Service example: -

```
=================
main.ts
=================
import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";

import { AppModule } from "./app.module";

platformBrowserDynamic().bootstrapModule(AppModule);

==============
app.module
==============
import { NgModule } from '@angular/core';

import { BrowserModule } from "@angular/platform-browser";

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

import { StudentListComponent } from "./student-list.component";

import { AddStudentComponent } from "./add-student.component";

import { EditStudentComponent } from "./edit-student.component";

import { DeleteStudentComponent } from "./delete-student.component";

@NgModule({
   imports: [BrowserModule],
    declarations: [AppComponent,StudentListComponent,AddStudentComponent,EditStudentComponent,DeleteStudentComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

===============

app.component

===============

```
import { Component, OnInit } from '@angular/core';

import {Student} from './student';

import {StudentService} from './student.service';

@Component({
    selector: 'project-app',
    templateUrl: `app/app.template.html`,
    providers: [StudentService]
})

export class AppComponent{
  studentId: number;
  currentView = 'list';

  navigateTo(view: string) {
    this.currentView = view;
  }
}

===========
app.template.html
===========

<div class="container-fluid">
  <h1>Students Maintenance</h1>
  <div [ngSwitch]="currentView">
    <student-list-view *ngSwitchDefault
      (onAdd)="navigateTo('add')"
      (onEdit)="studentId=$event; navigateTo('edit')"
      (onDelete)="studentId=$event; navigateTo('delete')">
    </student-list-view>
    <add-student-view *ngSwitchCase="'add'"
      (onClosed)="navigateTo('list')">
    </add-student-view>
    <edit-student-view *ngSwitchCase="'edit'"
      [studentId]="studentId"
      (onClosed)="navigateTo('list')">
    </edit-student-view>
    <delete-student-view *ngSwitchCase="'delete'">
    </delete-student-view>
</div>

================
add-student.component
```

================

```
import {Component, Input, Output, EventEmitter} from '@angular/core';

import {StudentService} from './student.service'

@Component({
  selector: 'add-student-view',
  templateUrl: 'app/add-student.template.html'
})
export class AddStudentComponent {
  @Output() onClosed = new EventEmitter();
  studentName: string;

  constructor(private studentService: StudentService) {
  }



  add() {
    this.studentService.addStudent({id: 0, name:this.studentName});
    this.onClosed.emit(null);
  }

  cancel() {
    this.onClosed.emit(null);
  }
}
```

================

```
add-student.template
================


<h3>Specify the name of the Student:</h3>
<div class="row">
  <div class="col-sm-6">
    <input #studentNameBox class="form-control input-lg" type="text"
      placeholder="student name"
      (keyup)="studentName=studentNameBox.value"
      (keyup.enter)="add()"
      />
  </div>
</div>
<div class="row" style="margin-top: 12px;">
  <div class="col-sm-6">
    <button class="btn btn-success btn"
      (click)="add()"
      [disabled]="!studentName">
      Add
    </button>
    <button class="btn btn-warning btn"
      (click)="cancel()">
      Cancel
    </button>
  </div>
</div>



=============
```

delete-student.component
=============

```
import {Component, Input, Output, EventEmitter} from '@angular/core';
import {Student} from './student';

@Component({
  selector: 'delete-student-view',
  templateUrl: 'app/delete-student.template.html'
})
export class DeleteStudentComponent {
  @Input() student: Student;
  @Output() onDeleted = new EventEmitter();
  @Output() onCancel = new EventEmitter();

  deleted() {
    this.onDeleted.emit(null);
  }

  cancel() {
    this.onCancel.emit(null);
  }
}
```

===================

delete-student.template

===================

```
<h3>Confirm delete</h3>
<div class="row">
  <div class="col-sm-8">
    <p>
      Are you sure you want to delete student'{{student.name}}'?
    </p>
  </div>
</div>
<div class="row" style="margin-top: 12px;">
  <div class="col-sm-6">
    <button class="btn btn-success btn"
      (click)="cancel()">
      No
    </button>
    <button class="btn btn-danger btn"
      (click)="deleted()">
      Yes
    </button>
  </div>
</div>


==================
edit-student.component
==================

import {Component, Input, Output, EventEmitter} from '@angular/core';
import {Student} from './student';
import {StudentService} from './student.service'
```

