Operazione Rif. P.A. 2022-17295/RER - Approvata dalla Regione Emilia Romagna con DGR 1379/2022 del 01/08/2022

Corso I.F.T.S. 2022 - 2023

"TECNICO PER LA PROGETTAZIONE E LO SVILUPPO DI APPLICAZIONI INFORMATICHE"

Periodo di svolgimento: novembre 2022 – giugno 2023

Durata: 800 ore

DISPENSE DIDATTICHE
04: Angular 2+

Modulo n° 10: LINGUAGGI DI SCRIPTING
Modulo n° 14: TECNOLOGIE PER WEB APPLICATION

Docente: DANIELE ARDUINI

Fondazione En.A.I.P. S. Zavatta Rimini
Viale Valturio, 4 47923 Rimini
Tel. 0541.367100 – fax. 0541.784001
www.enaiprimini.org; e-mail: info@enaiprimini.org

Angular 2, 3, ..., 9

Framework per Single Page Application (SPA)

- ~~il binding bidirezionale (two-way binding)~~
- la dependency injection
- il supporto al pattern MVC
- il supporto ai moduli
- la separazione delle competenze
- la testabilità del codice
- la riusabilità dei componenti

Angular 2/3/5 vs AngularJS

AngularJS

- JavaScript
- il binding bidirezionale (two-way binding)
-

Angular 2 ~ 3 ~ 5

- TypeScript
- il binding bidirezionale (two-way binding) (sconsigliato)
- binding:
  - interpolazione di variabile
  - property binding
  - event binding

Documentazione

Riferimento ufficiale:

- Angular 🇬🇧
  https://angular.io/docs

💰 : a pagamento
🇮🇹🇬🇧: lingua

Tutorial e corsi on-line:

- Guida Angular 2 🇮🇹
  (https://www.html.it/guide/guida-angularjs-2/)

- Guida Angular 7 🇮🇹
  (https://www.mrwebmaster.it/javascript/guida-angular/)

Video corsi:

- Angular - The Complete Guide (2020 Edition) 🇬🇧 💰
  https://www.udemy.com/course/the-complete-guide-to-angular-2/

- ANGULAR. FROM THEORY TO PRACTICE 🇬🇧
  https://codecraft.tv/courses/angular/

Indice:

- Setup ed Introduzione
- Componenti e Databinding
- Direttive
- Servizi e Dependency Injection
- Routing
- Observables
- Forms
- Pipes
- Http
- Autenticazione

Setup Ambiente

GIT

gestione versioni dei sorgenti di progetto

- download e install Git:
  http://git-scm.com/download

- test:

  c:\> git --version

Setup Ambiente

SourceTree

interfaccia grafica intuitiva per GIT

- download e install SourceTree: https://www.sourcetreeapp.com

- Al primo avvio registrare un account Atlassian

Setup Ambiente

Node.js

- node: interprete Javascript “fuori dal browser”
- npm: gestore pacchetti node (per sviluppare applicazione)

- download e install versione LTS:
  https://nodejs.org/en/download/

- test:

  c:\> node --version

Setup Ambiente

Angular CLI (Command Line Interface)

- Ora tutto gestito tramite “npm” e “@angular/cli”

- `c:\>` **`npm install -g @angular/cli`**

Verifica:

- `c:\>` **`ng version`**

Dependency Injection

TODO

Moduli / Componenti

TODO

Routing

TODO

Generare una applicazione

- Per generare una nuova applicazione:

  c:\> **ng new --defaults my-first-app**

- Viene creata la cartella **my-first-app** ed al suo interno vengono creati i file che rappresentano la nuova applicazione.
- Se GIT è installato, viene creato anche un repository locale per consentire di inviarlo ad un repository remoto.

- Avvio dell'applicazione:

  c:\> **cd my-first-app**
  c:\my-first-app> **ng serve**

Scaricare la app da GIT

- Clonare il repository (es. con TortoiseGIT): **https://bitbucket.org/enaip-ifts/my-first-app-2021.git**

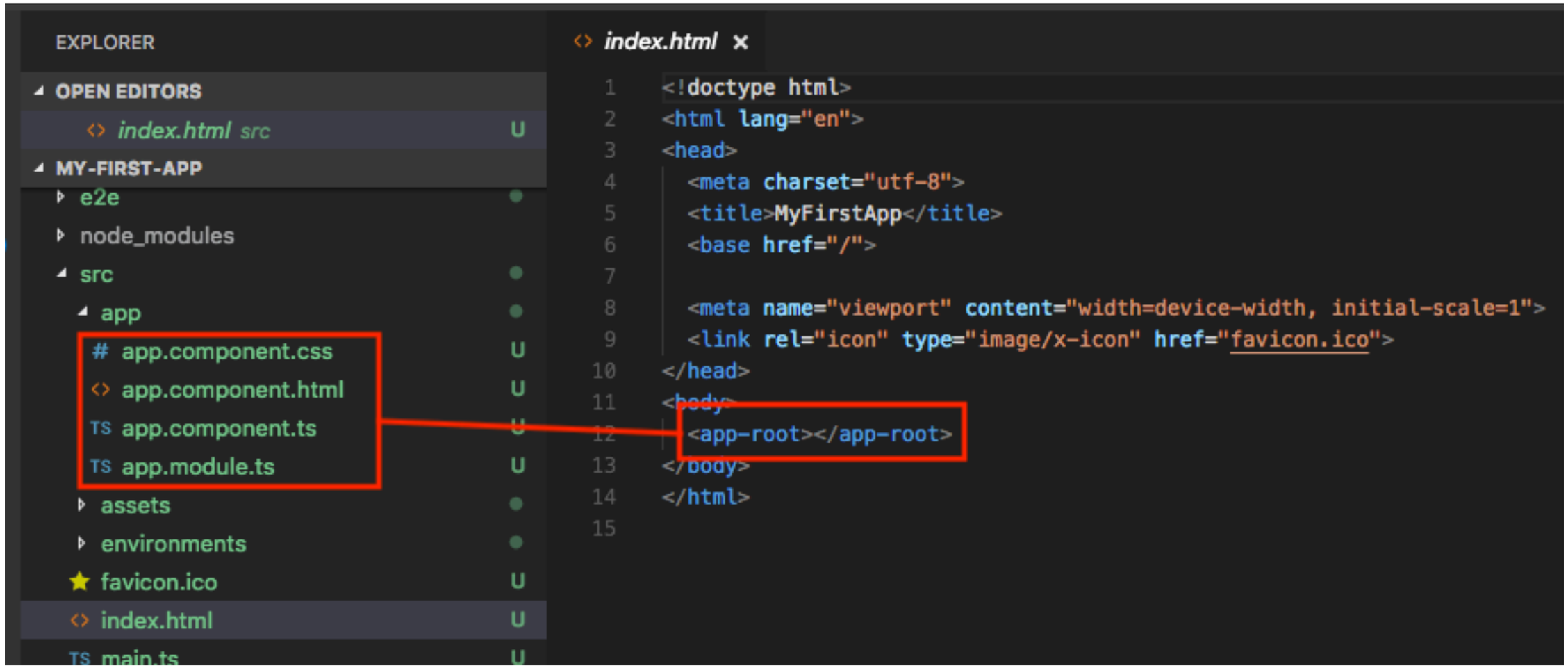
- Posizionarsi all'interno della cartella my-first-app-2021 con un prompt comandi

  (o in alternativa aprire la cartella in Visual Studio Code ed aprire un terminale in esso)

- `c:\my-first-app-2021>` **`npm install`**
- `c:\my-first-app-2021>` **`ng serve`**

App Root

Radice dell’applicazione: **<app-root>** in index.html
notare che non è un tag HTML standard, pertanto il browser lo ignorerebbe.
E' Angular che lo gestisce sostituendolo con l'HTML prodotto dall'applicazione!



Come “ragiona” Angular al boot?
Come fa a sapere come sostituire il tag **<app-root>** con la mia applicazione?

Angular bundle

index.html (in sviluppo)

```
<!doctype html>
<html lang="en">
<head>
…
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

index.html (nel browser)

```
  <!DOCTYPE html>
▼ <html lang="en">
    <script type="text/javascript">window["_gaUserPrefs"] = { ioo : function() { return true; } }</script>
  ▶ <head>…</head>
  ▼ <body> = $0
    ▶ <app-root _nghost-c0 ng-version="5.2.3">…</app-root>
      <script type="text/javascript" src="inline.bundle.js"></script>
      <script type="text/javascript" src="polyfills.bundle.js"></script>
      <script type="text/javascript" src="styles.bundle.js"></script>
      <script type="text/javascript" src="vendor.bundle.js"></script>
      <script type="text/javascript" src="main.bundle.js"></script>
    </body>
  </html>
```

Aggiunti da Angular!

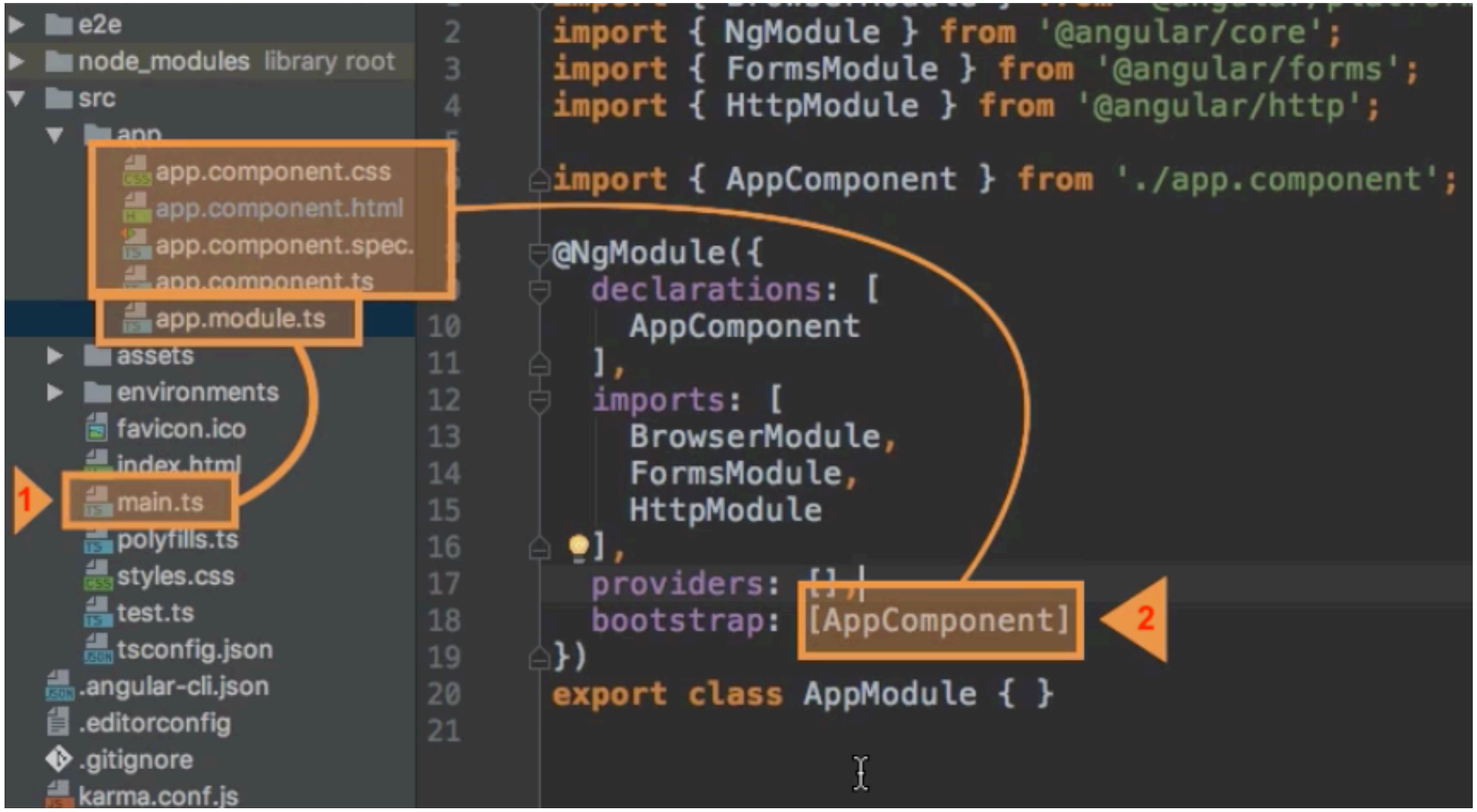
main.ts

main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```

- Rappresenta il punto d'ingresso di Angular
- Ha il compito di:
  - inizializzare i parametri di esecuzione
  - caricare il modulo principale della nostra applicazione (AppModule)
  - costruirsi in memoria la mappa di corrispondenza tra i tag HTML ed i componenti dell'applicazione

app.module.ts



- Un modulo Angular è una classe TypeScript con associato il "decorator" **@NgModule**
- Al decoratore **@NgModule** si passa un oggetto JSON che ha il compito di:
  - elencare i Component presenti al suo interno (**declarations: [...]**)
  - elencare altri moduli da cui dipende (**imports: [...]**)
  - elencare i componenti che posso essere utilizzati da altri moduli (**exports: [...]**)

app.component.ts

app.component.ts

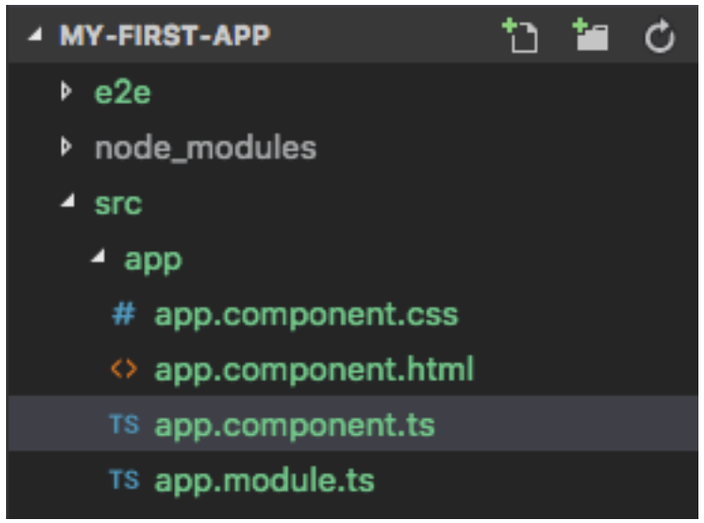
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'My First Application';
}
```

- Un componente Angular è una classe TypeScript con associato il "decorator" **@Component**
- Al decoratore **@Component** si passa un oggetto JSON che ha il compito di:
  - Definire come il Component si "attiva" nell'HTML, in questo caso associato al tag **<app-root>** (**selector: 'app-root'**)
  - Definire il template HTML da sostituire al tag (**templateUrl: './app.component.html'**)
  - Definire gli stili CSS da utilizzare (**styleUrls: ['./app.component.css']**)
- Nel corpo della classe si definisce il comportamento del componente

polyfills.ts

- Utilizzato per:
  - Aggiungere i Browser polyfills, componenti per la compatibilità di Angular all'interno di vecchie versioni di Browser
  - Aggiungere componenti JavaScript extra dell'applicazione
- https://angular.io/guide/browser-support

Hello World!




