Building CRUD Mobile App using Ionic 4, Angular 6 and Cordova

by Didin J., updated on Aug 19, 2019

A comprehensive step by step tutorial on build Ionic 4, Angular 6 and Cordova CRUD (Create, Read, Update, Delete) Mobile App

A comprehensive step by step tutorial on building Ionic 4, Angular 6 and Cordova CRUD (Create, Read, Update, Delete) Android and iOS Mobile App. We will access REST API or web service that previously we create on Node.js, Express.js and PostgreSQL (https://www.djamware.com/post/5b56a6cc80aca707dd4f65a9/nodejs-expressjs-sequelizejs-and-postgresql-restful-api) tutorial. For this tutorial, we are not using all REST API for CRUD, we just use a Classroom API only. We write this tutorial because we are curious about the new feature of Angular 6 when using it in Ionic 4 app.

Shortcut to the steps:

- Create a New Ionic 4 and Angular 6 Application
- Create Service for Accessing REST API
- View List of Data in Home Page
- Create A Page for Display Classroom Details
- Create A Page for Edit and Update Classroom Data
- Create A Page for Add Classroom Data
- Test and Run The App in Browser, Android and iOS Device

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

- Node.js (Recommended version)
- Ionic 4 beta
- Angular 6
- Express.js and PostgreSQL RESTful API (https://github.com/didinj/node-express-postgresql-sequelize.git)
- Terminal or Node command line
- IDE or Text Editor

Before going to the main steps, we assume that you have to install Node.js. Next, upgrade or install new Ionic 4 CLI by open the terminal or Node command line then type this command.

```
sudo npm install -g ionic
```

You will get the latest Ionic CLI pro 4.0.0 in your terminal or command line.

Create a New Ionic 4 and Angular 6 Application

To create a new Ionic 4 and Angular 6 application, type this command in your terminal.

```
ionic start ionic4-angular6-crud sidemenu --type=angular
```

If you see this question, just type `N` for because we will installing or adding Cordova later.

```
Integrate your new app with Cordova to target native iOS and Android? (y/N) N
```

After installing `NPM` modules and dependencies, you will see this question, just type `N` because we are not using it yet.

```
Install the free Ionic Pro SDK and connect your app? (Y/n) N
```

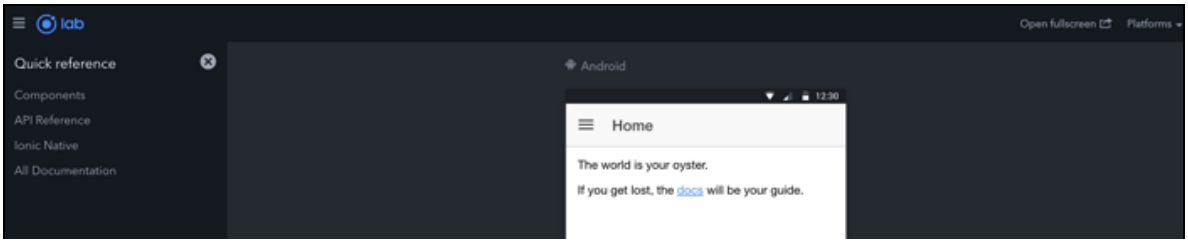
Next, go to the newly created app folder.

```
cd ./ionic4-angular6-crud
```

As usual, run the Ionic 4 and Angular 6 app for the first time, but before run as `lab` mode, type this command to install `@ionic/lab`.

```
npm install --save-dev @ionic/lab
ionic serve -l
```

Now, open the browser and you will the Ionic 4 and Angular 6 app with the iOS, Android, or Windows view.



