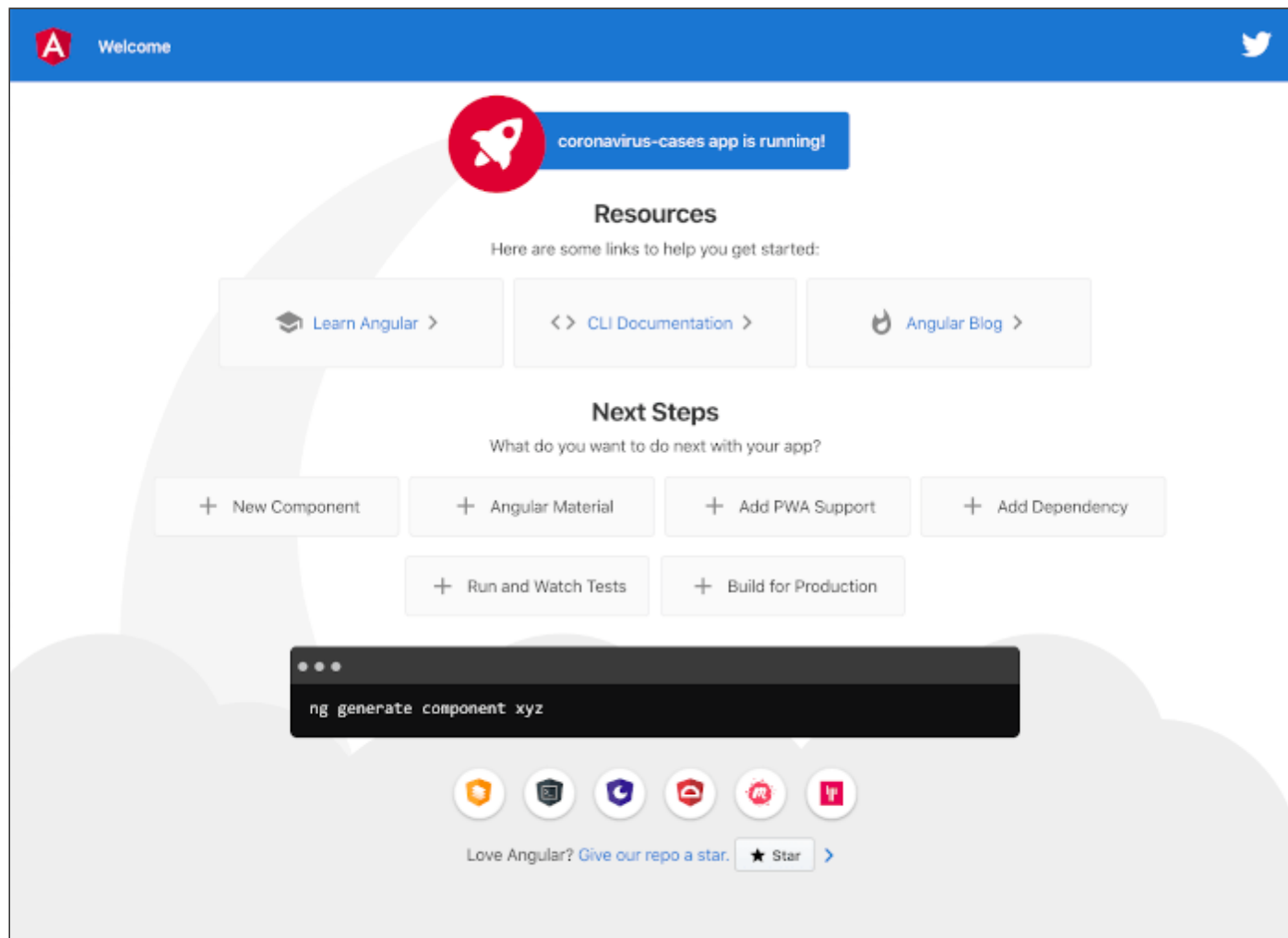
Open this Angular project with your IDE or Text editor. To use VSCode type this command.

```
code .
```

Type this command to run the Angular 9 app for the first time.

```
ng serve --open
```

Using the "--open" parameter will automatically open this Angular 9 app in the default browser. Now, the Angular initial app looks like this.