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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Hello World!';
}
```

Nuovo Component / 1

./server/server.component.ts

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

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html'
})
export class ServerComponent {

}
```

Creare i file ed inserire il contenuto

./server/server.component.html

```
<h3>Io sono un server</h3>
```

Nuovo Component / 2

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

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

@NgModule({
  declarations: [
    AppComponent,
    ServerComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Attenzione, aggiungere anche la virgola!

Nuovo Component / 3

app.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
</div>
<hr>
<app-server></app-server>
```

http://localhost:4200

Welcome to My First Application!

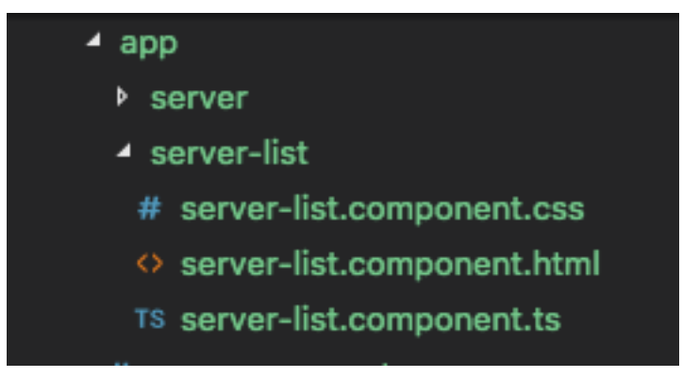
Io sono un server

Generatore di Component / 1

Utilizzo di ng:
ng help

Scorciatoia:
ng g c server-list

```
c:my-first-app> ng generate component server-list

  create src/app/server-list/server-list.component.css (0 bytes)
  create src/app/server-list/server-list.component.html (30 bytes)
  create src/app/server-list/server-list.component.spec.ts (657 bytes)
  create src/app/server-list/server-list.component.ts (288 bytes)
  update src/app/app.module.ts (559 bytes)
```



server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

server-list.component.html

```
<app-server></app-server>
<app-server></app-server>
```

Generatore di Component / 2

app.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
</div>
<hr>
<app-server-list></app-server-list>
```

http://localhost:4200

Welcome to My First Application!

Io sono un server

Io sono un server

Aspetto di un Component

app.component.html

```
<div class="container">
  <div class="row">
    <div class="col-xs-12">
        <h3>Welcome to {{ title }}!</h3>
        <hr>
        <app-server-list></app-server-list>
    </div>
  </div>
</div>
```

app.component.css

```
h3 {
    color: blue;
}
```

http://localhost:4200

Welcome to My First Application!

Io sono un server

Io sono un server

Component selector “element”

server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  constructor() { }

  ngOnInit() {
  }
}
```

app.component.html

```
<div class="container">
  <div class="row">
    <div class="col-xs-12">
        <h3>Welcome to {{ title }}!</h3>
        <hr>
        <app-server-list></app-server-list>
   </div>
  </div>
</div>
```

Component selector “attribute”

server-list.component.ts

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

@Component({
  selector: '[app-server-list]',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  constructor() { }

  ngOnInit() {
  }
}
```

app.component.html

```
<div class="container">
  <div class="row">
    <div class="col-xs-12">
        <h3>Welcome to {{ title }}!</h3>
        <hr>
        <div app-server-list></div>
   </div>
  </div>
</div>
```

Component selector “class”

server-list.component.ts

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

@Component({
  selector: '.app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  constructor() { }

  ngOnInit() {
  }
}
```

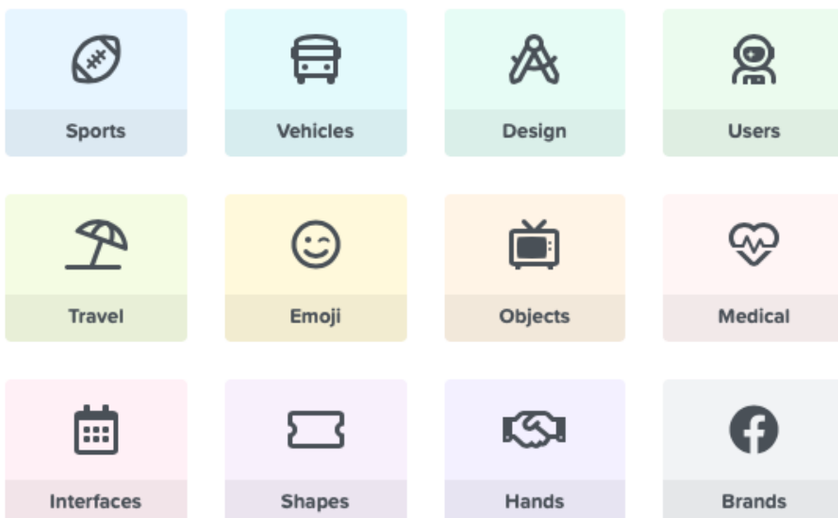
app.component.html

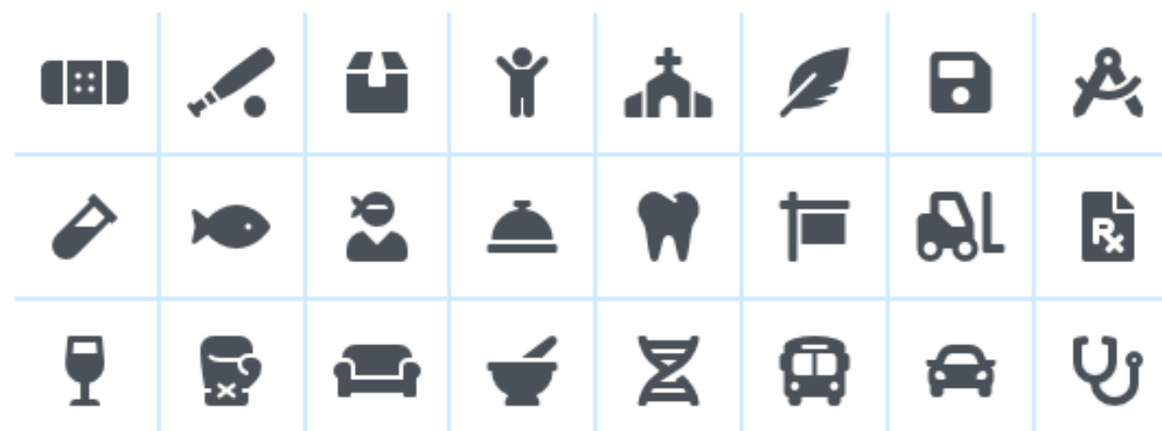
```
<div class="container">
  <div class="row">
    <div class="col-xs-12">
        <h3>Welcome to {{ title }}!</h3>
        <hr>
        <div class="app-server-list"></div>
   </div>
  </div>
</div>
```

Font Awesome: Icone

- https://fontawesome.com

Font Awesome





address-card

Regular Style (far) • • f2bb • •

Installare Font Awesome / 1

- Guida:
  https://fontawesome.com/how-to-use/on-the-web/using-with/angular
  https://www.npmjs.com/package/@fortawesome/angular-fontawesome

- Aprite un terminale sulla radice del progetto:
- c:\> **cd my-first-app**

- c:\my-first-app> **npm install @fortawesome/fontawesome-svg-core**
- c:\my-first-app> **npm install @fortawesome/free-solid-svg-icons**
- c:\my-first-app> **npm install @fortawesome/free-regular-svg-icons**
- c:\my-first-app> **npm install @fortawesome/free-brands-svg-icons**
- c:\my-first-app> **npm install @fortawesome/angular-fontawesome@0.6.x**

- **OPPURE:**

- c:\my-first-app> **ng add @fortawesome/angular-fontawesome@0.6.x**

  (e selezionare i font desiderati)

Installare Font Awesome / 2

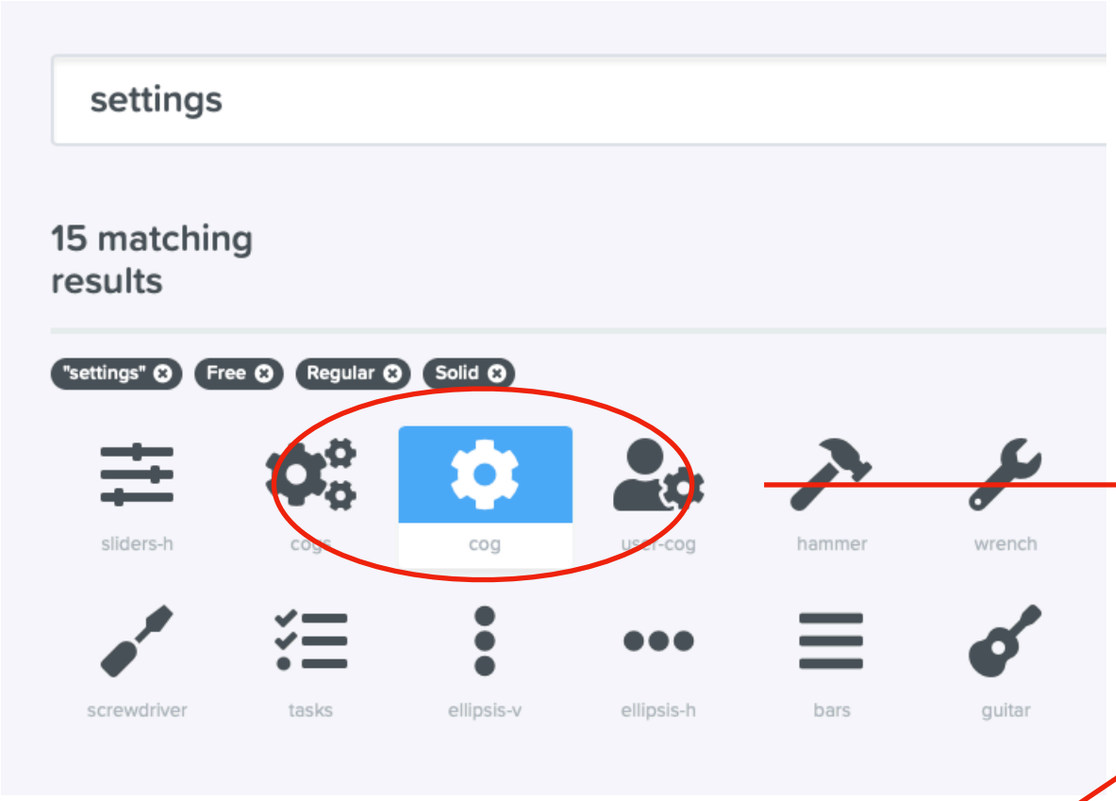
app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

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

import { CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { FontAwesomeModule, FaIconLibrary } from '@fortawesome/angular-fontawesome';
import { fas } from '@fortawesome/free-solid-svg-icons';
import { far } from '@fortawesome/free-regular-svg-icons';

@NgModule({
  schemas: [ CUSTOM_ELEMENTS_SCHEMA ],
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
    FontAwesomeModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {
  constructor(library: FaIconLibrary) {
    library.addIconPacks(fas, far);
  }
}
```

Completiamo l'installazione aggiungendo le parti in grassetto, per semplificare l'utilizzo dentro i template dei componenti e consentire a Visual Studio Code di riconoscere i tag delle icone

https://github.com/FortAwesome/angular-fontawesome/blob/master/docs/usage/icon-library.md

Installare Font Awesome / 3



Scegliere l'icona dal sito
https://fontawesome.com/icons?d=gallery

'cog'

Utilizzare il nome dell'icona nel template con il tag <fa-icon>

(all'interno di un template).html

```
...

  <button (click)="onImpostazioni()">
   <fa-icon icon="cog"></fa-icon>

   <fa-icon [icon]="['fas', 'cog']"></fa-icon>

  </button>

...
```

Risultato nella pagina

Installare Font Awesome / 4

- Ultimamente **@fortawesome/angular-fontawesome** ha problemi di compatibilità con
  angular 9.x, in tal caso utilizzare l'installazione solo HTML, senza componenti Angular:

- Aprite un terminale sulla radice del progetto:
- c:\> **cd my-first-app**

- `c:\my-first-app>` **`npm install @fortawesome/fontawesome-free`**

- `in 'src/app/styles.css' aggiungere la riga:`

  ```
  @import '~@fortawesome/fontawesome-free/css/all.css';
  ```

Bootstrap: CSS Framework

Bootstrap

Al momento la versione stabile è la 4.x

- https://getbootstrap.com/

- Bootstrap è il più diffuso framework HTML, CSS, and JS per sviluppare pagine web "responsive" (mobile first).