Create Service for Accessing REST API

To separate the presentation data from data access, we have to create a service for accessing REST API from our server.

```
ionic g service rest-api
```

Before modifying the service file, open and edit `src/app/app.module.ts` then add this import of HttpClientModule.

```
import { HttpClientModule } from '@angular/common/http';
```

Then add this module to the `@NgModule` imports.

```
imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, HttpClientModule,],
```

Next, open and edit `src/app/rest-api.service.ts` then add these imports of RxJS Observable, of, throwError, HttpClient, HttpHeaders, HttpErrorResponse, RxJS Operators catch error, tap, and map.

```
import { Observable, of, throwError } from 'rxjs';
import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
import { catchError, tap, map } from 'rxjs/operators';
```

Add constant variable before the `@Injectable` annotation.

```
const httpOptions = {
  headers: new HttpHeaders({'Content-Type': 'application/json'})
};
const apiUrl = "http://localhost:1337/localhost:3000/api/classroom";
```

That constant variable shows the long URL for API because we need to use `corsproxy` for accessing non-CORS enabled server. For that, install the `corsproxy` first by type this command.

```
sudo npm install -g corsproxy
```

Next, inject `HttpClient` to the constructor params.

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

Add the functions for handle error and extract response data.

```
private handleError(error: HttpErrorResponse) {
  if (error.error instanceof ErrorEvent) {
    // A client-side or network error occurred. Handle it accordingly.
    console.error('An error occurred:', error.error.message);
  } else {
    // The backend returned an unsuccessful response code.
    // The response body may contain clues as to what went wrong,
    console.error(
      `Backend returned code ${error.status}, ` +
      `body was: ${error.error}`);
  }
  // return an observable with a user-facing error message
  return throwError('Something bad happened; please try again later.');
}

private extractData(res: Response) {
  let body = res;
  return body || { };
}
```

Next, add all CRUD (Create, Read, Update, Delete) function for accessing the CRUD REST API from the server.

```
getClassroom(): Observable<any> {
  return this.http.get(apiUrl, httpOptions).pipe(
    map(this.extractData),
    catchError(this.handleError));
}

getClassroomById(id: string): Observable<any> {
  const url = `${apiUrl}/${id}`;
  return this.http.get(url, httpOptions).pipe(
    map(this.extractData),
    catchError(this.handleError));
}

postClassroom(data): Observable<any> {
  const url = `${apiUrl}/add_with_students`;
  return this.http.post(url, data, httpOptions)
    .pipe(
      catchError(this.handleError)
    );
}

updateClassroom(id: string, data): Observable<any> {
  const url = `${apiUrl}/${id}`;
  return this.http.put(url, data, httpOptions)
    .pipe(
      catchError(this.handleError)
    );
}

deleteClassroom(id: string): Observable<{}> {
  const url = `${apiUrl}/${id}`;
  return this.http.delete(url, httpOptions)
    .pipe(
      catchError(this.handleError)
    );
}
```

View List of Data in Home Page

We will put the list of classroom data in the existing home page. To view the list of the classroom from REST API that previously handles by the service, open and edits `src/app /list/list.page.ts` then add these imports of Ionic LoadingController and RestAPIService.

```
import { LoadingController } from '@ionic/angular';
import { RestApiService } from '../rest-api.service';
```

Inject the `RestApiService` to the constructor params.

```
constructor(public api: RestApiService, public loadingController: LoadingController) { }
```

Remove all default generated variable, function and constructor body then add this variable before the constructor for hold classroom data that get from the service.

```
classrooms: any;
```

Add this function for getting the list of the classroom from the Rest API service.

```
async getClassrooms() {
  const loading = await this.loadingController.create({
    content: 'Loading'
  });
  await loading.present();
  await this.api.getClassroom()
    .subscribe(res => {
      console.log(res);
      this.classrooms = res;
      loading.dismiss();
    }, err => {
      console.log(err);
      loading.dismiss();
    });
}
```

In the Angular 6 `ngOnInit` lifecycle hooks call that function.

```
ngOnInit() {
  this.getClassrooms();
}
```

Next, open and edit `src/app/list/list.page.html` then replace all HTML 5 tags with a list of classrooms.

