Contents

Task 01. Import Modules

1. Добавьте в файле **app.module.ts** следующий фрагмент кода:

```
import { FormsModule } from '@angular/forms';
import { HttpModule, JsonpModule } from '@angular/http';

imports: [
    BrowserModule,
    CommonModule,
    FormsModule,
    TasksModule,
    UsersModule,
    HttpModule,
    JsonpModule,
    AppRoutingModule
]
```

Task 02. Simulating Web API

1. npm install -g json-server
2. create file \db\db.json следующего содержания

```
{
	"tasks": [
		{ "id": 1, "action": "Estimate", "priority": 1, "estHours": 8},
		{ "id": 2, "action": "Create", "priority": 2, "estHours": 8},
		{ "id": 3, "action": "Edit", "priority": 3, "estHours": 4},
		{ "id": 4, "action": "Delete", "priority": 3, "estHours": 2},
		{ "id": 5, "action": "Build", "priority": 1, "estHours": 4},
		{ "id": 6, "action": "Deploy", "priority": 2, "estHours": 8}
	]
}
```

3. run json-server from cmd: json-server --watch db\db.json

Task 03. Task Promise Service

1. Создайте файл **tasks/services/task-promise.service.ts** используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/toPromise';

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

@Injectable()
export class TaskPromiseService {
  private tasksUrl = 'http://localhost:3000/tasks';

  constructor(
    private http: Http
  ) {}

  getTasks(): Promise<Task[]> {
    return this.http.get(this.tasksUrl)
            .toPromise()
            .then( response => <Task[]>response.json())
            .catch(this.handleError);
  }

  private handleError(error: any): Promise<any> {
    console.error('An error occurred', error);
    return Promise.reject(error.message || error);
  }
}
```

2. Внесите изменения в файл **tasks/index.ts**

```
export * from './services/task-array.service';
export * from './services/task-promise.service';
```

3. Внесите изменения в файл **tasks/tasks.module.ts**

```
import {
  TaskListComponent,
  TaskComponent,
  TaskFormComponent,
  TaskArrayService,
  TaskPromiseService
} from '.';

providers: [
    TaskArrayService,
    TaskPromiseService
]
```

4. Внесите изменения в компонент **TaskListComponent**

```
import { TaskArrayService } from './../task-array-service/task-array.service';
import { TaskPromiseService } from './../';

constructor(
    private tasksService: TaskArrayService TaskPromiseService
```

```
) { }
```

Task 04. GetTask

1. Добавьте метод **getTask** в сервис **TaskPromiseService** используя следующий фрагмент кода

```
getTask(id: number): Promise<Task> {
    return this.http.get(`${this.tasksUrl}/${id}`)
            .toPromise()
            .then( response => <Task>response.json() )
            .catch( this.handleError );
  }
```

2. Внесите изменения в компонент **TaskFormComponent**

```
import { TaskArrayService } from './../task-array-service/task-array.service';
import { TaskPromiseService } from './..';

constructor(
    private tasksService: TaskArrayService,
    private tasksPromiseService: TaskPromiseService,
    private router: Router,
    private route: ActivatedRoute
  ) { }
```

3. Внесите изменения в метод **ngOnInit** компонента **TaskFormComponent**

```
if (id) {
        this.tasksService.getTask(id)
        this.tasksPromiseService.getTask(id)
          .then(task => this.task = Object.assign({}, task))
          .catch((err) => console.log(err));
}
```

Task 05. UpdateTask

1. Внесите изменения в **TaskPromiseService**

```
import { Http, Headers, RequestOptions } from '@angular/http';
```

2. Добавьте метод **updateTask** в сервис **TaskPromiseService** используя следующий фрагмент кода

```
updateTask(task: Task): Promise<Task> {
    const url = `${this.tasksUrl}/${task.id}`,
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions();

    options.headers = headers;

    return this.http.put(url, body, options)
            .toPromise()
            .then( response => <Task>response.json() )
            .catch( this.handleError );
  }
```

3. Внесите изменения в метод **saveTask** компонента **TaskFormComponent**