- Componenti:
  - Stili CSS
  - Javascript (dipende da JQuery)

→ **HTML**

Installare Bootstrap 4.x

- 4 Metodi ALTERNATIVI:

1. Aggiungere gli stili Bootstrap in "index.html"
2. Aggiungere gli stili Bootstrap in "angular.json"
3. Aggiungere gli stili Bootstrap in "styles.css"
4. Aggiungere Bootstrap con componenti "ng-bootstrap"

Installare Bootstrap: 1. in index.html

- Aprite un terminale sulla radice del progetto:
- c:\> **cd my-first-app**
- `c:\my-first-app>` **`npm install --save bootstrap`**

- Aggiungere in "index.html":

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MyFirstApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.css">
</head>
<body>
 <app-root></app-root>
 <script src="../node_modules/jquery/dist/jquery.js"></script>
 <script src="../node_modules/bootstrap/dist/js/bootstrap.js"></script>
</body>
</html>
```

Installare Bootstrap: 2. in angular.json

- Aprite un terminale sulla radice del progetto:
- c:\> **cd my-first-app**
- `c:\my-first-app>` **`npm install --save bootstrap`**

- Aggiungere il riferimento dentro "angular.json":

```
"styles": [
  "../node_modules/bootstrap/dist/css/bootstrap.min.css",
  "styles.css"
],
```

- Si occuperà Angular a recuperare js e css per inserirli in automatico in “index.html” al prossimo…
- `c:\my-first-app>` **`ng serve`**

Installare Bootstrap: 3. in src/styles.css

- Aprite un terminale sulla radice del progetto:
- c:\> **cd my-first-app**
- `c:\my-first-app>` **`npm install --save bootstrap`**

- Aggiungere il riferimento dentro “src/styles.css”:

```
@import "~bootstrap/dist/css/bootstrap.css";
```

- Si occuperà Angular a recuperare css per inserirli in automatico in “index.html” al prossimo…
- `c:\my-first-app>` **`ng serve`**

Installare Bootstrap: 4. con ng-bootstrap

- Con riferimento alla documentazione (https://ng-bootstrap.github.io/#/getting-started):
- Aprite un terminale sulla radice del progetto:
- c:\> **cd my-first-app**
- `c:\my-first-app>` **`ng add @angular/localize`**
- `c:\my-first-app>` **`npm install --save bootstrap`**
- `c:\my-first-app>` **`npm install --save @ng-bootstrap/ng-bootstrap`**

src/app/app.module.ts

```
...
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';


@NgModule({
 ...
 imports: [
   ...
   NgbModule,
  ],
  ...
})
export class AppModule { }
```

src/styles.css

```
@import "~bootstrap/dist/css/bootstrap.css";
```

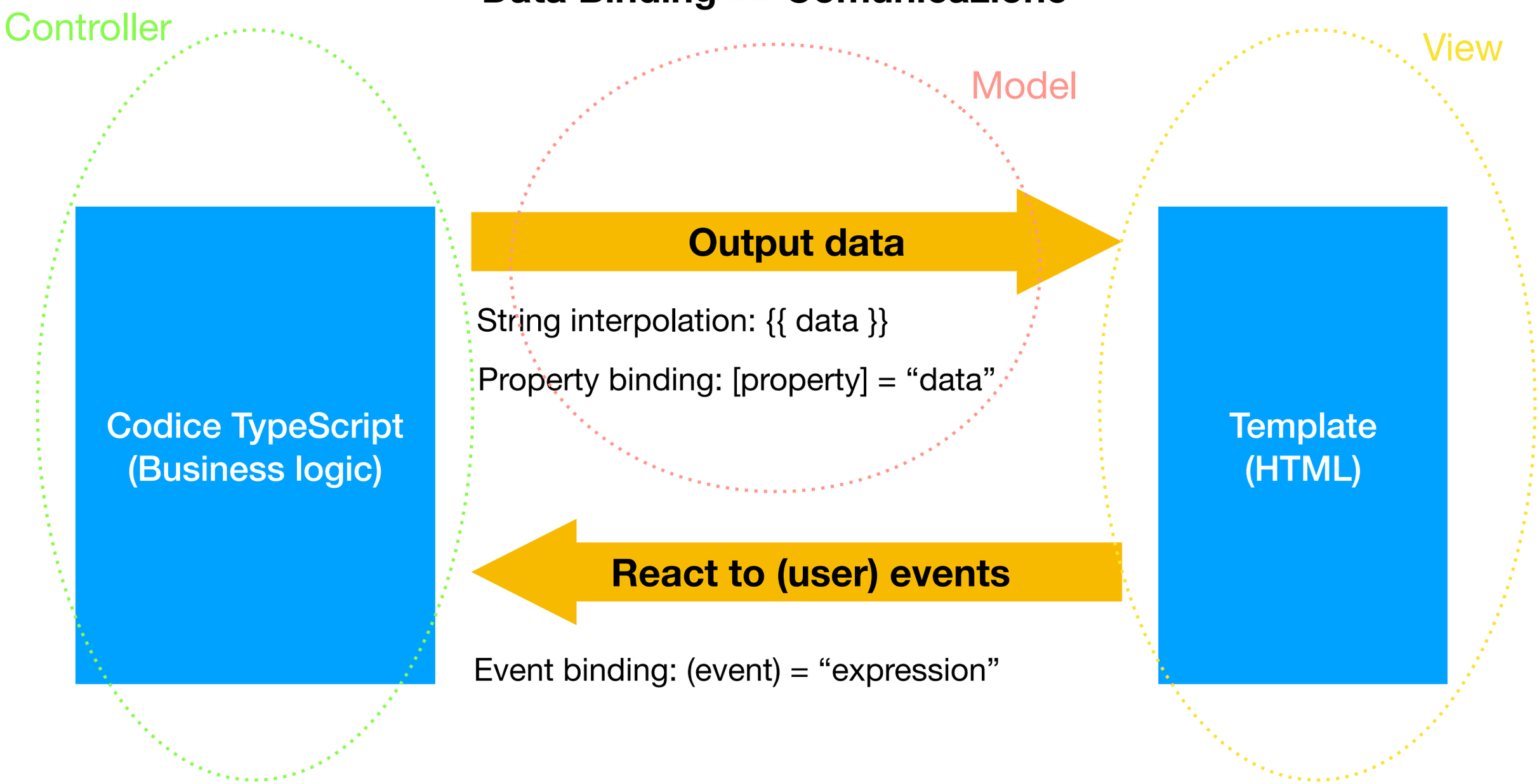
Esercizio: crea Component

Aggiungi 2 nuovi componenti:

- **Un componente per visualizzare degli avvertimenti di tipo Warning**
- **Un componente per visualizzare degli avvertimenti di tipo Success**

Data Binding

Data Binding == Comunicazione



Combinazione dei 2: 2-way binding: [(ngModel)] = “data”

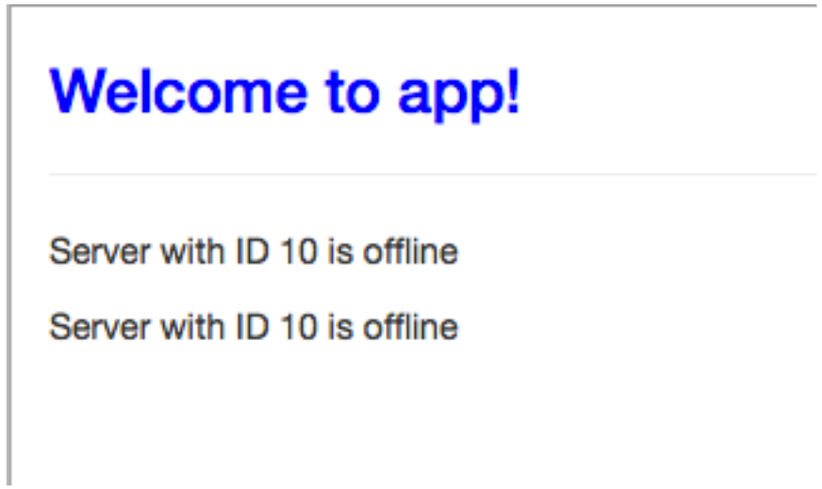
String interpolation

server.component.html

```
<p>{{ 'Server' }} with ID {{ serverId }} is {{ getServerStatus() }}</p>
```

server.component.ts

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

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html'
})
export class ServerComponent {
    serverId: number = 10;
    serverStatus: string = 'offline';

    getServerStatus() {
        return this.serverStatus;
    }
}
```



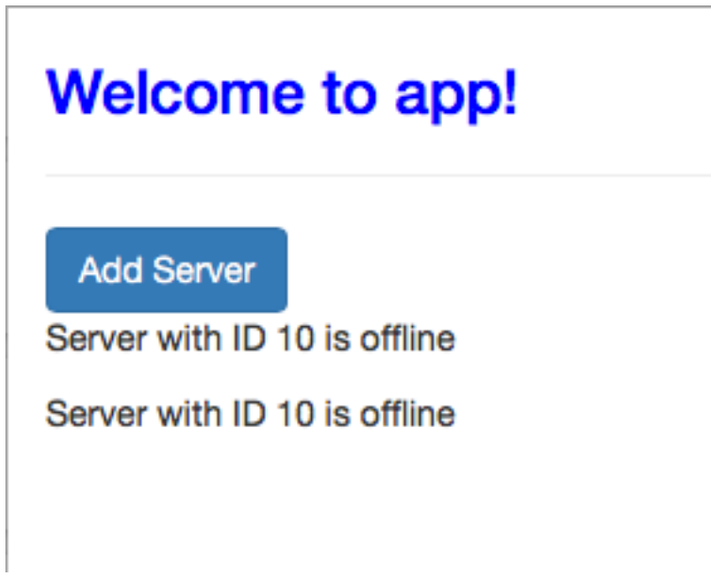
Property binding

server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  allowNewServer = false;

  constructor() {
    setTimeout(() => {
      this.allowNewServer = true;
    }, 2000);
  }

  ngOnInit() {
  }

}
```

server-list.component.html

```
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
>Add Server</button>

<app-server></app-server>
<app-server></app-server>
```



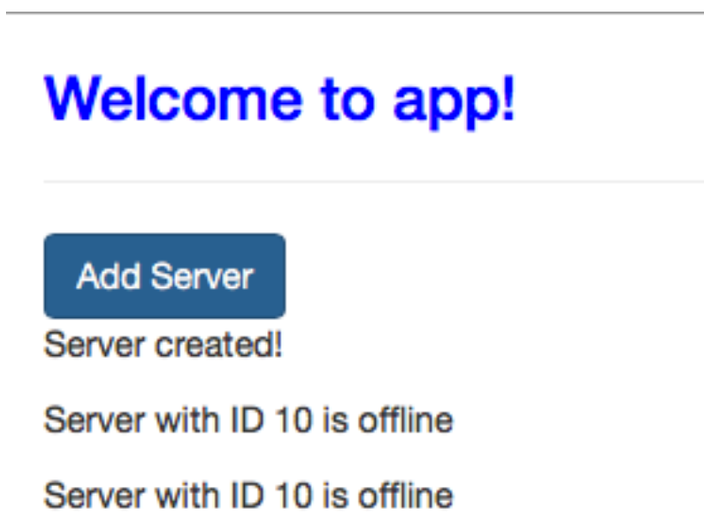
Event binding

server-list.component.ts

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

@Component({
 selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  allowNewServer = false;
  serverCreationStatus = 'No server created';

  constructor() {
    setTimeout(() => {
      this.allowNewServer = true;
    }, 2000);
  }

  ngOnInit() {
  }

  onCreateServer() {
    this.serverCreationStatus = 'Server created!';
  }
}
```

server-list.component.html

```
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>

<p>{{ serverCreationStatus }}</p>

<app-server></app-server>
<app-server></app-server>
```



Riferimenti:

https://angular.io/guide/event-binding
https://developer.mozilla.org/en-US/docs/Web/Events
https://www.w3schools.com/jsref/dom_obj_event.asp

Event binding parameters

server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  allowNewServer = false;
  serverCreationStatus = 'No server created';
  serverName = '';

...

  onUpdateServerName(event: InputEvent) {
    this.serverName =
            (<HTMLInputElement>event.target).value;

    // nuova sintassi a partire da Typescript >= 1.6.x
    this.serverName =
            (event.target as HTMLInputElement).value;
 }

}
```

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    (input)="onUpdateServerName($event)"
>
<p>{{ serverName }}</p>
…
```

Event binding: flusso

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    (input)="onUpdateServerName($event)"
>
<p>{{ serverName }}</p>
…
```

```
serverName = '';



onUpdateServerName(event: Event) {
   this.serverName =
            (<HTMLInputElement>event.target).value;
}
```

2-way binding

server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  allowNewServer = false;
  serverCreationStatus = 'No server created';
  serverName = '';

...
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
...
@NgModule({
...
 imports: [
    BrowserModule,
    FormsModule
  ],
...
})
export class AppModule { }
```

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
…
```

Directive

Le “Directive” sono speciali “Component” senza template

```
<p appTurnGreen>Receives a green background!</p>
```

```
@Directive({
   selector: '[appTurnGreen]'
})
export class TurnGreenDirective {
   ...
}
```

3 tipologie di “Directive”:

1. **Component**: Directive con template e css
2. **Structural directive**: modifica la struttura del DOM, aggiunge o rimove elementi
3. **Attribute directive**: modifica l’aspetto o il comportamento di un elemento

Directive built-in: *ngIf

server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent implements OnInit {
  allowNewServer = false;
  serverCreationStatus = 'No server created';
  serverName = '';
  serverWasCreated = false;

...

  onCreateServer() {
    this.serverWasCreated = true;
    this.serverCreationStatus = 'Server created!';
  }

}
```