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
    <ion-title>
      Clasroom List
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    <ion-item *ngFor="let cr of classrooms" routerLink="/detail/{{cr.id}}">
      <ion-icon name="desktop" slot="start"></ion-icon>
      {{cr.class_name}}
      <div class="item-note" slot="end">
        {{cr.createdAt | date}}
      </div>
    </ion-item>
  </ion-list>
  <!--
    <div *ngIf="selectedItem" padding>
      You navigated here from <b>{{selectedItem.title }}</b>
    </div>
  -->
</ion-content>
```

As you see, there's a `routerLink` to the details page that we will create in the next step.

Create A Page for Display Classroom Details

To create a detail page for view details of the classroom that pick from classroom list, type this command.

```
ionic g page detail
```

We have to modify the detailed route by open and edit `src/app/app-routing.module.ts` then make the detailed route like this.

```
{
  path: 'detail/:id',
  loadChildren: './detail/detail.module#DetailPageModule'
}
```

Next, open and edit `src/app/detail/detail.page.ts` then add these imports of Ionic LoadingController, RestApiService, Angular ActivatedRoute, and Router.

```
import { Component, OnInit } from '@angular/core';
import { LoadingController } from '@ionic/angular';
import { RestApiService } from '../rest-api.service';
import { ActivatedRoute, Router } from '@angular/router';
```

Inject that imported module to the constructor params.

```
constructor(public api: RestApiService,
  public loadingController: LoadingController,
  public route: ActivatedRoute,
  public router: Router) { }
```

Declare a variable before the constructor to hold the response from API.

```
classroom: any = {};
```

Create a function for getting classroom detail data from REST API service.

```
async getClassroom() {
  const loading = await this.loadingController.create({
    content: 'Loading'
  });
  await loading.present();
  await this.api.getClassroomById(this.route.snapshot.paramMap.get('id'))
    .subscribe(res => {
      console.log(res);
      this.classroom = res;
      loading.dismiss();
    }, err => {
      console.log(err);
      loading.dismiss();
    });
}
```

Modify `ngOnInit` function to call above function.

```
ngOnInit() {
  this.getClassroom();
}
```

Add function for delete data.

```
async delete(id) {
  const loading = await this.loadingController.create({
    content: 'Deleting'
  });
  await loading.present();
  await this.api.deleteClassroom(id)
    .subscribe(res => {
      loading.dismiss();
      this.location.back();
    }, err => {
      console.log(err);
      loading.dismiss();
    });
}
```

Next, open and edit `src/app/detail/detail.page.html` then replace all HTML 5 tags with Ionic Card Components.

```
<ion-header>
  <ion-toolbar>
    <ion-title>Classroom Details</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <ion-card>
    <ion-card-header>
      <ion-card-title>{{classroom.class_name}}</ion-card-title>
    </ion-card-header>

    <ion-card-content>
      <ion-card-subtitle>Students:</ion-card-subtitle>
      <ul>
        <li *ngFor="let student of classroom.students">{{student.student_name}}</li>
      </ul>
      <ion-buttons slot="primary">
        <ion-button routerLink="/edit/{{classroom.id}}">
          <ion-icon slot="icon-only" name="create"></ion-icon>
        </ion-button>
        <ion-button (click)="delete(classroom.id)">
          <ion-icon slot="icon-only" name="remove-circle"></ion-icon>
        </ion-button>
      </ion-buttons>
    </ion-card-content>
  </ion-card>
</ion-content>
```

Create A Page for Edit and Update Classroom

Data

Same as the previous step, type this command to generate an Ionic 4 page for edit and update classroom data.

```
ionic g page edit
```

Open and edit again `src/app/app-routing.module.ts` then add params to edit the route.

```
{
  path: 'edit/:id',
  loadChildren: './edit/edit.module#EditPageModule'
}
```

Next, open and edit `src/app/edit/edit.module.ts` then add or modify this import.

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

Add `ReactiveFormsModule` to `@NgModule` imports.