```
if (task.id) {
      this.tasksService.updateTask(task);
      this.tasksPromiseService.updateTask(task)
       .then( () => this.goBack() );
    }
    else {
      this.tasksService.addTask(task);
      this.goBack();
    }

    this.router.navigate(["home"]);
```

4. Внесите изменения в метод **completeTask** компонента **TaskListComponent**

```
completeTask(task: Task): void {
    task.done = true;
    this.taskgit Service.updateTask(task);
}
```

Task 06. CreateTask

1. Внесите изменения в темплейт компонента **TaskListComponent** использу следующий фрагмент разметки

```
<div>
       <button class="btn btn-primary"
              (click)="createTask()">New Task</button>
       <br><br>
       <task
         *ngFor='let task of tasks'
         [task]="task"
         (onComplete)="completeTask($event)">
       </task>
</div>
```

2. Внесите изменения в компонент **TaskListComponent**

```
import { Router } from '@angular/router';

constructor(
    private taskService: TaskPromiseService,
    private router: Router
  ) { }
```

3. Добавьте метод **createTask** в компонент **TaskListComponent** используя следующий фрагмент кода

```
createTask() {
    const link = ['add'];
    this.router.navigate(link);
  }
```

4. Внесите изменения в файл **tasks/tasks.routing.module.ts**

```
const tasksRoutes: Routes = [
  {
    path: 'home',
    component: TaskListComponent
  },
  {
    path: 'add',
    component: TaskFormComponent
  },
  {
    path: 'edit/:id',
    component: TaskFormComponent
  }
];
```

5. Добавьте метод **createTask** в сервис **TaskPromiseService** используя следующий фрагмент кода

```
createTask(task: Task): Promise<Task> {
    const url = this.tasksUrl,
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions({headers: headers});
```

```
    return this.http.post(url, body, options)
            .toPromise()
            .then( response => <Task>response.json() )
            .catch( this.handleError );
  }
```

6. Внесите изменения в метод **saveTask** компонента **TaskFormComponent** используя следующий фрагмент кода

```
if (task.id) {
      this.tasksPromiseService.updateTask(task)
            .then( () => this.goBack() );
    }
    else {
      this.tasksService.addTask(task);
      this.goBack();
    }
const method = task.id ? 'updateTask' : 'createTask';
    this.tasksPromiseService[method](task)
      .then( () => this.goBack() );
```

Task 07. DeleteTask

1. Внесите изменения в темплейт компонента **TaskComponent** используя следующий фрагмент разметки

```
<div class="panel panel-default">
	<div class="panel-heading">Task</div>
	<div class="panel-body">
		<ul>
			<li>Action: {{task.action}}</li>
			<li>Priority: {{task.priority}}</li>
			<li>Estimate Hours: {{task.estHours}}</li>
			<li>Actual Hours: {{task.actHours}}</li>
			<li>Done: {{task.done}}</li>
		</ul>
		<button class="btn btn-primary btn-sm"
			(click)="completeTask($event)">
			Done
		</button>
		<button class="btn btn-warning btn-sm"
			(click)="editTask(task)">
			Edit
		</button>
		<button class="btn btn-danger btn-sm"
			(click)="deleteTask(task)">
			Delete
		</button>
	</div>
</div>
```

2. Внесите изменения в компонент **TaskComponent** используя следующий фрагмент кода:

```
@Output() onComplete = new EventEmitter<Task>();
@Output() onDelete = new EventEmitter<Task>();

deleteTask(task: Task) {
    this.onDelete.emit(task);
}
```

3. Внесите изменения в темплейт компонента **TaskListComponent** используя следующий фрагмент разметки

```
<task
	*ngFor='let task of tasks'
	[task]="task"
	(onComplete)="completeTask($event)"
	(onDelete)="deleteTask($event)">
</task>
```

4. Добавьте метод **deleteTask** в сервис **TaskPromiseService** используя следующий фрагмент разметки