Step #2. Add the Angular 9 Routing and Navigation

As you see in the first step of creating an Angular 9 app. We already add the Angular Routing for this Angular 9 app. Next, we just add the required Angular components for this Coronavirus cases app. Just type these commands to generate them.

```
ng g component cases
ng g component cases-details
ng g component add-cases
ng g component edit-cases
ng g component cases-stat
```

Those components will automatically be registered to the app.module.ts. Next, open and edit `src/app/app-routing.module.ts` then add these imports.

```
import { CasesComponent } from './cases/cases.component';
import { CasesDetailsComponent } from './cases-details/cases-details.component';
import { CasesStatComponent } from './cases-stat/cases-stat.component';
import { AddCasesComponent } from './add-cases/add-cases.component';
import { EditCasesComponent } from './edit-cases/edit-cases.component';
```

Add these arrays to the existing routes constant that contain route for above-added components.

```
const routes: Routes = [
  {
    path: 'cases',
    component: CasesComponent,
    data: { title: 'List of Cases' }
  },
  {
    path: 'cases-details/:id',
    component: CasesDetailsComponent,
    data: { title: 'Cases Details' }
  },
  {
    path: 'cases-stat',
    component: CasesStatComponent,
    data: { title: 'Cases Statistic' }
  },
  {
    path: 'add-cases',
    component: AddCasesComponent,
    data: { title: 'Add Cases' }
  }
];
```

```
},
{
  path: 'edit-cases/:id',
  component: EditCasesComponent,
  data: { title: 'Edit Cases' }
},
{ path: '',
  redirectTo: '/cases',
  pathMatch: 'full'
}
];
```

Open and edit `src/app/app.component.html` and you will see the existing router outlet. Next, modify this HTML page to fit the CRUD page.

```
<div class="container">
  <router-outlet></router-outlet>
</div>
```

Open and edit `src/app/app.component.scss` then replace all SASS codes with this.

```
.container {
  padding: 20px;
}
```

Step #3. Add the Angular 9 Service

All-access (POST, GET, PUT, DELETE) to the REST API will put in the Angular 9 Service. The response from the REST API emitted by Observable that can subscribe and read from the Components. Before creating a service for REST API access, first, we have to install or register `HttpClientModule`. Open and edit `src/app/app.module.ts` then add these imports of FormsModule, ReactiveFormsModule (@angular/forms) and HttpClientModule (@angular/common/http).

```
import { FormsModule, ReactiveFormsModule } from '@angular/forms';  
import { HttpClientModule } from '@angular/common/http';
```

Add it to `@NgModule` imports after `BrowserModule`.

```
imports: [  
  BrowserModule,  
  FormsModule,  
  ReactiveFormsModule,  
  HttpClientModule,  
  AppRoutingModule  
],
```

We will use the type specifier to get a typed result object. For that, create a new Typescript file `src/app/cases.ts` then add these lines of Typescript codes.

```
export class Cases {  
  _id: string;  
  name: string;  
  gender: string;  
  age: number;  
  address: string;  
  city: string;  
  country: string;  
  status: string;  
  updated: Date;  
}
```

And create a new Typescript file `src/app/statistic.ts` then add these lines of Typescript codes.

```
export class Statistic {  
  _id: any;  
  count: number;  
}
```

Next, generate an Angular 9 service by typing this command.

```
ng g service api
```

Next, open and edit `src/app/api.service.ts` then add these imports.

```
import { Observable, of, throwError } from 'rxjs';
import { HttpClient, HttpHeaders, HttpResponse } from '@angular/common/http';
import { catchError, tap, map } from 'rxjs/operators';
import { Cases } from '../cases';
import { Statistic } from '../statistic';
```

Add these constants before the `@Injectable`.

```
const httpOptions = {
  headers: new HttpHeaders({'Content-Type': 'application/json'})
};
const apiUrl = '/api/';
```

Inject the `HttpClient` module to the constructor.

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

Add the error handler function that returns as an Observable.

```
private handleError<T> (operation = 'operation', result?: T) {
  return (error: any): Observable<T> => {

    // TODO: send the error to remote logging infrastructure
    console.error(error); // log to console instead
```



```

    // Let the app keep running by returning an empty result.
    return of(result as T);
  };
}

```

Add the functions for all CRUD (create, read, update, delete) REST API call of cases and statistic data.

```

getCases(): Observable<Cases[]> {
  return this.http.get<Cases[]>(`${apiUrl}`)
    .pipe(
      tap(cases => console.log('fetched cases')),
      catchError(this.handleError('getCases', []))
    );
}

getCasesById(id: string): Observable<Cases> {
  const url = `${apiUrl}/${id}`;
  return this.http.get<Cases>(url).pipe(
    tap(_ => console.log(`fetched cases id=${id}`)),
    catchError(this.handleError<Cases>(`getCasesById id=${id}`))
  );
}

addCases(cases: Cases): Observable<Cases> {
  return this.http.post<Cases>(apiUrl, cases, httpOptions).pipe(
    tap((c: Cases) => console.log(`added cases w/ id=${c._id}`)),
    catchError(this.handleError<Cases>('addCases'))
  );
}

updateCases(id: string, cases: Cases): Observable<any> {
  const url = `${apiUrl}/${id}`;
  return this.http.put(url, cases, httpOptions).pipe(
    tap(_ => console.log(`updated cases id=${id}`)),
    catchError(this.handleError<any>('updateCases'))
  );
}

deleteCases(id: string): Observable<Cases> {