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>
<p *ngIf="serverWasCreated">
   A new server was created with
   name: {{ serverName }}
</p>
…
```

Directive built-in: *ngIf + else

server-list.component.html

caso 1

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>

<p *ngIf="serverWasCreated">
   A new server was created with
   name: {{ serverName }}
</p>
<p *ngIf="!serverWasCreated">
   No server created
</p>

…
```

caso 2

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>

<p *ngIf="serverWasCreated; else noServer">
   A new server was created with
   name: {{ serverName }}
</p>
<ng-template #noServer>
  <p>No server created</p>
</ng-template>
…
```

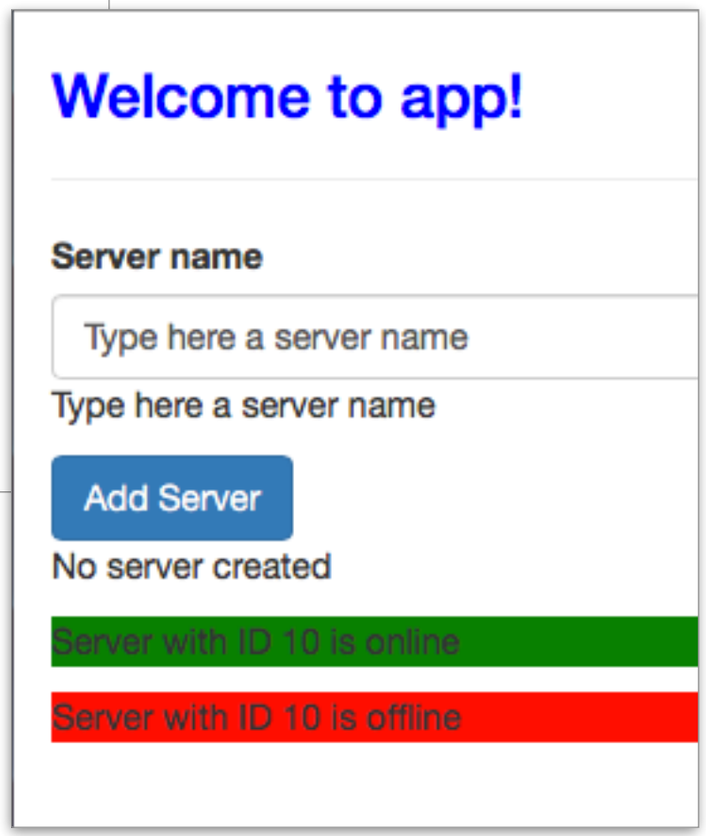
Directive built-in: ngStyle

server.component.ts

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

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html'
})
export class ServerComponent {
    serverId: number = 10;
    serverStatus: string = 'offline';

    constructor() {
        this.serverStatus = (Math.random() > 0.5 ? 'online' : 'offline');
    }

    getServerStatus() {
        return this.serverStatus;
    }

    getColor() {
        return (this.serverStatus === 'online' ? 'green' : 'red');
    }
}
```

server.component.html

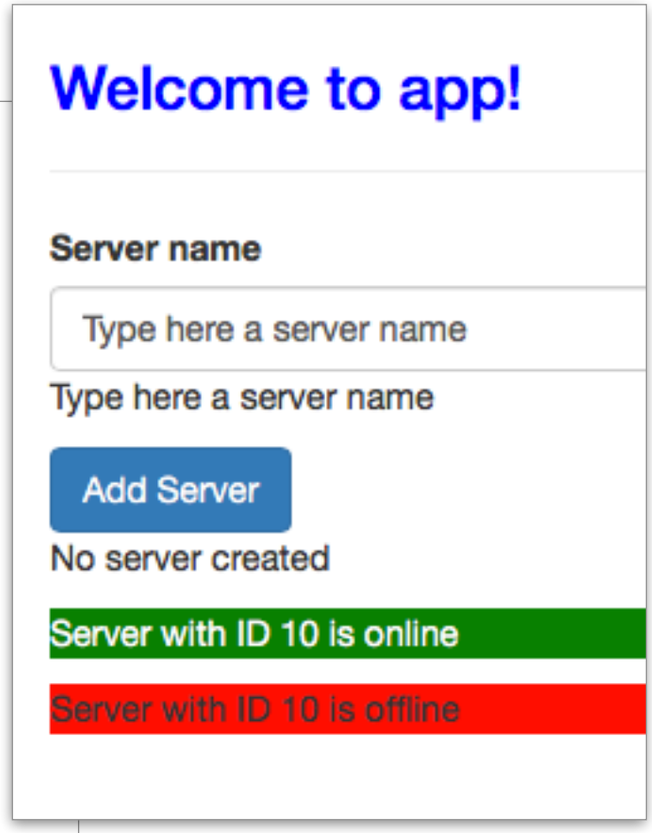
```
<p [ngStyle]="{ backgroundColor: getColor() }">
{{ 'Server' }} with ID {{ serverId }} is {{ serverStatus }}
</p>
```



Directive built-in: ngClass

server.component.ts

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

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html',
    styles: [`
        .online {
            color: white;
        }
    `]
})
export class ServerComponent {
    serverId: number = 10;
    serverStatus: string = 'offline';

    constructor() {
        this.serverStatus = (Math.random() > 0.5 ? 'online' : 'offline');
    }

    getColor() {
        return (this.serverStatus === 'online' ? 'green' : 'red');
    }
}
```



server.component.html

```
<p [ngStyle]="{ backgroundColor: getColor() }"
   [ngClass]="{ online: serverStatus === 'online' }"
>
    {{ 'Server' }} with ID {{ serverId }} is {{ serverStatus }}
</p>
```

Directive built-in: *ngFor

server-list.component.ts

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

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent {
  allowNewServer = false;
  serverCreationStatus = 'No server created';
  serverName = 'Type here a server name';
  serverWasCreated = false;
  serverList = [ 'Server 1', 'Server 2' ];

...

  onCreateServer() {
    this.serverWasCreated = true;
    this.serverList.push(this.serverName);
    this.serverCreationStatus = 'Server created!';
  }

}
```

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>
<p *ngIf="serverWasCreated; else noServer">
   New server created with name:
   {{ serverName }}
</p>
<ng-template #noServer>
    <p>No server created</p>
</ng-template>

<app-server *ngFor="let server of serverList">
</app-server>
```

Directive built-in: *ngFor + *ngIf

ATTENZIONE:
Non è possibile utilizzare più di una direttiva “strutturale” (quelle che iniziano per *) sullo stesso tag

server-list.component.html

ERRORE!

```
…

<app-server
    *ngIf="serverList.length > 0"
    *ngFor="let server of serverList">
</app-server>
```

Alternativa OK: con ng-container

```
…
<ng-container *ngIf="serverList.length > 0">
  <app-server
      *ngFor="let server of serverList">
  </app-server>
</ng-container>
```

Alternativa OK: con hidden (nell’HTML rimane!)

```
…

<app-server
    [hidden]="serverList.length > 0"
    *ngFor="let server of serverList">
</app-server>
```

*ngIf vs [hidden]

*ngIf=""	[hidden]=""
Direttiva Angular strutturale	Normale attributo HTML5
Nessun elemento DOM viene aggiunto se l’espressione è falsa	L’elemento DOM viene aggiunto comunque
Può essere lento se aggiunte e rimozioni sono molto frequenti, causa ripetute inizializzazioni del componente	Veloce se visualizza e nascondi sono frequenti perché l’elemento rimane sempre nel DOM
Contestualmente vengono aggiunti/ rimossi anche gli elementi/eventi collegati	Gli elementi/eventi collegati rimangono sempre attivi
Ideale quando il componente è complesso	Ideale quando il componente è semplice

Comunicazione tra Component: @Input()

server.component.ts

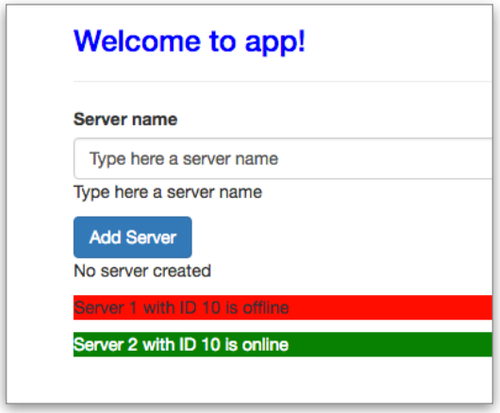
```
import { Component, Input } from '@angular/core';

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html',
    styles: [`
        .online {
            color: white;
        }
    `]
})
export class ServerComponent {
    @Input() serverName: string = '';

    serverId: number = 10;
    serverStatus: string = 'offline';

...
}
```

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>
<p *ngIf="serverWasCreated; else noServer”>
   New server created with name: {{ serverName }}
</p>
<ng-template #noServer>
    <p>No server created</p>
</ng-template>
<app-server *ngFor="let server of serverList"
            [serverName]="server"></app-server>
```

server.component.html

```
<p [ngStyle]="{ backgroundColor: getColor() }"
   [ngClass]="{ online: serverStatus === 'online' }"
>
    {{ serverName }} with ID {{ serverId }} is {{ serverStatus }}
</p>
```



Comunicazione tra Component: eventi interni

server.component.ts

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

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html',
    styles: [`
        .online {
            color: white;
        }
    `]
})
export class ServerComponent {
...

    onReboot() {
      this.serverStatus = 'offline';
      setTimeout(() => {
        this.serverStatus = 'online';
      }, 2000);
    }
}
```

Generazione di evento all'interno del componente stesso

server.component.html

```
<p [ngStyle]="{ backgroundColor: getColor() }"
   [ngClass]="{ online: serverStatus === 'online' }"
>
    {{ serverName }} with ID {{ serverId }} is {{ serverStatus }}

   <span style="float: right;">
    <button (click)="onReboot()"><i class="fas fa-sync"></i></button>
   </span>
</p>
```

PS: icone da font-awesome
https://fontawesome.com/v4.7.0/icons/

Comunicazione tra Component: @Output() /1

server.component.ts

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

@Component({
    selector: 'app-server',
    templateUrl: './server.component.html',
    styles: [`
        .online {
            color: white;
        }
    `]
})
export class ServerComponent {
  @Output() removed = new EventEmitter<string>();

  ...

  onRemove() {
    this.removed.emit(this.serverName);
  }
}
```

Generazione di evento verso l'esterno

server.component.html

```
<p [ngStyle]="{ backgroundColor: getColor() }"
   [ngClass]="{ online: serverStatus === 'online' }"
>
    {{ serverName }} with ID {{ serverId }} is {{ serverStatus }}

  <span style="float: right;">
    <button (click)="onReboot()"><i class="fas fa-sync"></i></button>
    <button (click)="onRemove()"><i class="fas fa-trash"></i></button>
  </span>
</p>
```

Comunicazione tra Component: @Output() /2

server-list.component.ts

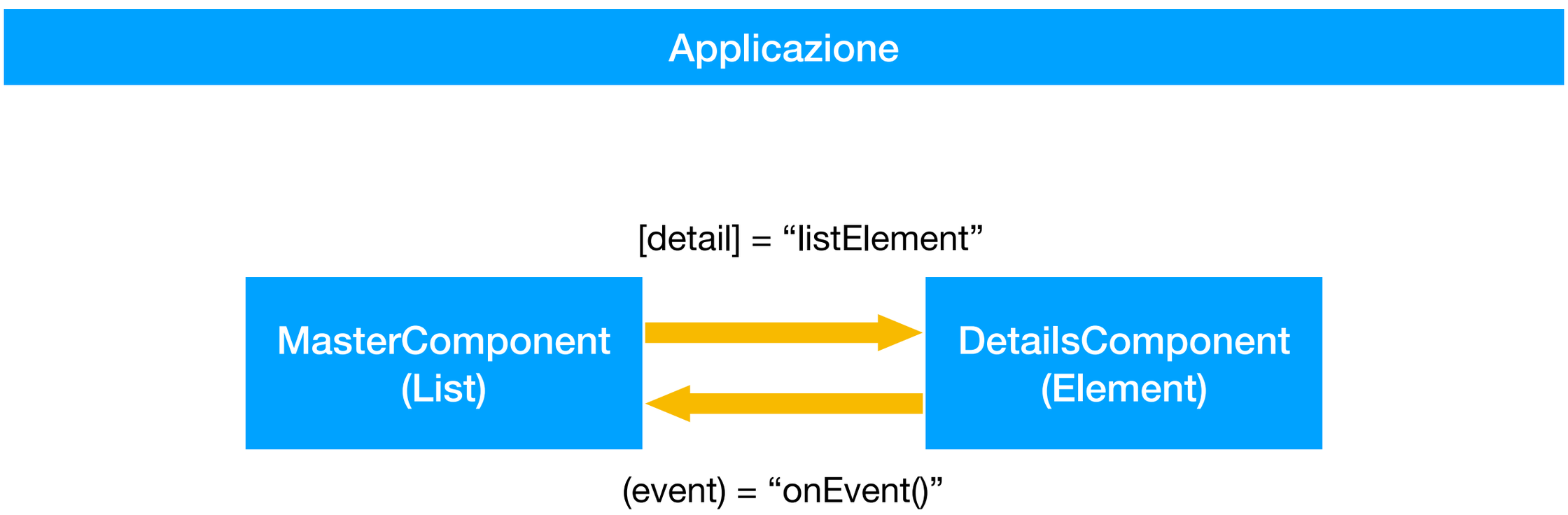
```
import { ServerComponent } from './../server/server.component';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-server-list',
  templateUrl: './server-list.component.html',
  styleUrls: ['./server-list.component.css']
})
export class ServerListComponent {
  allowNewServer = false;
  serverCreationStatus = 'No server created';
  serverName = 'Type here a server name';
  serverWasCreated = false;
  serverList = [ 'Server 1', 'Server 2' ];

...

  onServerRemoved(serverRemoved: string) {
    console.log("onServerRemoved(): ", serverRemoved);

    for (let i = 0; i < this.serverList.length; i++) {
      const server = this.serverList[i];
      if (server === serverRemoved) {
        this.serverList.splice(i, 1);
        break;
      }
    }
  }
}
```

server-list.component.html

```
<label>Server name</label>
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName"
>
<p>{{ serverName }}</p>
<button
    class="btn btn-primary"
    [disabled]="!allowNewServer"
    (click)="onCreateServer()"
>Add Server</button>
<p *ngIf="serverWasCreated; else noServer”>
   New server created with name: {{ serverName }}
</p>
<ng-template #noServer>
    <p>No server created</p>
</ng-template>
<app-server *ngFor="let server of serverList"
            [serverName]="server"
            (removed)="onServerRemoved($event)">
</app-server>
```

