# Week 12: Coursework

## Angular Framework

1. This coursework starts from the solution we created for coursework #11. We need to modify it to store the information in the back-end. The steps presented are based on the solution given for coursework #11. You must perform equivalent steps if you use your own coursework #11 solution.

<https://github.com/busyQA-java-developer/busyqa-11-angular-3-coursework-complete.git>

1. Inside the app folder, create a file called *task.model.ts*. Write the structure described below in the file. This structure will hold the id and name of the task.

*export interface Task {*

*idTask?: string;*

*name: string;*

*}*

1. Modify the *app.module.ts* file to import the HttpClientModule, then add the HttpClientModule to the imports array. Please add the lines in red to the *app.module.ts* file as described below.

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

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule,

*HttpClientModule*

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. In the *app’s* component TS file, you have to do the following:
   1. First, import the HttpClient module from the ‘*@angular/common/http’* package, the map module from the ‘rxjs/operators' package and the *Task* interface into your app component. Add the following lines into your app component.

import { HttpClient } from '@angular/common/http';

import { Task } from './task.model';

import { map } from 'rxjs/operators';

* 1. Change the task variable type to *Task* and initialize it.

task: Task = <Task>{};

* 1. Change the tasks array type to Task and initialize it as an empty array.

tasks: Task[] = [];

* 1. Create a constructor and inject the httpClient Module.

constructor(private http: HttpClient) {}

* 1. Create the following methods in the app’s component TS file. Use the Firebase back-end URL created in class. Also, you can use the project we worked on in class as a reference.
     1. fetchTasks(): This method sends a GET HTTP request to the back-end. The data will come as a JSON object, so convert it into a temporary Task type array using an RxJS pipe as we did in class. Finally, in the subscribe method, assign the temporary array to the tasks array specified in part c.
     2. createTask(task: Task): This method sends a POST HTTP request to the back-end with the task object. The subscribe method calls the fetchTasks() method to refresh the tasks array with the new task created.
     3. deleteTask(idTask: string): This method sends a DELETE HTTP request to the back-end with the idTask embedded in the URL. In the subscribe method, call the fetchTasks() to refresh the tasks array.
     4. ngOnInit(): Angular calls this life cycle hook when the component is initialized. This method calls the fetchTasks() method to load the tasks array with data from the back-end.
  2. Modify the following methods in the app’s component TS file.
     1. onCreateTask(): Call the createTask(task: Task) method created in the previous step instead of pushing the task object into the Tasks array.
     2. onDeleteTask(): Call the deleteTask(idTask: string) method coded in the step before instead of splicing the Tasks array.
  3. Change the variable *task* to *task.name* to use the Task’s name property in the *app’s* component HTML template.

1. After completing the project, test it, and create a new repository on GitHub, finally commit and push your application to GitHub.
2. You can find the solution for the project in the Github repo:

<https://github.com/busyQA-java-developer/busyqa-12-angular-3-coursework-complete.git>

// End