```

```

const url = `${apiUrl}/${id}`;
return this.http.delete<Cases>(url, httpOptions).pipe(
  tap(_ => console.log(`deleted cases id=${id}`)),
  catchError(this.handleError<Cases>('deleteCases'))
);
}

getStatistic(status: string): Observable<Statistic> {
  const url = `${apiUrl}/daily/${status}`;
  return this.http.get<Statistic>(url).pipe(
    tap(_ => console.log(`fetched statistic status=${status}`)),
    catchError(this.handleError<Statistic>('getStatistic status=${status}'
`))
  );
}

```

You can find more examples of Angular 9 Observable and RXJS [here](#).

Step #4. Display List of Data using Angular Material

We will display the list of data using the Angular Material Table. The data published from the API service read by subscribing as a Cases model in the Angular 9 component. For that, open and edit ``src/app/cases/cases.component.ts`` then add this import of the previously created API Service.

```
import { ApiService } from '../api.service';
```

Next, inject the API Service to the constructor.

```
constructor(private api: ApiService) { }
```

Next, for the user interface (UI) we will use Angular Material and CDK. There's a CLI for generating a Material component like Table as a

component, but we will create or add the Table component from scratch to the existing component. Type this command to install Angular Material (@angular/material).

```
ng add @angular/material
```

If there are questions like below, just use the default and "Yes" answer.

```
? Choose a prebuilt theme name, or "custom" for a custom theme: Indigo/Pink
  [ Preview: https://material.angular.io?theme=indigo-pink ]
? Set up global Angular Material typography styles? Yes
? Set up browser animations for Angular Material? Yes
```

We will register all required Angular Material components or modules to `src/app/app.module.ts`. Open and edit that file then add these imports of required Angular Material Components.

```
import { MatInputModule } from '@angular/material/input';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatSortModule } from '@angular/material/sort';
import { MatTableModule } from '@angular/material/table';
import { MatIconModule } from '@angular/material/icon';
import { MatButtonModule } from '@angular/material/button';
import { MatCardModule } from '@angular/material/card';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatSliderModule } from '@angular/material/slider';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
import { MatButtonModule } from '@angular/material/button-toggle';
import { MatSelectModule } from '@angular/material/select';
```

Register the above modules to `@NgModule` imports.

```
imports: [  
  ...  
  MatInputModule,  
  MatPaginatorModule,  
  MatProgressSpinnerModule,  
  MatSortModule,  
  MatTableModule,  
  MatIconModule,  
  MatButtonModule,  
  MatCardModule,  
  MatFormFieldModule,  
  MatSliderModule,  
  MatSlideToggleModule,  
  MatButtonModule,  
  MatSelectModule,  
],
```

Next, back to `src/app/cases/cases.component.ts` then add this import.

```
import { Cases } from '../cases';
```

Declare the variables of Angular Material Table Data Source before the constructor.

```
displayedColumns: string[] = ['name', 'age', 'status'];  
data: Cases[] = [];  
isLoadingResults = true;
```

Modify the `ngOnInit` function to get a list of cases immediately.

```
ngOnInit(): void {  
  this.api.getCases()  
    .subscribe((res: any) => {  
      this.data = res;  
      console.log(this.data);  
      this.isLoadingResults = false;  
    }, err => {  
      console.log(err);  
      this.isLoadingResults = false;  
    });  
}
```

```
});  
}
```

Next, open and edit `src/app/cases/cases.component.html` then replace all HTML tags with this Angular Material tags.

```
<div class="example-container mat-elevation-z8">  
  <h2>Corona Virus Cases List</h2>  
  <div class="example-loading-shade"  
    *ngIf="isLoadingResults">  
    <mat-spinner *ngIf="isLoadingResults"></mat-spinner>  
  </div>  
  <div class="button-row">  
    <a mat-flat-button color="primary" [routerLink]="['/add-cases']"><mat-ic  
on>add</mat-icon> Cases</a>  
    <a mat-flat-button color="accent" [routerLink]="['/cases-stat']"><mat-ic  
on>bar_chart</mat-icon> Statistic</a>  
  </div>  
  <div class="mat-elevation-z8">  
    <table mat-table [dataSource]="data" class="example-table"  
      matSort matSortActive="name" matSortDisableClear matSortDirection  
="asc">  
  
      <!-- Cases Name Column -->  
      <ng-container matColumnDef="name">  
        <th mat-header-cell *matHeaderCellDef>Cases Name</th>  
        <td mat-cell *matCellDef="let row">{{row.name}}</td>  
      </ng-container>  
  
      <!-- Cases Age Column -->  
      <ng-container matColumnDef="age">  
        <th mat-header-cell *matHeaderCellDef>Age</th>  
        <td mat-cell *matCellDef="let row">{{row.age}}</td>  
      </ng-container>  
  
      <!-- Cases Status Column -->  
      <ng-container matColumnDef="status">  
        <th mat-header-cell *matHeaderCellDef>Status</th>  
        <td mat-cell *matCellDef="let row">{{row.status}}</td>  
      </ng-container>
```

```
<tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
<tr mat-row *matRowDef="let row; columns: displayedColumns;" [routerLi
nk]="['/cases-details/', row._id]"></tr>
</table>
</div>
</div>
```