```
imports: [
  CommonModule,
  FormsModule,
  IonicModule,
  ReactiveFormsModule,
  RouterModule.forChild(routes)
],
```

Next, open and edit `src/app/edit/edit.page.ts` then add these imports of Ionic LoadingController, Angular ActivatedRoute, Router, FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators, and FormArray.

```
import { LoadingController } from '@ionic/angular';
import { RestApiService } from '../rest-api.service';
import { ActivatedRoute, Router  } from '@angular/router';
import { FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators, FormArr
ay } from '@angular/forms';
```

Inject above modules into constructor params.

```
constructor(public api: RestApiService,
  public loadingController: LoadingController,
  private route: ActivatedRoute,
  public router: Router,
  private formBuilder: FormBuilder) {}
```

Add these variables before the constructor.

```
classroomForm: FormGroup;
students: FormArray;
```

Create a function to get Classroom by ID from REST API service.

```
async getClassroom(id) {
  const loading = await this.loadingController.create({
    content: 'Loading'
  });
  await loading.present();
  await this.api.getClassroomById(id).subscribe(res => {
    this.classroomForm.controls['class_name'].setValue(res.class_name);
    let controlArray = <FormArray>this.classroomForm.controls['students'];
    res.students.forEach(std => {
      controlArray.push(this.formBuilder.group({
         student_name: ''
      }));
    });
    for(let i=0;i<res.students.length;i++) {
      controlArray.controls[i].get('student_name').setValue(res.students[i].student_name);
    }
    console.log(this.classroomForm);
    loading.dismiss();
  }, err => {
    console.log(err);
    loading.dismiss();
  });
}
```

Initialize the form group by adding it to the body of the constructor.

```
constructor(public api: RestApiService,
  public loadingController: LoadingController,
  private route: ActivatedRoute,
  public router: Router,
  private formBuilder: FormBuilder) {
  this.getClassroom(this.route.snapshot.paramMap.get('id'));
  this.classroomForm = this.formBuilder.group({
    'class_name' : [null, Validators.required],
    'students' : this.formBuilder.array([])
  });
}
```

Add the functions for adding form array and delete an item from the form array.

```
createStudent(): FormGroup {
  return this.formBuilder.group({
    student_name: ''
  });
}

addBlankStudent(): void {
  this.students = this.classroomForm.get('students') as FormArray;
  this.students.push(this.createStudent());
}

deleteStudent(control, index) {
  control.removeAt(index)
}
```

Add function for saving classroom data by post the form data to REST API service.

```
async updateClassroom(){
  await this.api.updateClassroom(this.route.snapshot.paramMap.get('id'), this.classroomForm.v
alue)
  .subscribe(res => {
      let id = res['id'];
      this.router.navigate(['/detail', JSON.stringify(id)]);
    }, (err) => {
      console.log(err);
    });
}
```

Next, open and edit `src/app/edit/edit.page.html` then replace all HTML 5 tags with Angular form that contains FormArray, FormGroup and FormControl.

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>
    <ion-title>Edit Classroom</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <form [formGroup]="classroomForm">
    <ion-item>
      <ion-label position="floating">Classroom Name</ion-label>
      <ion-input type="text" formControlName="class_name"></ion-input>
    </ion-item>
    <ion-item-divider>
      <ion-label>
        Students <ion-button shape="round" (click)="addBlankStudent()"><ion-icon name="add-ci
rcle"></ion-icon></ion-button>
      </ion-label>
    </ion-item-divider>
    <div formArrayName="students">
      <ion-item *ngFor="let student of classroomForm.get('students').controls; let i=index"
[formGroupName]="i">
        <ion-label position="floating">Student Name</ion-label>
        <ion-row>
          <ion-col size="10">
            <ion-input type="text" formControlName="student_name" item-start></ion-input>
          </ion-col>
          <ion-col size="2">
            <ion-button shape="round" color="dark" (click)="deleteStudent(classroomForm.get('
students').controls,i)"><ion-icon name="trash"></ion-icon></ion-button>
          </ion-col>
        </ion-row>
      </ion-item>
    </div>
    <ion-button shape="round" color="primary" expand="block" [disabled]="!classroomForm.vali
d" (click)="updateClassroom()">Submit</ion-button>
  </form>
</ion-content>
```

Create A Page for Add Classroom Data

Same as the previous step, type this command to generate an Ionic 4 page for add and save classroom data.

```
ionic g page create
```

Next, open and edit `src/app/create.module.ts` then add or modify this import to add `ReactiveFormsModule`.