```
@Component({
  selector: 'edit-student-view',
  templateUrl: 'app/edit-student.template.html'
})
export class EditStudentComponent {
  @Input() studentId: number;
  studentName: string;
  @Output() onClosed = new EventEmitter();

constructor(private studentService: StudentService) {
    this.studentName = this.studentService
      .getStudentById(this.studentId).name;
  }

 save() {
    this.studentService.saveStudent({id: this.studentId, name:this.studentName});
    this.onClosed.emit(null);
  }

  cancel() {
    this.onClosed.emit(null);
  }
}



=================
edit-student.template
=================
<h3>Edit the name of the student:</h3>
<div class="row">
  <div class="col-sm-6">
    <input #studentNameBox class="form-control input-lg" type="text"
      [value]="student.name"
      placeholder="student name"
      (keyup)="student.name=studentNameBox.value"
      (keyup.enter)="save()" />
  </div>
</div>
<div class="row" style="margin-top: 12px;">
  <div class="col-sm-6">
    <button class="btn btn-success btn"
      (click)="save()"
      [disabled]="!student.name">
      Save
    </button>
    <button class="btn btn-warning btn"
      (click)="cancel()">
      Cancel
    </button>
  </div>
</div>
```

```
=================
student-list.component
=================
import {Component,Input, Output, EventEmitter} from '@angular/core';
import {Student} from './student';
import {StudentService} from './student.service'

@Component({
  selector: 'student-list-view',
  templateUrl: 'app/student-list.template.html'
})
export class StudentListComponent {
  students: Student[];
  @Output() onAdd = new EventEmitter();
    @Output() onEdit = new EventEmitter<Student>();
  @Output() onDelete = new EventEmitter<Student>();

  constructor(private studentService: StudentService) {
    this.students = studentService.getAllStudents();
  }

  add() {
    this.onAdd.emit(null);
  }
  edit(studentId: number) {
    this.onEdit.emit(studentId);
  }

  delete(studentId: number) {
    this.onDelete.emit(studentId);
  }
}
```

```
==============
student-list.template
==============
<div class="row">
  <div class="col-sm-12">
    <button class="btn btn-primary btn-lg" (click)="add()">
      Add new student
    </button>
  </div>
</div>
<h2>List of Students</h2>
<div class="row" *ngFor="let student of students">
  <div class="col-sm-8">
    <h4>{{student.id}}: {{student.name}}</h4>
  </div>
  <div class="col-sm-4" style="margin-top: 5px;">
    <div class="pull-right">
      <button class="btn btn-warning btn-sm"
        (click)="edit(student.id)">
        Edit
      </button>
      <button class="btn btn-danger btn-sm"
        (click)="delete(student.id)">
        Delete
      </button>
    </div>
  </div>
```

```
</div>

================
student.service
================

import {Injectable} from '@angular/core';
import {Student} from './student';

@Injectable()
export class StudentService {
  private students = Student.students.slice(0);

  getAllStudents(): Student[] {
    return this.students.slice(0);
  }

  getStudentById(id: number): Student {
    let student = this.students.filter(s => s.id == id)[0];
    return student
      ? {id: student.id, name: student.name }
      : null;
  }

  addStudent(newStudent: Student) {
    newStudent.id = this.students.length > 0
      ? this.students.map(s => s.id).reduce((p, c) => p < c ? c : p) + 1
      : 1;
    this.students.push(newStudent);
```

```
 }

 saveStudent(student: Student) {
  let oldStudent = this.students.filter(s => s.id == student.id)[0];
  if (oldStudent) {
   oldStudent.name = student.name;
  }
 }

 deleteStudent(id: number) {
  let oldStudent = this.students.filter(s => s.id == id)[0];
  if (oldStudent) {
   let studentIndex = this.students.indexOf(oldStudent);
   if (studentIndex >= 0) {
    this.students.splice(studentIndex, 1);
   }
  }
 }
}

===========
student
===========
export class Student {
  id: number;
  name: string;

  static students: Student[] = [
    { id: 1, name: 'student1'},
    { id: 2, name: 'student2'},
    { id: 3, name: 'student3'},
    { id: 4, name: 'student4'}
  ];
}
```