Finally, to make a little UI adjustment, open and edit
`src/app/cases/cases.component.scss` then add these CSS codes.

```
/* Structure */
.example-container {
  position: relative;
  padding: 5px;
}

.example-table-contai
```

Step #5. Show and Delete Data Details using Angular Material

On the list page, there are 2 buttons to navigate to the Details and Statistic page. For, Details page the button action also sends an ID parameter. Next, open and edit `src/app/cases-details/cases-details.component.ts` then add these lines of imports.

```
import { ActivatedRoute, Router } from '@angular/router';
import { ApiService } from '../api.service';
import { Cases } from '../cases';
```

Inject the above modules to the constructor.

```
constructor(private route: ActivatedRoute, private api: ApiService, private
router: Router) { }
```

Declare the variables before the constructor for hold cases data that get from the API.

```
cases: Cases = { _id: '', name: '', gender: '', age: null, address: '', ci
ty: '', country: '', status: '', updated: null };
isLoadingResults = true;
```

Add a function for getting Cases data from the API.

```
getCasesDetails(id: string) {
  this.api.getCasesById(id)
    .subscribe((data: any) => {
      this.cases = data;
      console.log(this.cases);
      this.isLoadingResults = false;
    });
}
```

Call that function when the component is initiated.

```
ngOnInit(): void {
  this.getCasesDetails(this.route.snapshot.params.id);
}
```

Add this function to delete a case.

```
deleteCases(id: any) {
  this.isLoadingResults = true;
  this.api.deleteCases(id)
    .subscribe(res => {
      this.isLoadingResults = false;
      this.router.navigate(['/cases']);
    }, (err) => {
      console.log(err);
      this.isLoadingResults = false;
    })
};
}
```

For the view, open and edit `src/app/cases-details/cases-details.component.html` then replace all HTML tags with this.

```
<div class="example-container mat-elevation-z8">
  <h2>Corona Virus Cases Details</h2>
  <div class="example-loading-shade"
    *ngIf="isLoadingResults">
    <mat-spinner *ngIf="isLoadingResults"></mat-spinner>
  </div>
  <div class="button-row">
    <a mat-flat-button color="primary" [routerLink]="['/cases']"><mat-icon>list</mat-icon></a>
  </div>
  <mat-card class="example-card">
    <mat-card-header>
      <mat-card-title><h2>{{cases.name}}</h2></mat-card-title>
      <mat-card-subtitle>{{cases.age}} year old</mat-card-subtitle>
    </mat-card-header>
    <mat-card-content>
      <dl>
        <dt>Gender:</dt>
        <dd>{{cases.gender}}</dd>
        <dt>Address:</dt>
        <dd>{{cases.address}}</dd>
        <dt>City:</dt>
```



```

        <dd>{{cases.city}}</dd>
        <dt>Country:</dt>
        <dd>{{cases.country}}</dd>
        <dt>Status:</dt>
        <dd><h2>{{cases.status}}</h2></dd>
    </dl>
</mat-card-content>
<mat-card-actions>
    <a mat-flat-button color="primary" [routerLink]="['/edit-cases', case
s._id]"><mat-icon>edit</mat-icon> Cases</a>
    <a mat-flat-button color="warn" (click)="deleteCases(cases._id)"><mat-
icon>delete</mat-icon> Cases</a>
</mat-card-actions>
</mat-card>
</div>

```

Finally, open and edit `src/app/cases-details/cases-details.component.scss` then add this lines of CSS codes.

```

/* Structure */
.example-container {
  position: relative;
  padding: 5px;
}

.example-loading-shade {
  position: absolute;
  top: 0;
  left: 0;
  bottom: 56px;
  right: 0;
  background: rgba(0, 0, 0, 0.15);
  z-index: 1;
  display: flex;
  align-items: center;
  justify-content: center;
}

.mat-flat-button {
  margin: 5px;
}

```

Step #6. Show Statistic using Ng2Charts and Chart.js

We will use a bar chart to display the statistic of Coronavirus cases. So, we need to install Ng2Charts and Chart.js modules by type this command.

```
npm i --save ng2-charts chart.js
```