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

Add this `ReactiveFormsModule` to `@NgModule` imports.

```
imports: [
  CommonModule,
  FormsModule,
  IonicModule,
  ReactiveFormsModule,
  RouterModule.forChild(routes)
],
```

Next, open and edit `src/app/create/create.page.ts` then add these imports of LoadingController, ActivatedRoute, Router, FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators, and FormArray.

```
import { LoadingController } from '@ionic/angular';
import { RestApiService } from '../rest-api.service';
import { ActivatedRoute, Router  } from '@angular/router';
import { FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators, FormArr
ay } from '@angular/forms';
```

Inject that modules to the constructor params.

```
constructor(public api: RestApiService,
  public loadingController: LoadingController,
  private route: ActivatedRoute,
  public router: Router,
  private formBuilder: FormBuilder) { }
```

Add a variable of `FormGroup` and `FormArray` before the constructor.

```
classroomForm: FormGroup;
students: FormArray;
```

Initialize the classroom form inside the constructor body.

```
constructor(public api: RestApiService,
  public loadingController: LoadingController,
  private route: ActivatedRoute,
  public router: Router,
  private formBuilder: FormBuilder) {
    this.getClassroom(this.route.snapshot.paramMap.get('id'));
    this.classroomForm = this.formBuilder.group({
      'class_name' : [null, Validators.required],
      'students' : this.formBuilder.array([])
    });
  }
```

Add the functions for adding an item to the form array.

```
createStudent(): FormGroup {
  return this.formBuilder.group({
    student_name: ''
  });
}

addBlankStudent(): void {
  this.students = this.classroomForm.get('students') as FormArray;
  this.students.push(this.createStudent());
}
```

Also for remove item from the form array.

```
deleteStudent(control, index) {
  control.removeAt(index)
}
```

Next, add the function to post the form data to the REST API service.

```
async saveClassroom(){
  await this.api.postClassroom(this.classroomForm.value)
  .subscribe(res => {
      let id = res['id'];
      this.router.navigate(['/detail/'+id]);
    }, (err) => {
      console.log(err);
    });
}
```

Next, open and edit `src/app/create/create.page.html` then replace all HTML 5 tags with Angular FormGroup, FormArray, and FormControl.