Services & Dependency Injection

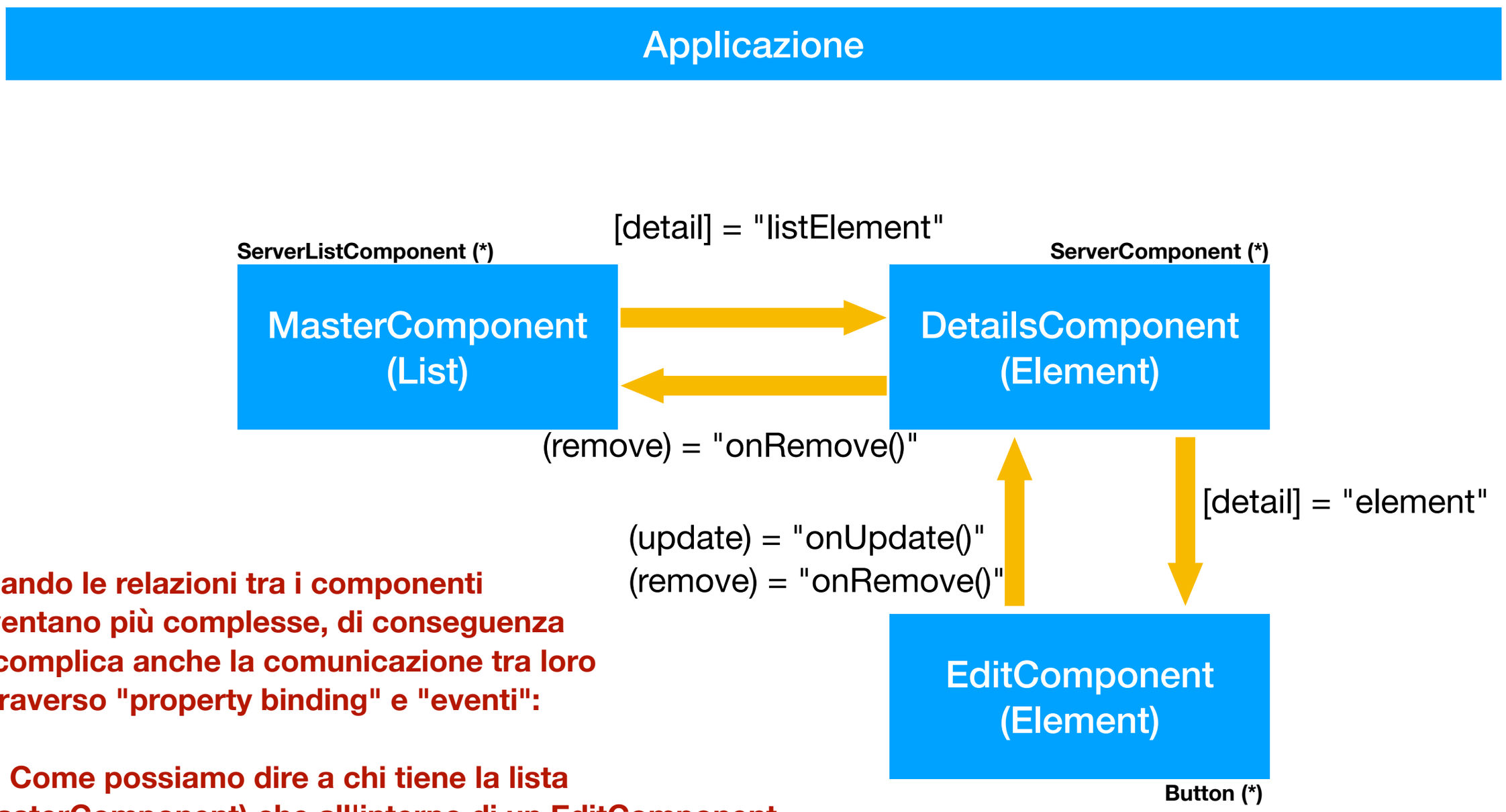
Comunicazione “semplice” tra “pochi” componenti



Quando i componenti sono pochi e tra loro le relazioni sono semplici, la comunicazione tra loro attraverso "property binding" e "eventi" è sufficiente e facile da realizzare

Services & Dependency Injection

Comunicazione tra componenti



Quando le relazioni tra i componenti diventano più complesse, di conseguenza si complica anche la comunicazione tra loro attraverso "property binding" e "eventi":

es: Come possiamo dire a chi tiene la lista (MasterComponent) che all'interno di un EditComponent l'utente ha scelto di eliminare un elemento?

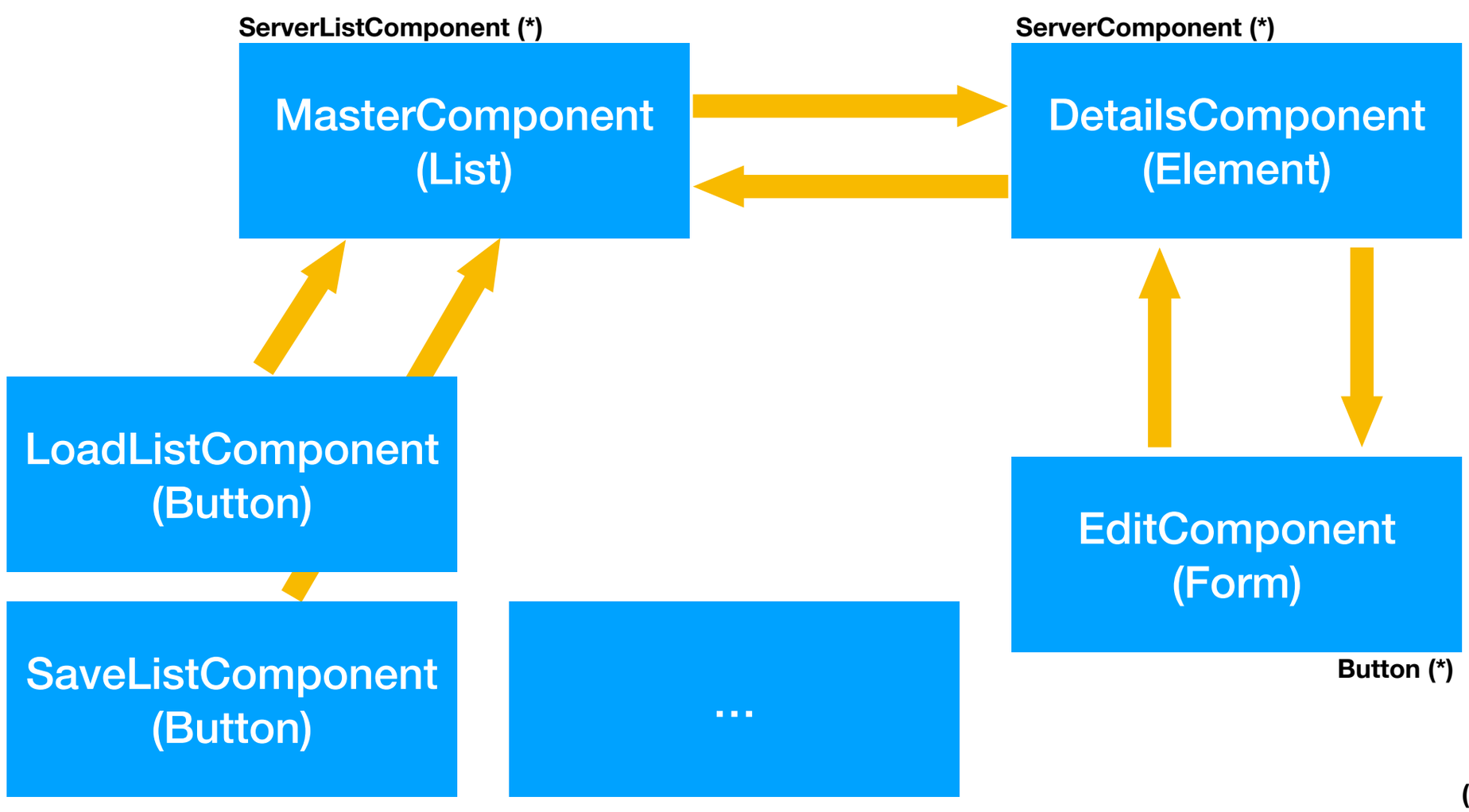
(*) = componenti nell'esempio

Services & Dependency Injection

Comunicazione tra componenti

Applicazione

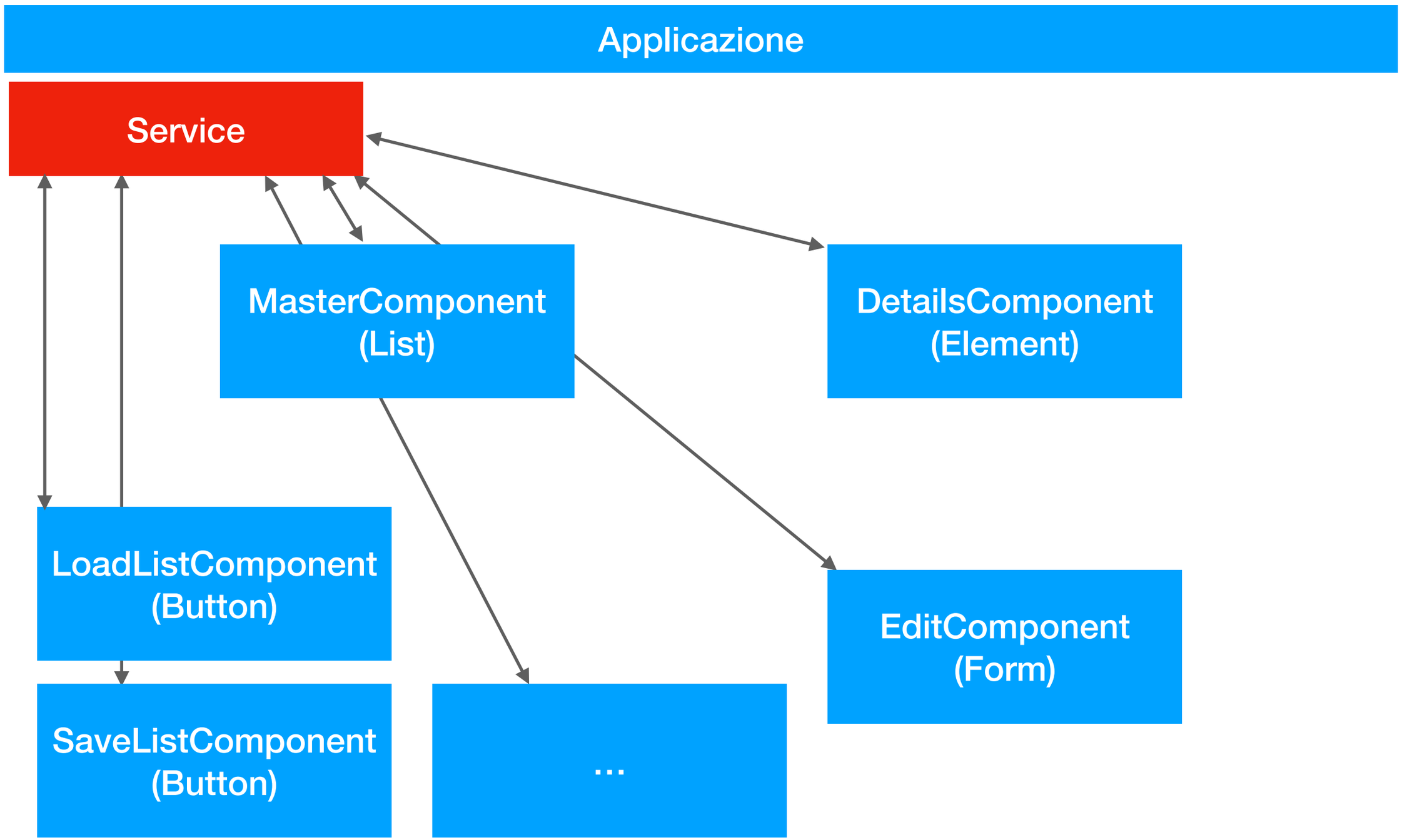
E quando i componenti sono molti di più e più complesse le relazioni?