Next, open and edit `src/app/app.module.ts` then add this import of ng2-charts.

```
import { ChartsModule } from 'ng2-charts';
```

Add this module to the @NgModule imports.

```
imports: [  
  ...  
  ChartsModule  
],
```

Next, open and edit `src/app/cases-stat/cases-stat.component.ts` then add these imports of chart.js ChartOptions, ChartType, ChartDataSets, ng2-charts Label, ApiService, and Statistic data type.

```
import { ChartOptions, ChartType, ChartDataSets } from 'chart.js';  
import { Label } from 'ng2-charts';  
import { ApiService } from '../api.service';  
import { Statistic } from '../statistic';
```

Declare these required variables before the constructor for building a bar chart.

```

stats: Statistic[] = [];
label = 'Positive';
isLoadingResults = true;
barChartOptions: ChartOptions = {
  responsive: true,
};
barChartLabels: Label[] = [];
barChartType: ChartType = 'bar';
barChartLegend = true;
barChartPlugins = [];
barChartData: ChartDataSets[] = [{ data: [], backgroundColor: [], label: this.label }];

```

Inject ApiService to the constructor.

```

constructor(private api: ApiService) { }

```

Add a function to load statistic data from REST API then implement it as a bar chart.

```

getStatistic(status: string) {
  this.barChartData = [{ data: [], backgroundColor: [], label: this.label }];
  this.barChartLabels = [];
  this.api.getStatistic(status)
    .subscribe((res: any) => {
      this.stats = res;
      const chartdata: number[] = [];
      const chartcolor: string[] = [];
      this.stats.forEach((stat) => {
        this.barChartLabels.push(stat._id.date);
        chartdata.push(stat.count);
        if (this.label === 'Positive') {
          chartcolor.push('rgba(255, 165, 0, 0.5)');
        } else if (this.label === 'Dead') {
          chartcolor.push('rgba(255, 0, 0, 0.5)');
        } else {
          chartcolor.push('rgba(0, 255, 0, 0.5)');
        }
      });
    });
}

```

```

        this.barChartData = [{ data: chartdata, backgroundColor: chartcolor, label: this.label }];
        this.isLoadingResults = false;
    }, err => {
        console.log(err);
        this.isLoadingResults = false;
    });
}

```

Call that function to the NgOnInit function.

```

ngOnInit(): void {
    this.getStatistic(this.label);
}

```

Add a function to switch or reload statistic data by status value.

```

changeStatus() {
    this.isLoadingResults = true;
    this.getStatistic(this.label);
}

```

Next, open and edit `src/app/cases-stat/cases-stat.component.html` then replace all HTML tags with this implementation of an ng2-charts/Chart.js bar chart with statistic data.

```

<div class="example-container mat-elevation-z8">
  <h2>Corona Virus Cases Statistic</h2>
  <div class="example-loading-shade"
    *ngIf="isLoadingResults">
    <mat-spinner *ngIf="isLoadingResults"></mat-spinner>
  </div>
  <div class="button-row">
    <a mat-flat-button color="primary" [routerLink]="['/cases']"><mat-icon>list</mat-icon></a>
  </div>
  <div class="button-row">
    <mat-button-toggle-group name="status" aria-label="Status" [(ngModel)]
      ="label" (ngModelChange)="changeStatus()">
      <mat-button-toggle value="Positive">Positive</mat-button-toggle>

```

```

    <mat-button-toggle value="Dead">Dead</mat-button-toggle>
    <mat-button-toggle value="Recovered">Recovered</mat-button-toggle>
  </mat-button-toggle-group>
</div>
<div style="display: block;">
  <canvas baseChart
    [datasets]="barChartData"
    [labels]="barChartLabels"
    [options]="barChartOptions"
    [plugins]="barChartPlugins"
    [legend]="barChartLegend"
    [chartType]="barChartType">
  </canvas>
</div>
</div>

```

Finally, give it a little style by modify `src/app/cases-stat/cases-stat.component.scss` with these.

```

/* Structure */
.example-container {
  position: relative;
  padding: 5px;
}

.example-loading-shade {
  position: absolute;
  top: 0;
  left: 0;
  bottom: 56px;
  right: 0;
  background: rgba(0, 0, 0, 0.15);
  z-index: 1;
  display: flex;
  align-items: center;
  justify-content: center;
}

.mat-flat-button {
  margin: 5px;
}

```

Step #7. Add a Data using Angular Material Form

To create a form for adding a Coronavirus case, open and edit `src/app/add-cases/add-cases.component.ts` then add these imports.

```
import { Router } from '@angular/router';
import { ApiService } from '../api.service';
import { FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators } from '@angular/forms';
import { ErrorStateMatcher } from '@angular/material/core';
```