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>
    <ion-title>Add Classroom</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <form [formGroup]="classroomForm">
    <ion-item>
      <ion-label position="floating">Classroom Name</ion-label>
      <ion-input type="text" formControlName="class_name"></ion-input>
    </ion-item>
    <ion-item-divider>
      <ion-label>
        Students <ion-button shape="round" (click)="addBlankStudent()"><ion-icon name="add-ci
rcle"></ion-icon></ion-button>
      </ion-label>
    </ion-item-divider>
    <div formArrayName="students">
      <ion-item *ngFor="let student of classroomForm.get('students').controls; let i=index"
[formGroupName]="i">
        <ion-label position="floating">Student Name</ion-label>
        <ion-row>
          <ion-col size="10">
            <ion-input type="text" formControlName="student_name" item-start></ion-input>
          </ion-col>
          <ion-col size="2">
            <ion-button shape="round" color="dark" (click)="deleteStudent(classroomForm.get('
students').controls,i)"><ion-icon name="trash"></ion-icon></ion-button>
          </ion-col>
        </ion-row>
      </ion-item>
    </div>
    <ion-button shape="round" color="primary" expand="block" [disabled]="!classroomForm.vali
d" (click)="saveClassroom()">Submit</ion-button>
  </form>
</ion-content>
```

Finally, open and edit `src/app/list/list.page.html` then add this button to the toolbar for open above page.

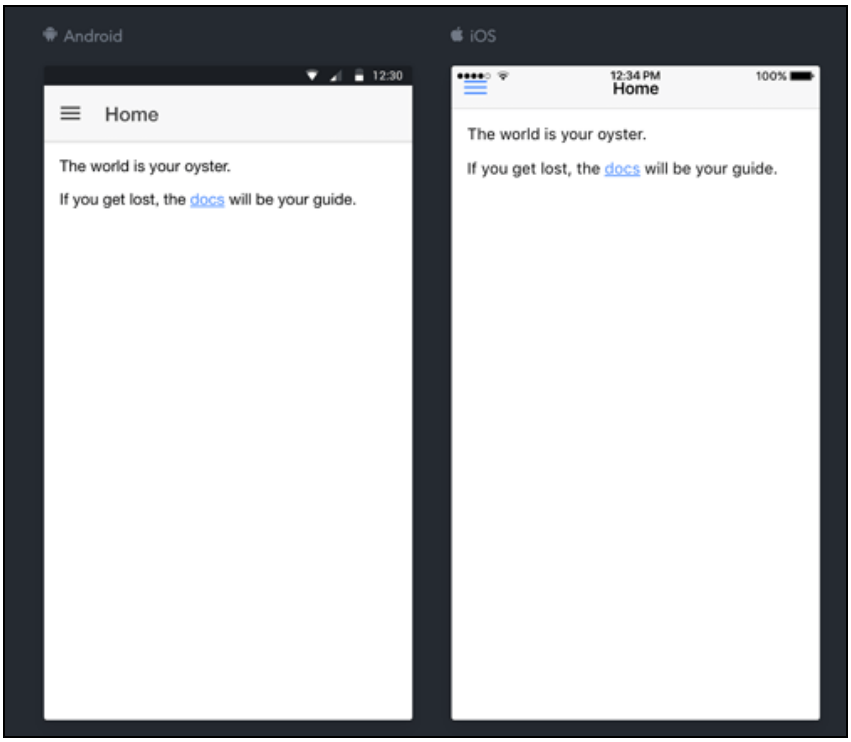
```
<ion-toolbar>
  <ion-buttons slot="start">
    <ion-menu-button></ion-menu-button>
  </ion-buttons>
  <ion-title>
    Clasroom List
  </ion-title>
  <ion-buttons slot="end">
    <ion-button routerLink="/create"><ion-icon name="add-circle"></ion-icon></ion-button>
  </ion-buttons>
</ion-toolbar>
```

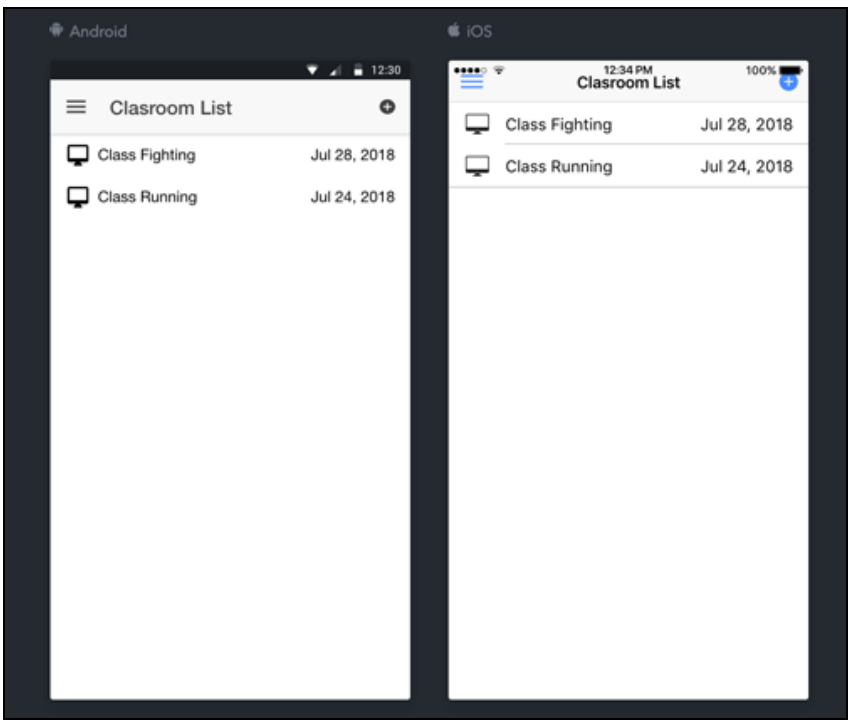
Test and Run The App in Browser, Android and iOS Device

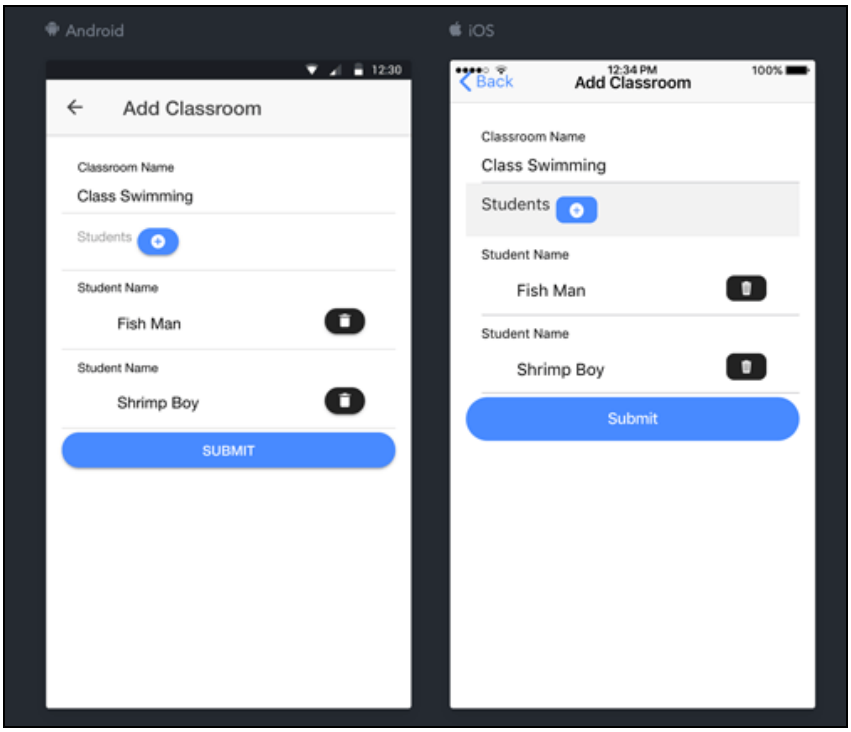
First, we have to run the Ionic 4 and Angular 6 app to the browser. Type again this command to test, make sure you have run the recommended RESTful API server and `corsproxy`.

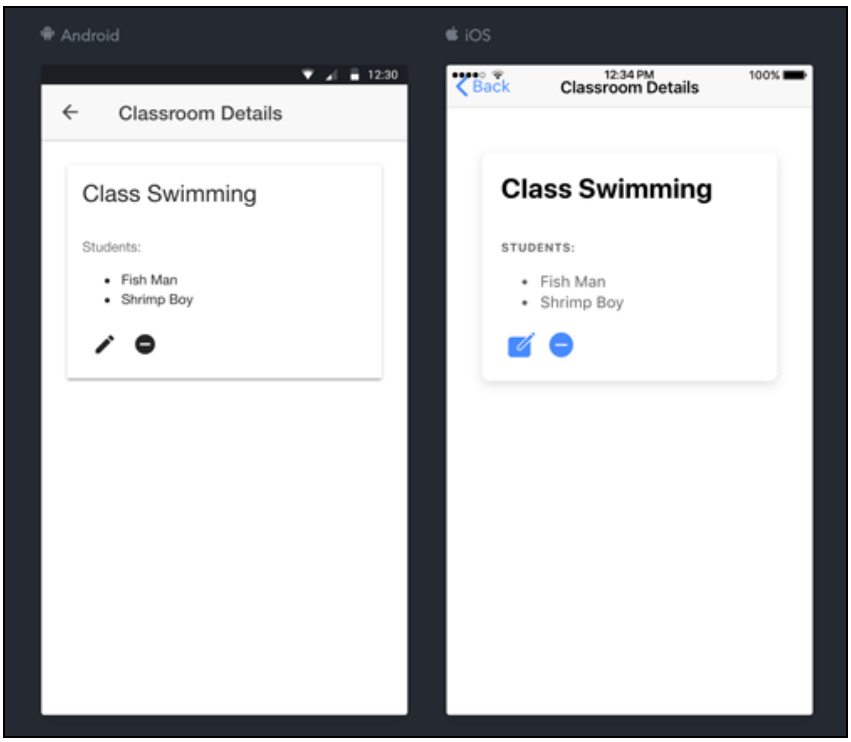
```
ionic serve -l
```

It will automatically open the browser and show the main page of the app. Just navigate the app menu to find out the classroom list, detail, create and edit data.









To run on iOS Device/Simulator, add the platform then build and run from XCode.