```
deleteTask(task: Task): Promise<Task> {
    const url = `${this.tasksUrl}/${task.id}`;

    return this.http.delete(url)
            .toPromise()
```

```
            .then( response => <Task>response.json())
            .catch( this.handleError );
}
```

5. Добавьте метод **deleteTask** в компонент **TaskListComponent** используя следующий фрагмент разметки

```
deleteTask(task: Task) {
    this.tasksService.deleteTask(task)
      .then(() => this.tasks = this.tasks.filter(t => t !== task))
      .catch(err => console.log(err));
  }
```

Task 08. Task Observable Service

1. Создайте файл app/rxjs-extensions.ts используя следующий фрагмент кода

```
export { Subscription } from 'rxjs/Subscription';

// Observable class extensions
import 'rxjs/add/observable/throw';

// Observable operators
import 'rxjs/add/operator/catch';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/filter';
import 'rxjs/add/operator/map';
```

2. Создайте сервис **TaskObservableService** в файле **tasks/services/task-observable.service.ts** используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { Http, Headers, Response, RequestOptions } from '@angular/http';

import { Observable } from 'rxjs/Observable';
import './../../rxjs-extensions';

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

@Injectable()
export class TaskObservableService {
  private tasksUrl = 'http://localhost:3000/tasks';

  constructor(
    private http: Http
  ) {}

  getTasks(): Observable<Task[]> {
    return this.http.get(this.tasksUrl)
            .map( this.handleData )
            .catch( this.handleError );
  }

  getTask(id: number) {

  }

  updateTask(task: Task) {

  }

  createTask(task: Task) {

  }

  deleteTask(task: Task) {

  }

  private handleData(response: Response) {
```

```
    const body = response.json();
    return body || {};
  }

  private handleError(error: any) {
    let errMsg = (error.message)
                    ? error.message
                    : error.status
                        ? `${error.status} - ${error.statusText}`
                        : 'Server error';
    console.error(errMsg);
    return Observable.throw(errMsg);
  }
}
```

3. Внесите изменения в файл **tasks/index.ts**

```
export * from './services/task-promise.service';
export * from './services/task-observable.service';
```

4. Внесите изменения в файл **tasks/tasks.module.ts**

```
import {
  TaskListComponent,
  TaskComponent,
  TaskFormComponent,
  TaskArrayService,
  TaskPromiseService,
  TaskObservableService
} from '.';
providers: [
    TaskArrayService,
    TaskPromiseService,
    TaskObservableService
  ]
```

5. Внесите изменения в файл **app\rxjs-extensions.ts**

```
export { Subscription } from 'rxjs/Subscription';
```

6. Внесите изменения в **TaskListComponent**

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Subscription } from './../../rxjs-extensions';

import { Task } from './../../models/task';
import { TaskPromiseService, TaskObservableService } from './../';

export class TaskListComponent implements OnInit, OnDestroy {

tasks: Array<Task>;
errorMessage: string;
private sub: Subscription[] = [];

constructor(
    private taskPromiseService: TaskPromiseService,
```

```
    private taskObservableService: TaskObservableService,
    private router: Router
  ) { }

ngOnInit() {
    this.tasksService.getTasks()
      .then(tasks => this.tasks = tasks)
      .catch((err) => console.log(err));

    const sub = this.taskObservableService.getTasks()
      .subscribe(
         tasks => this.tasks = tasks,
         error => this.errorMessage = <any>error
      );
    this.sub.push(sub);
  }

ngOnDestroy() {
    this.sub.forEach(sub => sub.unsubscribe());
}
```

Task 09. GetTask

1. Внесите изменения в метод **getTask** сервиса **TaskObservableService** используя следующий фрагмент кода

```
getTask(id: number) {
     return this.http.get(`${this.tasksUrl}/${id}`)
            .map( this.handleData )
            .catch(this.handleError);
}
```

2. Внесите изменения в компонент TaskFormComponent