Inject the above modules to the constructor.

```
constructor(private router: Router, private api: ApiService, private formBuilder: FormBuilder) { }
```

Declare variables for the Form Group and all of the required fields inside the form before the constructor.

```
casesForm: FormGroup;
name = '';
gender = '';
age: number = null;
address = '';
city = '';
country = '';
status = '';
statusList = ['Positive', 'Dead', 'Recovered'];
genderList = ['Male', 'Female'];
isLoadingResults = false;
matcher = new MyErrorStateMatcher();
```

Add initial validation for each field.

```

ngOnInit(): void {
  this.casesForm = this.formBuilder.group({
    name : [null, Validators.required],
    gender : [null, Validators.required],
    age : [null, Validators.required],
    address : [null, Validators.required],
    city : [null, Validators.required],
    country : [null, Validators.required],
    status : [null, Validators.required]
  });
}

```

Create a function for submitting or POST cases form.

```

onFormSubmit() {
  this.isLoadingResults = true;
  this.api.addCases(this.casesForm.value)
    .subscribe((res: any) => {
      const id = res._id;
      this.isLoadingResults = false;
      this.router.navigate(['/cases-details', id]);
    }, (err: any) => {
      console.log(err);
      this.isLoadingResults = false;
    });
}

```

Create a new class before the main class `@Components`.

```

/** Error when invalid control is dirty, touched, or submitted. */
export class MyErrorStateMatcher implements ErrorStateMatcher {
  isErrorState(control: FormControl | null, form: FormGroupDirective | NgForm | null): boolean {
    const isSubmitted = form && form.submitted;
    return !!(control && control.invalid && (control.dirty || control.touched || isSubmitted));
  }
}

```

Next, open and edit `src/app/add-cases/add-cases.component.html` then replace all HTML tags with this.

```
<div class="example-container mat-elevation-z8">
  <h2>Coronavirus Add Cases</h2>
  <div class="example-loading-shade"
    *ngIf="isLoadingResults">
    <mat-spinner *ngIf="isLoadingResults"></mat-spinner>
  </div>
  <div class="button-row">
    <a mat-flat-button color="primary" [routerLink]="['/cases']"><mat-icon>l
ist</mat-icon></a>
  </div>
  <mat-card class="example-card">
    <form [formGroup]="casesForm" (ngSubmit)="onFormSubmit()">
      <mat-form-field class="example-full-width">
        <mat-label>Name</mat-label>
        <input matInput placeholder="Name" formControlName="name"
          [errorStateMatcher]="matcher">
        <mat-error>
          <span *ngIf="!casesForm.get('name').valid && casesForm.get('nam
e').touched">Please enter Name</span>
        </mat-error>
      </mat-form-field>
      <mat-form-field class="example-full-width">
        <mat-label>Gender</mat-label>
        <mat-select formControlName="gender">
          <mat-option *ngFor="let gl of genderList" [value]="gl">
            {{gl}}
          </mat-option>
        </mat-select>
        <mat-error>
          <span *ngIf="!casesForm.get('gender').valid && casesForm.get('gend
er').touched">Please choose Gender</span>
        </mat-error>
      </mat-form-field>
      <mat-form-field class="example-full-width">
        <mat-label>Age</mat-label>
        <input matInput type="number" placeholder="Age" formControlName="ag
e"
          [errorStateMatcher]="matcher">
```



```

    <mat-error>
      <span *ngIf="!casesForm.get('age').valid && casesForm.get('age').t
ouched">Please enter Age</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>Address</mat-label>
    <input matInput placeholder="Address" formControlName="address"
      [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!casesForm.get('address').valid && casesForm.get('add
ress').touched">Please enter Address</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>City</mat-label>
    <input matInput placeholder="City" formControlName="city"
      [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!casesForm.get('city').valid && casesForm.get('cit
y').touched">Please enter City</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>Country</mat-label>
    <input matInput placeholder="Country" formControlName="country"
      [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!casesForm.get('country').valid && casesForm.get('cou
ntry').touched">Please enter Country</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>Status</mat-label>
    <mat-select formControlName="status">
      <mat-option *ngFor="let sl of statusList" [value]="sl">
        {{sl}}
      </mat-option>
    </mat-select>
    <mat-error>
      <span *ngIf="!casesForm.get('status').valid && casesForm.get('stat

```

```

us').touched">Please select Status</span>
    </mat-error>
    </mat-form-field>
    <div class="button-row">
        <button type="submit" [disabled]="!casesForm.valid" mat-flat-button
color="primary"><mat-icon>save</mat-icon></button>
    </div>
</form>
</mat-card>
</div>