(*) = componenti nell'esempio

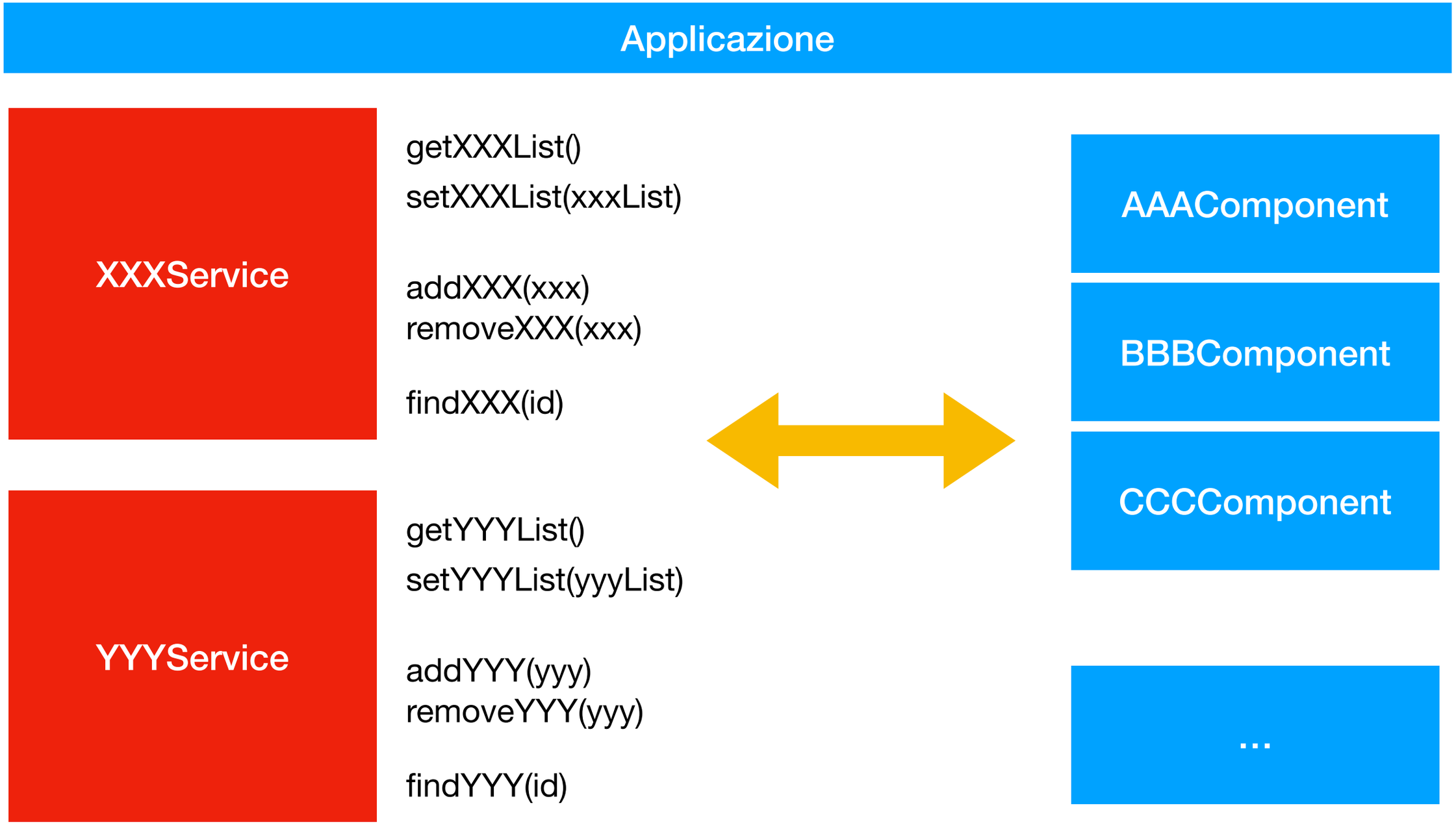
Services & Dependency Injection

Comunicazione “semplificata” tra “molti” componenti



Services & Dependency Injection

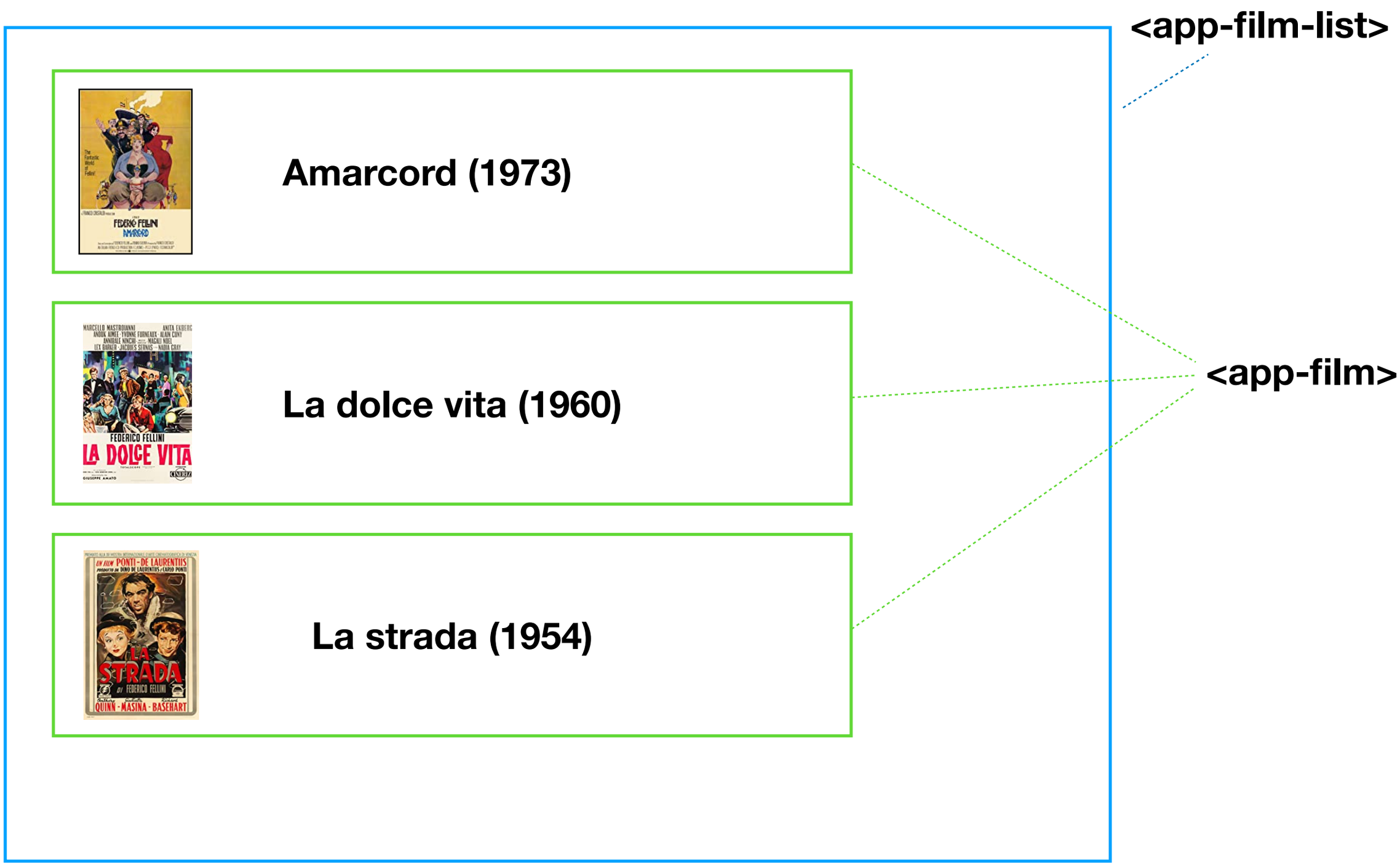
Comunicazione “semplificata” tra “molti” componenti



APP: Specifiche della "Fellini App”

- Obiettivo: sviluppare un'app Angular
  con le seguenti funzionalità:
  - Una pagina con l'elenco dei principali film di Federico Fellini
  - Una pagina di dettaglio (scheda) per ogni film dell'elenco, contenente:
    - immagine locandina
    - nome
    - descrizione (breve trama)
    - regista
    - elenco personaggi ed interpreti principali

APP: Pagina Elenco Film



APP: creazione componenti

Posizionarsi nella cartella contenitore dei progetti e crearne uno nuovo con:

- c:\> **ng new --skip-tests --style css fellini-app**
- c:\> **cd fellini-app**
- c:\fellini-app> **ng generate component film-list --skip-tests**
- c:\fellini-app> **ng generate component film --skip-tests**
- c:\fellini-app> **ng generate service film --skip-tests**

Aprire il nuovo progetto con Visual Studio Code

APP: Modello dati

film.service.ts

```
import { Injectable } from '@angular/core';
import { Film } from './film.model';

@Injectable({
  providedIn: 'root'
})
export class FilmService {
  filmList: Film[] = [];

  constructor() {
    // Amarcord
    let film = new Film();
    film.id = 1;
    film.name = 'Amarcord';
    film.year = 1973;
    film.description = '........';
    film.imageUrl = 'https://m.media-amazon.com/images/M/...jpg';
    film.director = 'Federico Fellini';
    film.cast = [ 'Pupella Maggio', 'Armando Brancia', 'Magali Noël' ];
    this.filmList.push(film);
    //...
  }

  getFilmList(): Promise<Film[]> {
    return Promise.resolve(this.filmList);
  }
}
```

film.model.ts

```
export class Film {
  id: number;
  name: string;
  year: number;
  description: string;
  imageUrl: string;
  director: string;
  cast: string[];
}
```

APP: collegamento al FilmService

film-list/film-list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FilmService } from '../film.service';

@Component({
  selector: 'app-film-list',
  templateUrl: './film-list.component.html',
  styleUrls: ['./film-list.component.css']
})
export class FilmListComponent implements OnInit {

  filmList: Film[] = [];

  constructor(public filmService: FilmService) {
  }

  ngOnInit(): void {
    this.filmService.getFilmList()
      .then(list => {
        this.filmList = list;
      });
  }
}
```

film/film.component.ts

```
import { Component, Input, OnInit } from '@angular/core';
import { FilmService } from '../film.service';
import { Film } from '../film.model';

@Component({
  selector: 'app-film',
  templateUrl: './film.component.html',
  styleUrls: ['./film.component.css']
})
export class FilmComponent implements OnInit {
  @Input() film: Film = null;

  constructor(public filmService: FilmService) {
  }

  ngOnInit(): void {
  }

}
```

APP: prima bozza template

app.component.html

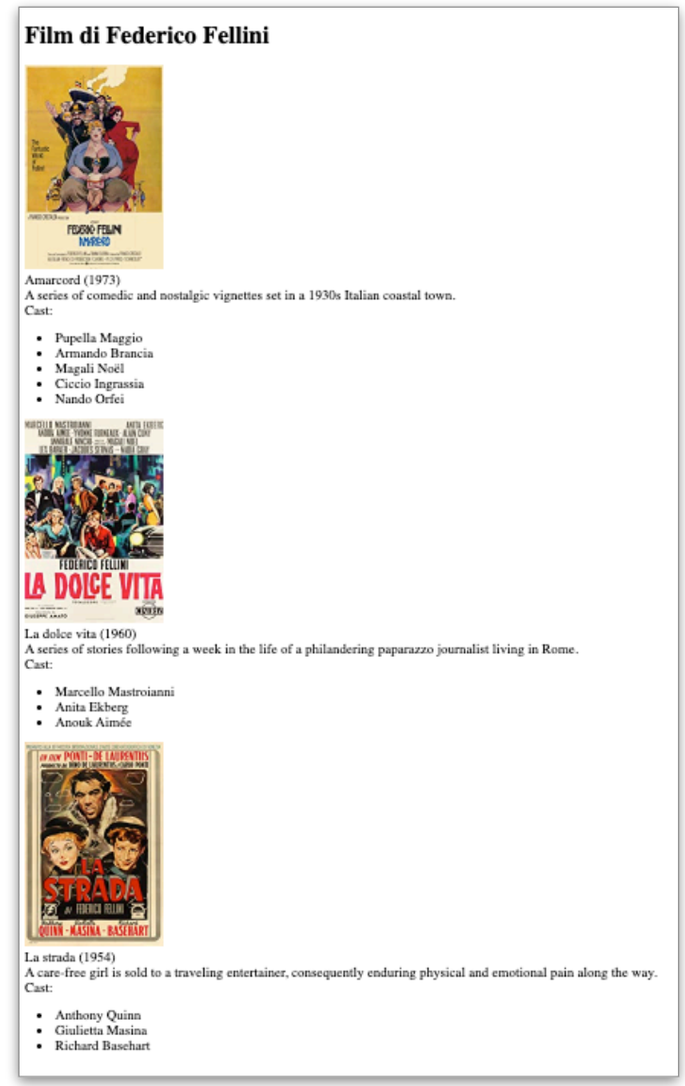
```
<div class="container">
  <h1>Film di Federico Fellini</h1>
  <app-film-list></app-film-list>
</div>
```

film-list/film-list.component.html

```
<app-film *ngFor="let film of filmList"
            [film]="film"
></app-film>
```

film/film.component.html

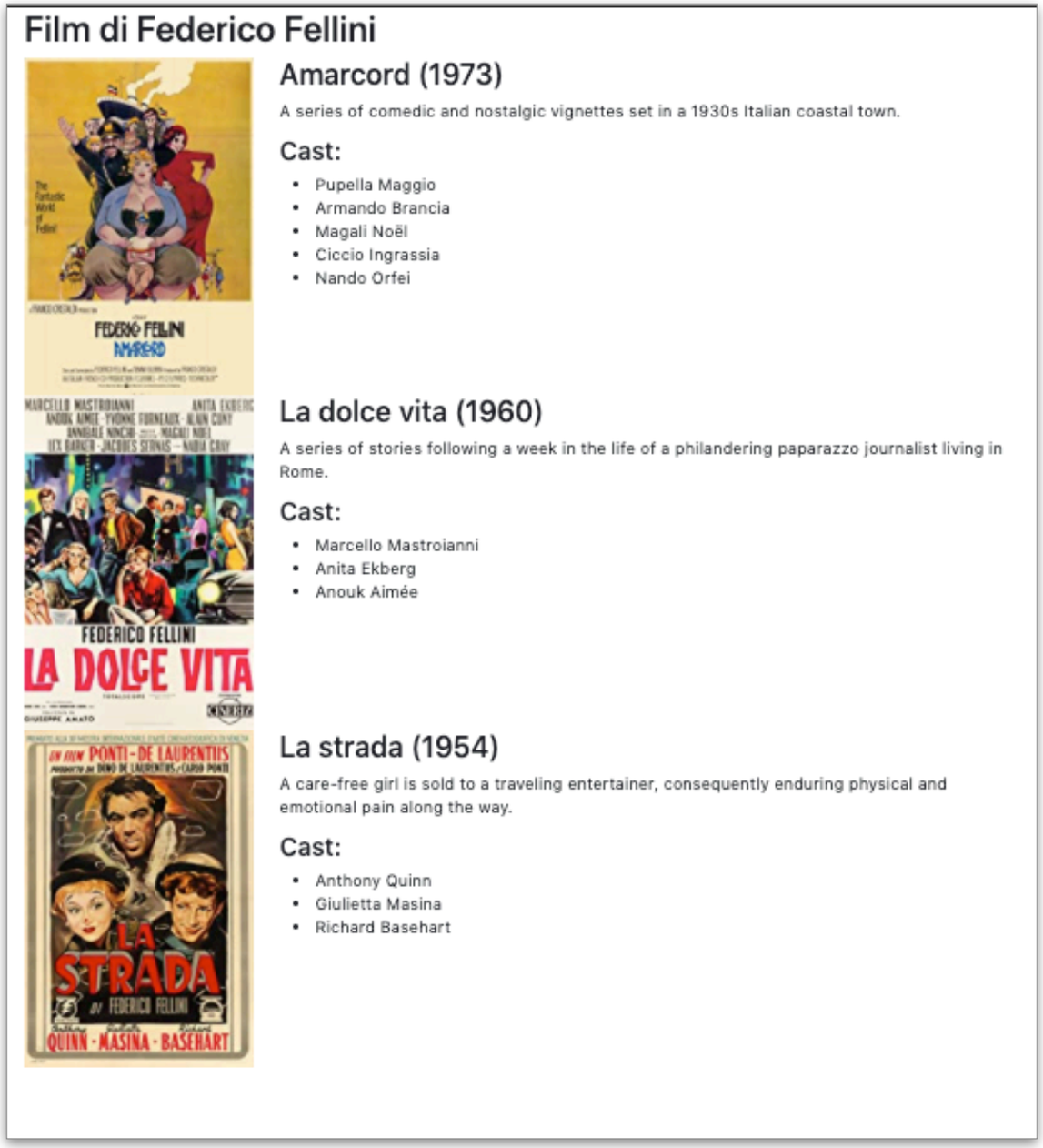
```
<img [src]="film.imageUrl">
<div>
  {{ film.name }} ({{ film.year }})
</div>
<div>
  {{ film.description }}
</div>
<div>
  Cast:
  <ul>
    <li *ngFor="let attore of film.cast">
      {{ attore }}
    </li>
  </ul>
</div>
```



APP: Bootstrap template

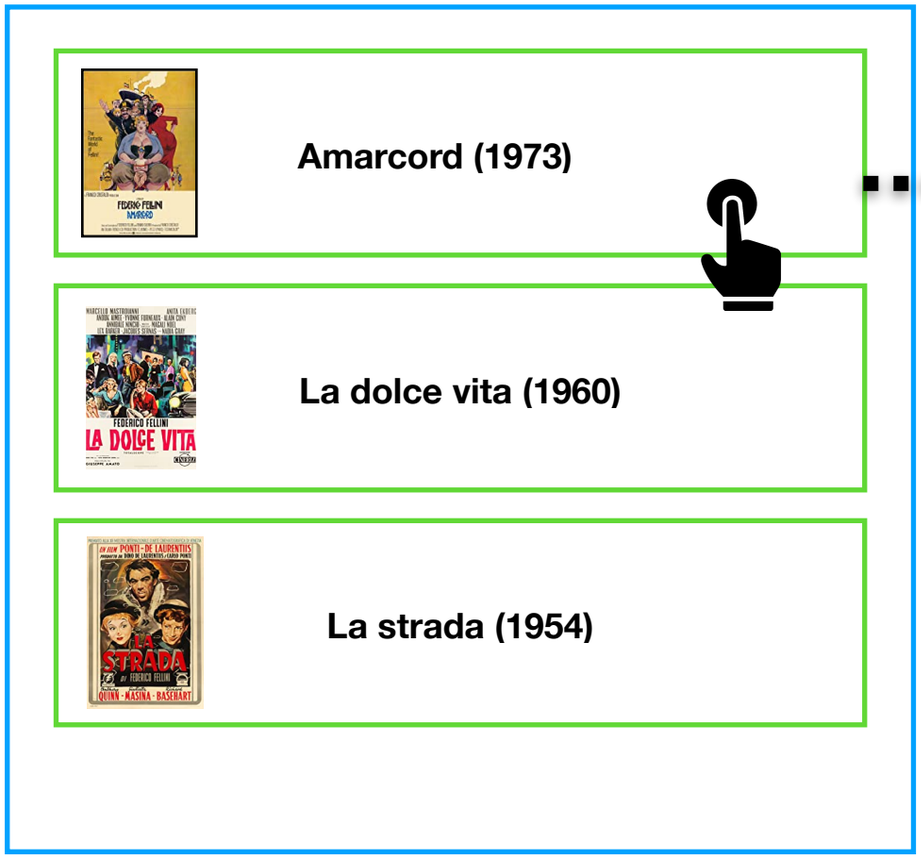
film/film.component.html