```
ionic cordova platform add ios
ionic cordova build ios
```

If you have the problem when running this app to simulator or device.

```
ERROR: Backend returned code 0, body was: [object XMLHttpRequestProgressEvent]
```

Try removing the `WKWebView` by type this command.

```
ionic cordova plugin rm cordova-plugin-ionic-webview
```

To run on Android Device/Simulator, add the platform then build and run the app.

```
ionic cordova platform add android
ionic cordova run android
```

Of course, you will get a lot of bugs or errors, because this is the beta version which still reviews and bugs findings and reporting. You can raise an issue here (https://github.com/ionic-team/ionic/issues) after there's no same issue exists.

That it's, the comprehensive step by step tutorial on building Ionic 4, Angular 6 and Cordova CRUD (Create, Read, Update, Delete) Mobile App. You can find the full source code in our GitHub (https://github.com/didinj/ionic4-angular6-crud-example.git).

We know that building beautifully designed Ionic apps from scratch can be frustrating and very time-consuming. Check Ionic 4 - Full Starter App (https://ionicthemes.com/product/ionic4-full-starter-app?ref=djamware) and save development and design time. Android, iOS, and PWA, 100+ Screens and Components, the most complete and advance Ionic Template.

That just the basic. If you need more deep learning about Ionic, Angular, and Typescript, you can take the following cheap course:

- IONIC 4 Design Hybrid Mobile Applications IOS & Android (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=507388.2259072&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fhybrid-mobile-applications-with-ionic4%2F)
- Wordpress Rest API and Ionic 4 (Angular) App With Auth (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=507388.1515202&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fwordpress-rest-api-and-ionic-3-crud%2F)
- Mobile App from Development to Deployment - IONIC 4 (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=507388.2276149&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fmobile-app-from-development-to-deployment-ionic-4%2F)
- Ionic 4 Crash Course with Heartstone API & Angular (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=507388.1861088&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fionic-4-crash-course-with-heartstone-api-angular%2F)
- Ionic 4 Mega Course: Build 10 Real World Apps (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=507388.2361638&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fionic-4-mega-course-build-real-world-apps%2F)

Thanks!

Follow

(http://www.facebook.com/djamwarecom)

(http://twitter.com/intent/follow?source=followbutton&variant=1.0&screen_name=djamware)

(http://www.pinterest.com/djamware)

(http://www.linkedin.com/in/didin-jamaludin-7a530351)

(http://www.youtube.com/channel/UCtI81hYLh2Ae_45KHkyy0vw?sub_confirmation=1)

(https://github.com/didinj)