```

Finally, open and edit `src/app/add-cases/add-cases.component.scss` then add this CSS codes.

```

/* Structure */
.example-container {
  position: relative;
  padding: 5px;
}

.example-form {
  min-width: 150px;
  max-width: 500px;
  width: 100%;
}

.example-full-width {
  width: 100%;
}

.example-full-width:nth-last-child(0) {
  margin-bottom: 10px;
}

.button-row {
  margin: 10px 0;
}

.mat-flat-button {
  margin: 5px;
}

```

Step #8. Edit a Data using Angular Material Form

We already put an edit button inside the Cases Details component for the call Edit page. Now, open and edit `src/app/edit-cases/edit-cases.component.ts` then add these imports.

```
import { Router, ActivatedRoute } from '@angular/router';
import { ApiService } from '../api.service';
import { FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators } from '@angular/forms';
import { ErrorStateMatcher } from '@angular/material/core';
```

Inject the above modules to the constructor.

```
constructor(private router: Router, private route: ActivatedRoute, private api: ApiService, private formBuilder: FormBuilder) { }
```

Declare the Form Group variable and all of the required variables for the cases-form before the constructor.

```
casesForm: FormGroup;
_id = '';
name = '';
gender = '';
age: number = null;
address = '';
city = '';
country = '';
status = '';
statusList = ['Positive', 'Dead', 'Recovered'];
genderList = ['Male', 'Female'];
isLoadingResults = false;
matcher = new MyErrorStateMatcher();
```

Next, add validation for all fields when the component is initiated.

```
ngOnInit(): void {  
  this.getCasesById(this.route.snapshot.params.id);  
  this.casesForm = this.formBuilder.group({  
    name : [null, Validators.required],  
    gender : [null, Validators.required],  
    age : [null, Validators.required],  
    address : [null, Validators.required],  
    city : [null, Validators.required],  
    country : [null, Validators.required],  
    status : [null, Validators.required]  
  });  
}
```

Create a function for getting cases data that filled to each form field.

```
getCasesById(id: any) {  
  this.api.getCasesById(id).subscribe((data: any) => {  
    this._id = data._id;  
    this.casesForm.setValue({  
      name: data.name,  
      gender: data.gender,  
      age: data.age,  
      address: data.address,  
      city: data.city,  
      country: data.country,  
      status: data.status  
    });  
  });  
}
```

Create a function to update the case changes.

```
onFormSubmit() {  
  this.isLoadingResults = true;  
  this.api.updateCases(this._id, this.casesForm.value)  
    .subscribe((res: any) => {  
    const id = res._id;  
    this.isLoadingResults = false;  
  });  
}
```

```

        this.router.navigate(['/cases-details', id]);
      }, (err: any) => {
        console.log(err);
        this.isLoadingResults = false;
      })
    );
  }
}

```

Add a function for handling the show cases details button.

```

casesDetails() {
  this.router.navigate(['/cases-details', this._id]);
}

```

Create a new class before the main class `@Components`.

```

/** Error when invalid control is dirty, touched, or submitted. */
export class MyErrorStateMatcher implements ErrorStateMatcher {
  isErrorState(control: FormControl | null, form: FormGroupDirective | NgForm | null): boolean {
    const isSubmitted = form && form.submitted;
    return !!(control && control.invalid && (control.dirty || control.touched || isSubmitted));
  }
}

```

Next, open and edit `src/app/edit-cases/edit-cases.component.html` then replace all HTML tags with this.

```

<div class="example-container mat-elevation-z8">
  <h2>Coronavirus Edit Cases</h2>
  <div class="example-loading-shade"
    *ngIf="isLoadingResults">
    <mat-spinner *ngIf="isLoadingResults"></mat-spinner>
  </div>
  <div class="button-row">
    <a mat-flat-button color="primary" (click)="casesDetails()"><mat-icon>
info</mat-icon></a>
  </div>
  <mat-card class="example-card">

```

```

<form [formGroup]="casesForm" (ngSubmit)="onFormSubmit()">
  <mat-form-field class="example-full-width">
    <mat-label>Name</mat-label>
    <input matInput placeholder="Name" formControlName="name"
      [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!casesForm.get('name').valid && casesForm.get('name').t
e').touched">Please enter Name</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>Gender</mat-label>
    <mat-select formControlName="gender">
      <mat-option *ngFor="let gl of genderList" [value]="gl">
        {{gl}}
      </mat-option>
    </mat-select>
    <mat-error>
      <span *ngIf="!casesForm.get('gender').valid && casesForm.get('gend
er').touched">Please choose Gender</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>Age</mat-label>
    <input matInput type="number" placeholder="Age" formControlName="ag
e"
      [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!casesForm.get('age').valid && casesForm.get('age').t
ouched">Please enter Age</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <mat-label>Address</mat-label>
    <input matInput placeholder="Address" formControlName="address"
      [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!casesForm.get('address').valid && casesForm.get('add
ress').touched">Please enter Address</span>
    </mat-error>
  </mat-form-field>