```
<div class="row">
  <div class="col-3">
    <img [src]="film.imageUrl" width="100%">
  </div>
  <div class="col">
    <h2>{{ film.name }} ({{ film.year }})</h2>
    <p>{{ film.description }}</p>

    <h3>Cast:</h3>
    <ul>
      <li *ngFor="let attore of film.cast">
        {{ attore }}
      </li>
    </ul>
  </div>
</div>
```



APP: Navigazione tra pagine

Pagina Elenco



Pagina Dettaglio

Angular è un framework per la creazione di applicazioni "Single-Page", pertanto la navigazione tra "Pagine" è in realtà una navigazione tra "Component" contenitori (chiamiamoli "View") che in base alle azioni dell'utente si alternano ad occupare la parte centrale della pagina, cioé del componente più esterno della nostra app: AppComponent (<app-root>).

Per la gestione della navigazione, Angular mette a disposizione dei componenti all'interno del "RoutingModule"

https://angular.io/guide/router
https://www.mrwebmaster.it/javascript/rounting-angular-2-angular-router_12764.html
https://www.html.it/pag/63865/routing-in-angular-2/

APP: Routing / 1

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FelliniApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Abilita URL relativi formato HTML5:
pre HTML5: /#film/1
HTML5: /film/1

dove:
/ : URL di pagina
film/1 : anchor all'interno della pagina

app.module.ts

```
...
import { RouterModule, Routes } from '@angular/router';


const routes: Routes = [
  { path: '', component: FilmListComponent },
  { path: 'film', component: FilmListComponent },
  { path: 'film/:id', component: FilmComponent },

  // otherwise redirect to home
  { path: '**', redirectTo: '' }
];

@NgModule({
  ...
  imports: [
   ...
   RouterModule.forRoot(routes),
  ],
  ...
})
export class AppModule {
   ...
}
```

Gestisce lo scambio dei componenti in base all'URL

app.component.html

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

APP: Routing / 2

film-list/film-list.component.html

```
<div class="row mt-3 mb-3">
  <div class="col">
    <h1>Film di Federico Fellini</h1>
  </div>
</div>

<div class="d-flex align-content-stretch flex-wrap">
    <div class="card shadow m-2 p-1" style="width: 15rem" [routerLink]="'/film/' + film.id"
         *ngFor="let film of filmService.getFilmList()">
      <img [src]="film.imageUrl" class="card-img-top" [alt]="film.name">
      <div class="card-body">
        <h5 class="card-title">{{film.name}}</h5>
        <p class="card-text">{{film.year}}</p>
        <div class="card-footer">
          <a [routerLink]="'/film/' + film.id"
             class="btn btn-primary">Vai alla scheda</a>
        </div>
      </div>
    </div>
</div>
```

film/film.component.html

```
<div class="row mt-3 mb-3">
  <a [routerLink]="'/'" class="btn btn-primary">
    <i class="fas fa-chevron-left"></i>Elenco
  </a>
</div>

<ng-container *ngIf="film != null">
  <div class="row m-2" *ngIf="film != null">
    <div class="col-3">
      <img class="shadow" [src]="film.imageUrl" width="100%">
    </div>
    <div class="col">
      <h2>{{ film.name }} ({{ film.year }})</h2>
      <h4 class="mt-3">Regia:</h4> {{film.director}}

      <h4 class="mt-3">Cast:</h4>
      <ul>
        <li *ngFor="let attore of film.cast">
          {{ attore }}
        </li>
      </ul>
    </div>
  </div>
  <div class="row p-3">
    <p>{{ film.description }}</p>
  </div>
</ng-container>
```

Passando un URL all'attributo routerLink si attiva il Routing

```
<tag [routerLink]="'url/:parametro'">
</tag>
```

APP: Routing / 4

film/film.component.ts

```
import { Component, Input, OnInit } from '@angular/core';
import { FilmService } from '../film.service';
import { Film } from '../film.model';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-film',
  templateUrl: './film.component.html',
  styleUrls: ['./film.component.css']
})
export class FilmComponent implements OnInit {

  @Input() film: Film = null;

  constructor(private filmService: FilmService,
              private activatedRoute: ActivatedRoute) {

  }

  ngOnInit(): void {
    const id = this.activatedRoute.snapshot.paramMap.get('id');
    console.log('FilmComponent.ngOnInit(): film id=', id);
    if (id != null) {
      this.film = this.filmService.getFilm(Number(id));
      console.log('FilmComponent.ngOnInit(): film ', this.film);
    }
  }

}
```

Il routing non inizializza le variabili @Input() con il property binding!

Devo provvedere io all'inizializzazione utilizzando il parametro fornito nell'URL.

Per recuperare il parametro fornito dal routing nell'URL:

- utilizzo il servizio ActivatedRoute
- con il parametro recupero il film dal filmService
- lo assegno all'attributo di classe this.film
- il template ora può essere elaborato

APP: Routing Guard

TODO: descrivere come funziona il Route Guard
esempio di implementazione di login

REST client: HttpClient

- Angular mette a disposizione un "service" per effettuare le chiamate REST verso l'esterno. Per utilizzarlo:
  - Importare il modulo HttpClientModule
  - Utilizzare il service HttpClient

src/app/app.module.ts

```
...
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';


@NgModule({
 ...
 imports: [
   ...
   BrowserModule,
   // importare HttpClientModule dopo BrowserModule.
   HttpClientModule,
  ],
  ...
})
export class AppModule { }
```

src/app/xxxx.service.ts

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


@Injectable()
export class XXXXService {
  constructor(private http: HttpClient) { }
}
```

REST client: API per test

- https://jsonplaceholder.typicode.com/
- https://picsum.photos

- https://rapidapi.com/
- https://github.com/toddmotto/public-apis
- https://swapi.dev
- https://reqres.in
- http://www.icndb.com

REST client: modello dati

- Prima di utilizzare il HttpClient per effettuare una chiamata REST è necessario definire il modello dati del risultato:

https://swapi.dev/api/people/1/

src/app/swapi.model.ts

```
export class People {
  name: string;
  height: string;
  mass: string;
  hair_color: string;
  skin_color: string;
  eye_color: string;
  birth_year: string;
  gender: string;
  homeworld: string;
  films: string[];
  species: string[];
  vehicles?: string[];
  starships: string[];
  created: string;
  edited: string;
  url: string;
}
```

GET


```
{
     "name": "Luke Skywalker",
     "height": "172",
     "mass": "77",
     "hair_color": "blond",
     "skin_color": "fair",
     "eye_color": "blue",
     "birth_year": "19BBY",
     "gender": "male",
     "homeworld": "http://swapi.dev/api/planets/1/",
     "films": [
          "http://swapi.dev/api/films/1/",
          "http://swapi.dev/api/films/2/",
          "http://swapi.dev/api/films/3/",
          "http://swapi.dev/api/films/6/"
     ],
     "species": [],
     "starships": [
          "http://swapi.dev/api/starships/12/",
          "http://swapi.dev/api/starships/22/"
     ],
     "created": "2014-12-09T13:50:51.644000Z",
     "edited": "2014-12-20T21:17:56.891000Z",
     "url": "http://swapi.dev/api/people/1/"
}
```

REST client: utilizzo

- Si utilizza HttpClient per effettuare una chiamata REST:

```
getPeople(url: string): Promise<People> {
    return this.http.get<People>(url).toPromise();
}
```

```
http: variabile servizio
get: variabile servizio
People: dato da recuperare (e ritornare)
url: indirizzo da chiamare
toPromise(): da Observable<People> a Promise<People>
```

swapi.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { People } from './swapi.model';

@Injectable({
  providedIn: 'root'
})
export class SwapiService {
  baseUrl = 'https://swapi.dev/api';

  constructor(private http: HttpClient) {
  }

  getPeople(url: string): Promise<People> {
    return this.http.get<People>(url)
      .toPromise();
  }

}
```

-

Test Applicazione

- https://www.browserstack.com/
- https://crossbrowsertesting.com

- http://browsershots.org
- https://www.browserling.com

- 108 byte CSS Layout Debugger:

```
[].forEach.call($$("*"),function(a){a.style.outline="1px solid #"+(~~(Math.random()*(1<<24))).toString(16)})
```

Angular UI Components

- Angular Material
  https://material.angular.io
- PrimeNG
  https://www.primefaces.org/primeng/
- Onsen UI for Angular 2+
  https://onsen.io/angular2/
- Ant Design of Angular (ng-zorro)
  https://ng.ant.design/docs/introduce/en
- NG Bootstrap
  https://ng-bootstrap.github.io
- Kendo UI
  https://www.telerik.com/kendo-ui#angular

Nuova app: “recipe-app”

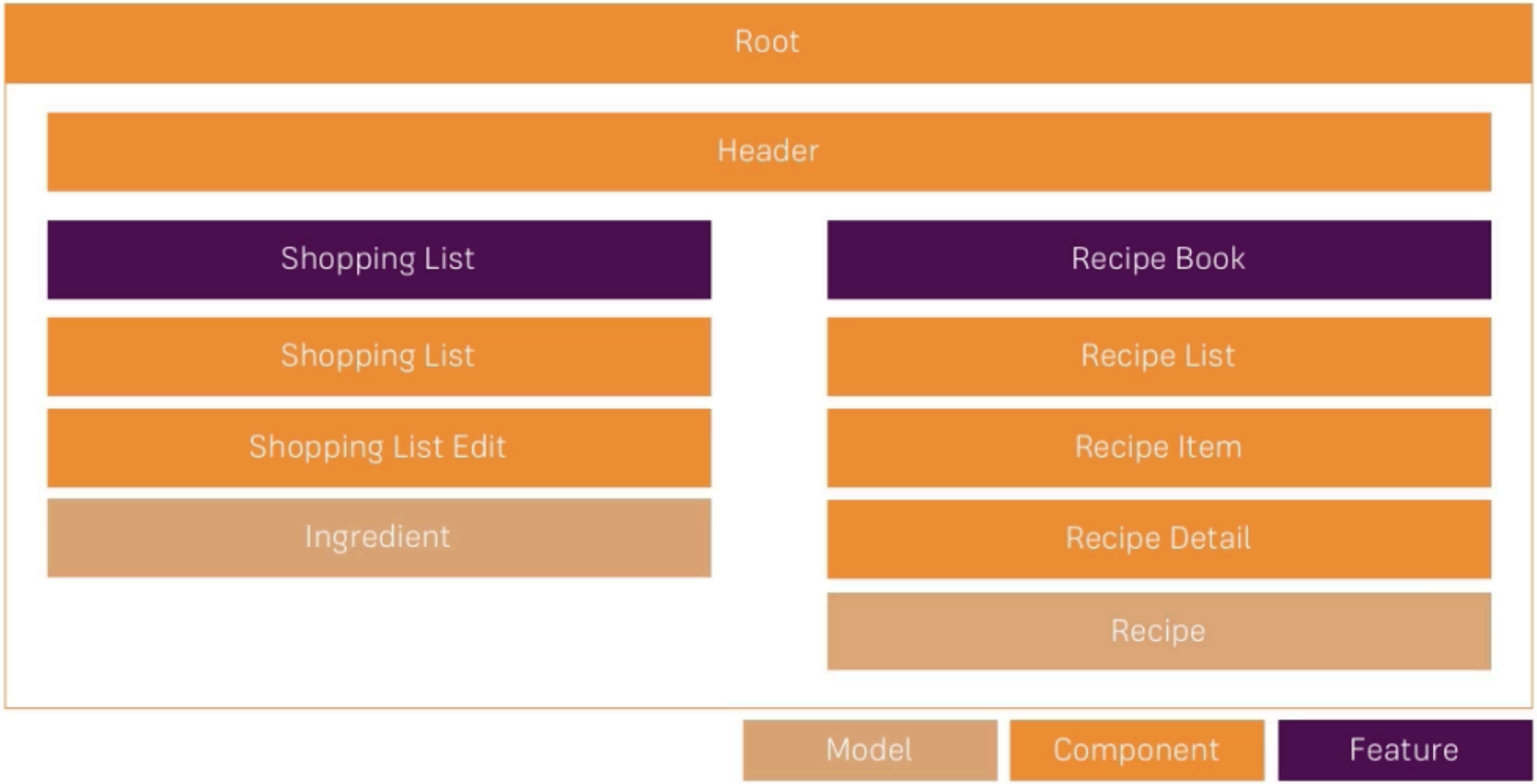
app.component.html

```
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <h2>OK, funziona!</h2>
    </div>
  </div>
</div>
```

OK, funziona!

- c:\recipe-app> **npm start**

Planning the App



recipe-app: Header

header.component.ts

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

@Component({
    selector: 'app-header',
    templateUrl: './header.component.html'
})
export class HeaderComponent {

}
```

header.component.html

```
<h1>Header</h1>
```

app.component.html

```
<app-header></app-header>
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <h2>OK, funziona!</h2>
    </div>
  </div>
</div>
```

app.module.ts

```
...
import { AppComponent } from './app.component';
import { HeaderComponent } from ‘./header/header.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent
  ],
...
})
export class AppModule { }
```

recipe-app: Component(s)

- c:\recipe-app> **ng g c recipes --spec false**
- c:\recipe-app> **ng g c recipes/recipe-list --spec false**
- c:\recipe-app> **ng g c recipes/recipe-detail --spec false**
- c:\recipe-app> **ng g c recipes/recipe-item --spec false**