```
// 1
import { TaskArrayService } from './../services/task-array.service';
import { TaskPromiseService, TaskObservableService } from './..';

// 2
private sub: Subscription[] = [];

// 3
constructor(
    private tasksService: TaskArrayService,
    private tasksPromiseService: TaskPromiseService,
    private tasksObservableService: TaskObservableService,
    private router: Router,
    private route: ActivatedRoute
  ) { }

// 4
ngOnInit(): void {
    this.task = new Task(null, '', null, null);

    this.sub const sub = this.route.params.subscribe(params => {
      const id = +params['id'];

      // NaN - for new task, id - for edit
      if (id) {
        this.tasksPromiseService.getTask(id)
          .then(task => this.task = Object.assign({}, task))
          .catch((err) => console.log(err));
       const s = this.tasksObservableService.getTask(id)
          .subscribe(
            task => this.task = Object.assign({}, task),
            err => console.log(err)
          );
        this.sub.push(s);
      }
    });
    this.sub.push(sub);
  }

// 5
ngOnDestroy(): void {
    this.sub.unsubscribe();
    this.sub.forEach(sub => sub.unsubscribe());
}
```

Task 10. UpdateTask and CreateTask

1. Внесите изменения в метод **updateTask** сервиса **TaskObservableService** используя следующий фрагмент кода

```
updateTask(task: Task): Observable<Task> {
     const url = `${this.tasksUrl}/${task.id}`,
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions();

    options.headers = headers;

    return this.http.put(url, body, options)
            .map( this.handleData )
            .catch(this.handleError);
  }
```

2. Внесите изменения в метод **createTask** сервиса **TaskObservableService** используя следующий фрагмент кода

```
createTask(task: Task): Observable<Task> {
    let url = this.tasksUrl,
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions();

    options.headers = headers;

    return this.http.post(url, body, options)
            .map( this.handleData )
            .catch( this.handleError );
  }
```

3. Внести изменения в компонент **TaskFormComponent** используя следующий фрагмент кода

```
import { TaskPromiseService, TaskObservableService } from './..';

constructor(
    private tasksPromiseService: TaskPromiseService,
    private tasksObservableService: TaskObservableService,
    private router: Router,
    private route: ActivatedRoute
  ) { }

const method = task.id ? "updateTask" : "createTask";
this.tasksPromiseService[method](task)
      .then( () => this.goBack() );
    const sub = this.tasksObservableService[method](task)
      .subscribe(
        () => this.goBack(),
        err => console.log(err)
      );
    this.sub.push(sub);
```

4. Внесите изменения в компонент **TaskListComponent**

```
completeTask(task: Task): void {
    task.done = true;
    this.tasksService.updateTask(task);
    const sub = this.taskObservableService.updateTask(task)
      .subscribe(
        null,
        err => console.log(err));
    this.sub.push(sub);
  }
```

Task 11. DeleteTask

1. Внесите изменения в метод **deleteTask** сервиса **TaskObservableService** используя следующий фрагмент кода

```
deleteTask(task: Task): Observable<Task> {
    const url = `${this.tasksUrl}/${task.id}`;

    return this.http.delete(url)
      .map(response => <Task>response.json())
      .catch(this.handleError);
  }
```

2. Внесите изменения в компонент **TaskListComponent** используя следующий фрагмент кода

```
// 1
import { TaskPromiseService, TaskObservableService } from './../';

// 2
constructor(
    private tasksService: TaskPromiseService,
    private taskObservableService: TaskObservableService,
    private router: Router
  ) { }

// 3
deleteTask(task: Task) {
    this.tasksService.deleteTask(task)
    .then(() => this.tasks = this.tasks.filter(t => t !== task))
    .catch(err => console.log(err));
    const sub = this.taskObservableService.deleteTask(task)
      .subscribe(
      () => this.tasks = this.tasks.filter(t => t !== task),
      err => console.log(err)
      );
    this.sub.push(sub);
  }
```