```

```

<mat-form-field class="example-full-width">
  <mat-label>City</mat-label>
  <input matInput placeholder="City" formControlName="city"
    [errorStateMatcher]="matcher">
  <mat-error>
    <span *ngIf="!casesForm.get('city').valid && casesForm.get('city').touched">Please enter City</span>
  </mat-error>
</mat-form-field>
<mat-form-field class="example-full-width">
  <mat-label>Country</mat-label>
  <input matInput placeholder="Country" formControlName="country"
    [errorStateMatcher]="matcher">
  <mat-error>
    <span *ngIf="!casesForm.get('country').valid && casesForm.get('country').touched">Please enter Country</span>
  </mat-error>
</mat-form-field>
<mat-form-field class="example-full-width">
  <mat-label>Status</mat-label>
  <mat-select formControlName="status">
    <mat-option *ngFor="let sl of statusList" [value]="sl">
      {{sl}}
    </mat-option>
  </mat-select>
  <mat-error>
    <span *ngIf="!casesForm.get('status').valid && casesForm.get('status').touched">Please select Status</span>
  </mat-error>
</mat-form-field>
<div class="button-row">
  <button type="submit" [disabled]="!casesForm.valid" mat-flat-button color="primary"><mat-icon>save</mat-icon></button>
</div>
</form>
</mat-card>
</div>

```

Finally, open and edit `src/app/edit-cases/edit-cases.component.scss` then add these lines of CSS codes.

```
/* Structure */
.example-container {
  position: relative;
  padding: 5px;
}

.example-form {
  min-width: 150px;
  max-width: 500px;
  width: 100%;
}

.example-full-width {
  width: 100%;
}

.example-full-width:nth-last-child(0) {
  margin-bottom: 10px;
}

.button-row {
  margin: 10px 0;
}

.mat-flat-button {
  margin: 5px;
}
```

Step #9. Run and Test the Angular 9 Coronavirus Cases App

Let's see the performance of the Angular 9 with the Ivy CRUD App. Now, we have to build the Angular 9 app using this command.

```
ng build --prod
```


Now, we have ES5 and ES2015 build of the Angular 9 app build for production. Next, we have to test the whole application, first, we have to run MongoDB server and Node/Express API in the different terminal.

```
mongod  
nodemon
```

Then run the Angular 9 app build, simply type this command.

```
ng serve
```

Now, you will see the Coronavirus Cases app the same as you saw in the first paragraph of this tutorial. That it's the Angular 9 Tutorial: Learn to Build a CRUD Angular App Quickly. You can find the full source code in our [GitHub](#).

If you don't want to waste your time design your own front-end or your budget to spend by hiring a web designer then Angular Templates is the best place to go. So, speed up your front-end web development with premium [Angular templates](#). Choose your template for your front-end project [here](#).

That just the basic. If you need more deep learning about MEAN Stack, Angular, and Node.js, you can take the following cheap course:

- [Master en JavaScript: Aprender JS, jQuery, Angular 8, NodeJS](#)
- [Angular 8 - Complete Essential Guide](#)
- [Learn Angular 8 by creating a simple Full Stack Web App](#)
- [Angular 5 Bootcamp FastTrack](#)
- [Angular 6 - Soft & Sweet](#)
- [Angular 6 with TypeScript](#)

Thanks!



[report this ad](#)

[← Previous Article](#)

[Angular Material Form Controls
Select \(mat-select\) Example](#)

Related Articles

- [Angular Material Form Controls Select \(mat-select\) Example](#)
- [Angular 8 Tutorial: How to Create an Angular Web App Quickly](#)
- [Angular Material Form Controls, Form Field and Input Examples](#)
- [Angular 8 Tutorial: Observable and RXJS Examples](#)
- [Angular 8 Tutorial: REST API and HttpClient Examples](#)
- [Angular 8 Google Maps Firebase Realtime Blood Donor App](#)
- [Angular 8 Tutorial: Routing & Navigation Example](#)
- [Angular 8 Tutorial: Facebook Login](#)
- [Angular 8 RxJS Multiple HTTP Request using the forkJoin Example](#)
- [Angular 8 Universal and MongoDB Server-side Rendering \(SSR\)](#)
- [Angular 8 Tutorial: Learn to Build Angular 8 CRUD Web App](#)
- [Angular 7 Tutorial: Create Angular Material CDK Virtual Scroll](#)
- [Angular 7 Tutorial: Building CRUD Web Application](#)
- [Angular 6 Firebase Tutorial: Firestore CRUD Web Application](#)
- [Angular 6 HttpClient: Consume RESTful API Example](#)
- [Angular 6 Tutorial: Getting Started Build Angular 6 Web Application](#)
- [Getting Started Angular 4 using Angular CLI](#)