- c:\recipe-app> **ng g c shopping-list --spec false**
- c:\recipe-app> **ng g c shopping-list/shopping-edit --spec false**

recipe-app: connect components

recipe.component.html

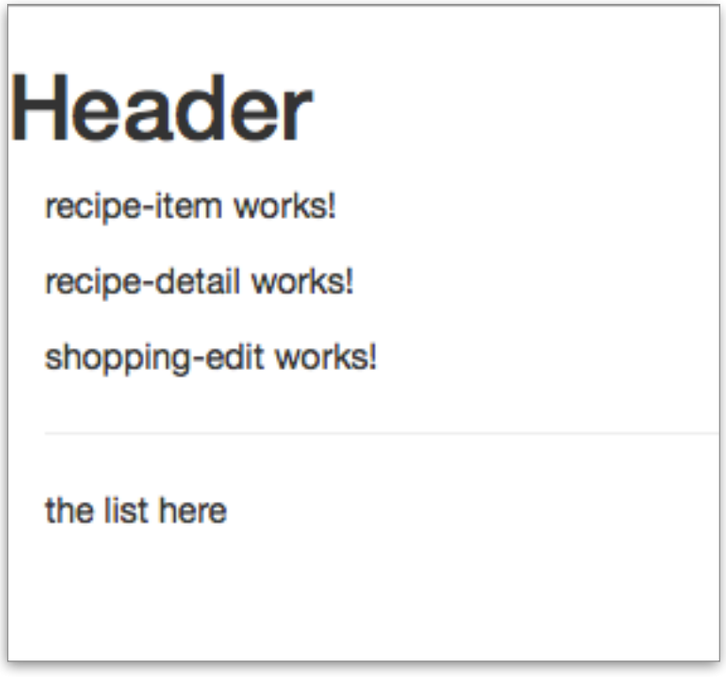
```
<div class="row">
  <div class="col-md-5">
    <app-recipe-list></app-recipe-list>
  </div>
  <div class="col-md-7">
    <app-recipe-detail></app-recipe-detail>
  </div>
</div>
```

recipe-list.component.html

```
<app-recipe-item></app-recipe-item>
```

shopping-list.component.html

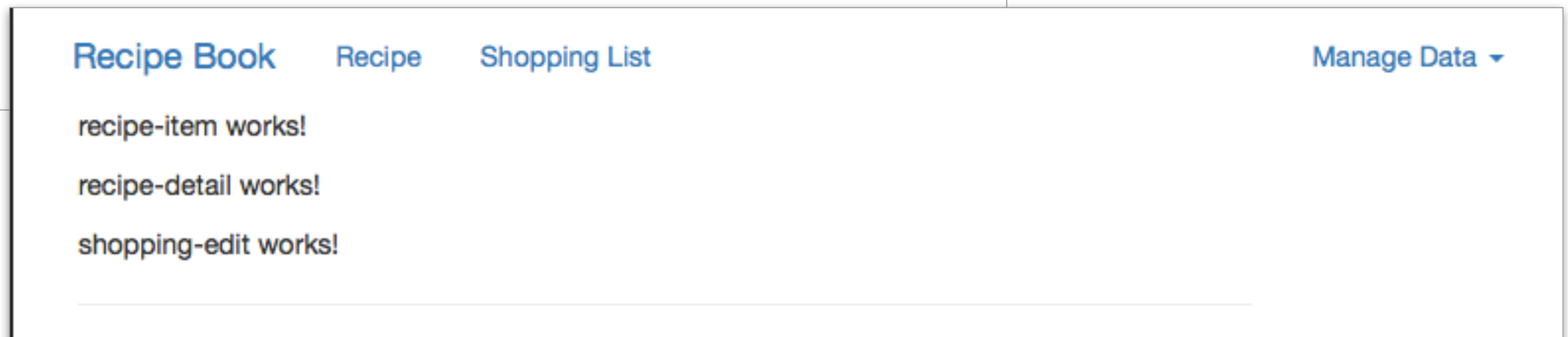
```
<div class="row">
  <div class="col-xs-10">
    <app-shopping-edit></app-shopping-edit>
    <hr>
    <p>the list here</p>
  </div>
</div>
```



recipe-app: navigation bar

header.component.html

```
<nav navbar navbar-default>
    <div class="container-fluid">
        <div class="navbar-header">
            <a href="#" class="navbar-brand">Recipe Book</a>
        </div>
        <div class="collapse navbar-collapse">
            <ul class="nav navbar-nav">
                <li><a href="#">Recipe</a></li>
                <li><a href="#">Shopping List</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
                <li class="dropdown">
                    <a class="dropdown-toggle" role="button" href="#">
                        Manage Data <span class="caret"></span>
                    </a>
                    <ul class="dropdown-menu">
                        <li><a href="#">Load Data</a></li>
                        <li><a href="#">Save Data</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </div>
</nav>
```



recipe-app: recipe-list

recipe-list.component.html

```
<div class="row">
    <div class="col-xs-12">
        <button class="btn btn-success">New Recipe</button>
    </div>
</div>
<hr>
<div class="row">
    <div class="col-xs-12">
        <a href="#" class="list-group-item clearfix" *ngFor="let recipe of recipeList">
            <div class="pull-left">
                <h4 class="list-group-item-heading">{{ recipe.name }}</h4>
                <p class="list-group-item-text">{{ recipe.description }}</p>
            </div>
            <span class="pull-right">
                <img [src]="recipe.imageUrl"
                     alt="{{ recipe.name }}"
                     class="img-responsive" style="max-height: 50px;">
            </span>
        </a>
        <app-recipe-item></app-recipe-item>
    </div>
</div>
```

recipe-app: recipe-list

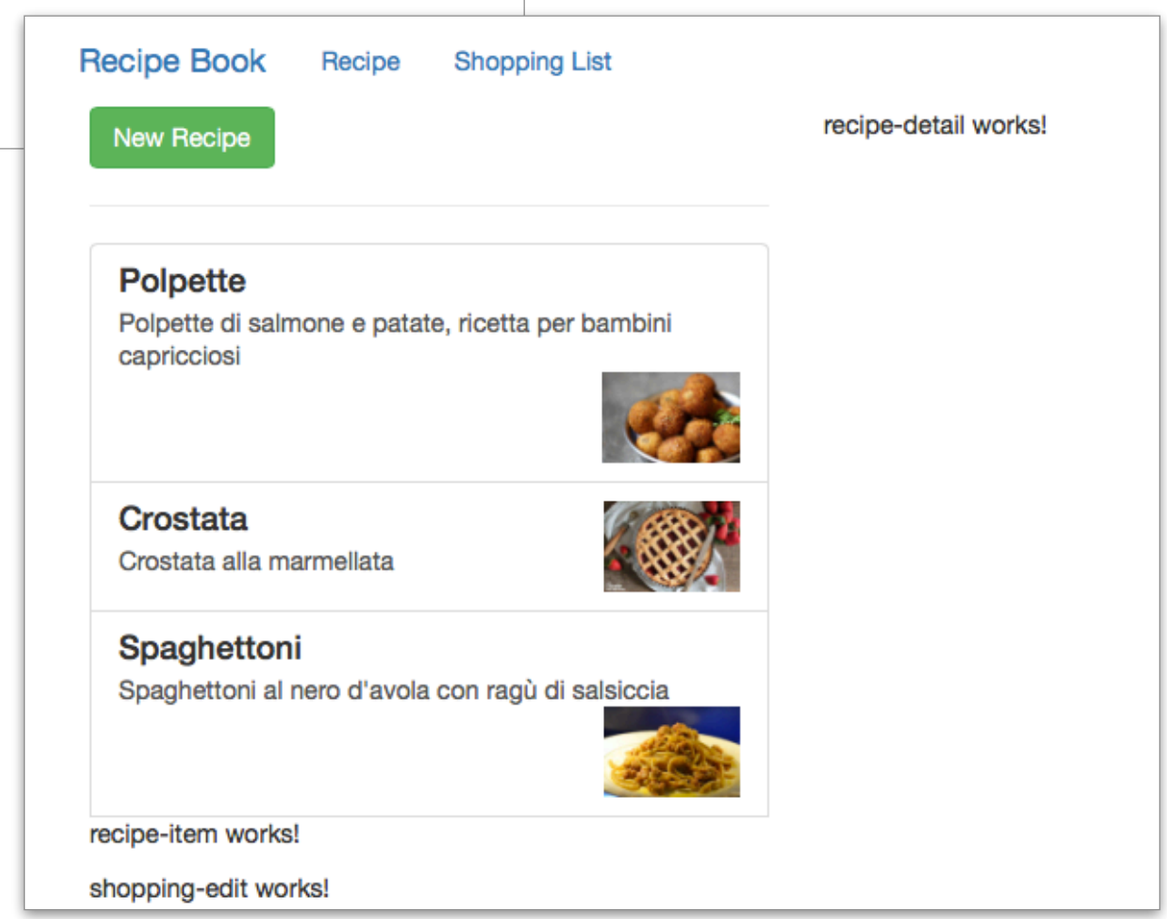
recipe-list.component.ts

```
import { Component } from '@angular/core';
import { Recipe } from '../recipe.model';

@Component({
  selector: 'app-recipe-list',
  templateUrl: './recipe-list.component.html',
  styleUrls: ['./recipe-list.component.css']
})
export class RecipeListComponent {
  recipeList: Recipe[] = [
    new Recipe('Polpette',
               'Polpette di salmone e patate, ricetta per bambini capricciosi',
               'http://ricette.donnaclick.it/images/2013/10/Polpette-di-salmone-e-patate.jpg'),

    new Recipe('Crostata',
               'Crostata alla marmellata',
               'https://www.ricettedellanonna.net/wp-content/uploads/2016/05/Ricetta-crostata-alla-
marmellata-620x414.jpg'),

    new Recipe('Spaghettoni',
               'Spaghettoni al nero d\'avola con ragù di salsiccia',
               'http://www.nonnagilda.it/wp-content/uploads/2012/06/SPAGHETTONI-AL-NERO-DAVOLA-CON-RAGU-
DI-SALSICCIA-LE-RICETTE-DI-NONNA-GILDA.jpg'),
  ];

  constructor() { }

}
```

recipe-app: recipe-list

recipe.model.ts

```
export class Recipe {
    public name: string;
    public description: string;
    public imageUrl: string;

    constructor(name: string, description: string, imageUrl: string) {
        this.name = name;
        this.description = description;
        this.imageUrl = imageUrl;
    }
}
```



recipe-app: recipe-detail

recipe-detail.component.html

```
<div class="row">
  <div class="col-xs-12">
    <img src="" alt="" class="img-responsive">
  </div>
</div>
<div class="row">
  <div class="col-xs-12">
    <h1>Recipe Name</h1>
  </div>
</div>
<div class="row">
  <div class="col-xs-12">
    <div class="btn-group">
      <button type="button" class="btn btn-primary dropdown-toggle">
        Manage Recipe <span class="caret"></span>
      </button>
      <ul class="dropdown-menu">
        <li><a href="#">To Shopping List</a></li>
        <li><a href="#">Edit Recipe</a></li>
        <li><a href="#">Delete Recipe</a></li>
      </ul>
    </div>
  </div>
</div>
<div class="row">
    <div class="col-xs-12">
      Recipe Description
    </div>
</div>
<div class="row">
    <div class="col-xs-12">
      Ingredients
    </div>
</div>
```

recipe-app: shopping-list

shopping-list.component.ts

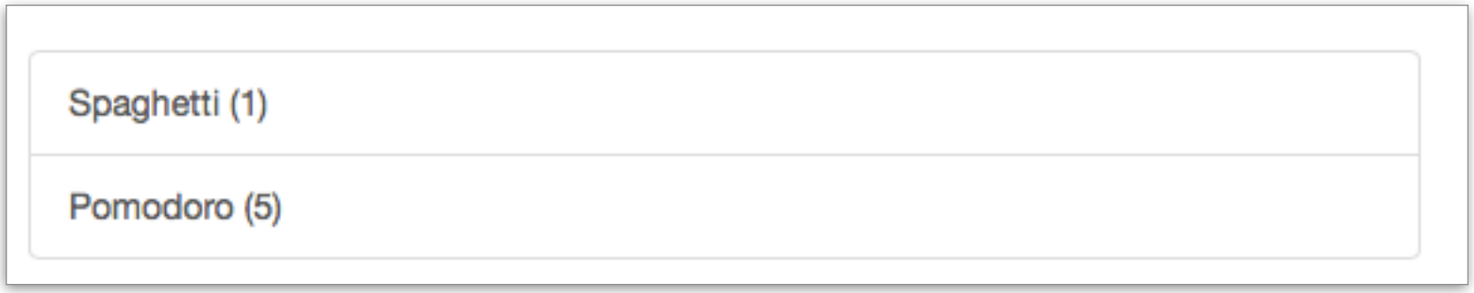
```
import { Component, OnInit } from '@angular/core';

import { Ingredient } from './../shared/ingredient.model';

@Component({
  selector: 'app-shopping-list',
  templateUrl: './shopping-list.component.html',
  styleUrls: ['./shopping-list.component.css']
})
export class ShoppingListComponent implements OnInit {
  ingredientList: Ingredient[] = [
    new Ingredient('Spaghetti', 1),
    new Ingredient('Pomodoro', 5),
  ];

  constructor() { }

  ngOnInit() {
  }

}
```

shopping-list.component.html

```
<div class="row">
  <div class="col-xs-10">
    <app-shopping-edit></app-shopping-edit>
    <hr>
    <ul class="list-group">
      <a class="list-group-item"
         style="cursor: pointer"
         *ngFor="let ingredient of ingredientList"
      >
        {{ ingredient.name }} ({{ingredient.quantity}})
      </a>
    </ul>
  </div>
</div>
```

ingredient.model.ts

```
export class Ingredient {
    constructor(
        public name: string,
        public quantity: number
    ) {}
}
```



recipe-app: shopping-edit

shopping-edit.component.html

```
<div class="row">
  <div class="col-xs-12">
    <form>
        <div class="row">
          <div class="col-sm-5 form-group">
            <label for="name">Name</label>
            <input type="text" id="name" class="form-control">
          </div>
          <div class="col-sm-2 form-group">
              <label for="quantity">Quantity</label>
              <input type="number" id="quantity" class="form-control">
          </div>
        </div>
        <div class="row">
          <div class="col-xs-12">
            <button type="submit" class="btn btn-success">Add</button>
            <button type="button" class="btn btn-danger">Delete</button>
            <button type="button" class="btn btn-primary">Clear</button>
          </div>
        </div>
      </form>
  </div>
</div>
```