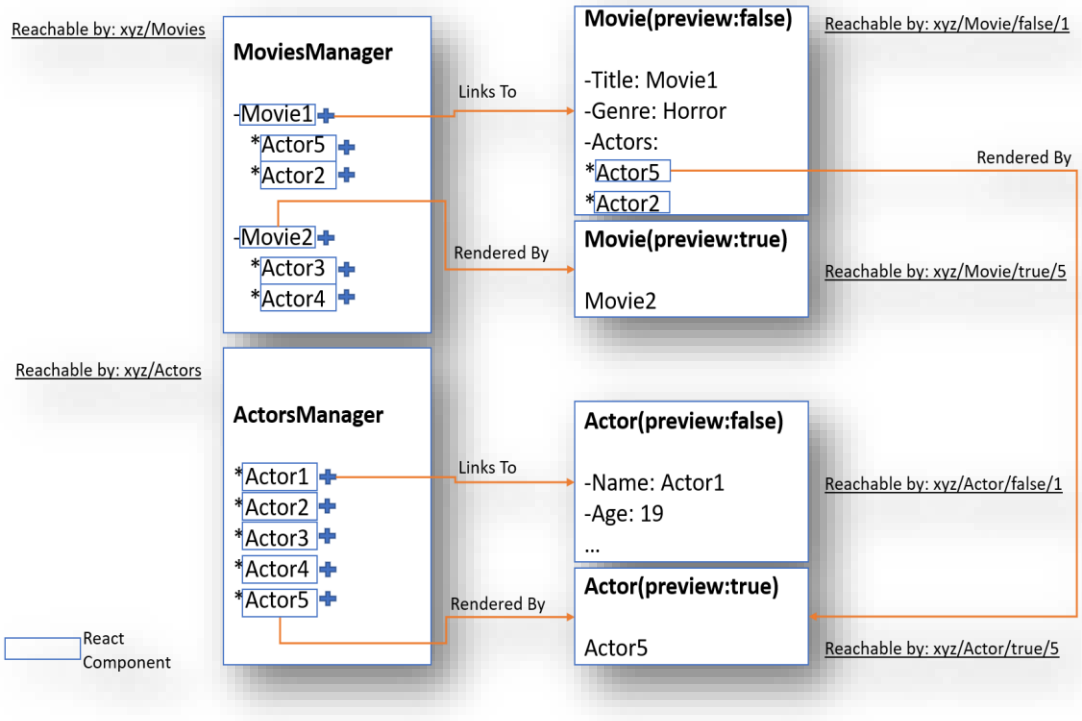
The focus of this practical exercise is to let you experience with SPA (Single Page Application) with React. More specifically you will build a series of react components to visualize the entities of our DB: Movies and Actors. Remember, React components are run on the browser, not on the server (see Razor/template engines).

STEP 0

- Make sure you have Node installed on your computer. If not use for example https://chocolatey.org/ and run `choco install nodejs` (windows only)
- Run the command: `new react project`
- Repeat the steps of your practice 3 in this new project (do not forget to run *migration + update*). Moreover, make sure to inject the database context in the Startup file.
- Since the focus of this practice is letting you work with react, remove the generated folders Movie and Actor in Views

STEP 1 – Implementation description

To start you need to know how many components are needed for your application. In general, every component is mapped to an entity of our database, but not only. In our example we have a *collection* of movies and a *collection* of actors. (total **2 components**). The goal of these 2 *collection* components is to render in group each movie and each actor (again **2 components**). In total we will need at least 4 components: 2 to present the grouped items and 2 to present them in isolation. Below see a schema representing the component structure that we have just mentioned. In this schema, we also emphasize the transitions (***Links To***) between components. An advantage of using components is that we can group the logic of an entity in a single place and reuse it (***Rendered By***) on-demand by passing some attributes to change some visualization behavior (***preview=true|false***). Moreover, components can also be called/instantiated directly, thus without nesting them inside other components. This is done by referencing a path (***Referenced By***). **Note:** in this picture the interaction with a controller is missing. This is discussed later in this document.



STEP 2 – check list

For this specific implementation you need to understand and apply the following concepts:

- Models
- Components
- Routing/Navigation Link
- Life cycle
- Promises/Async calls

STEP 2.1 Models

Each entity returned by your controller (in our case Movie/ Actor) must be mapped to an equivalent one in the front end. Create a Models.tsx file in the ClientApp folder and add to it a definition in typescript for Movie and Actor.

```
export type Movie = {
  id:number,
  title:string,
  genre:string,
  actors:Actor[]
}

export type Actor = {
  id:number,
  name:string,
  surname:string
}
```

STEP 2.2 Components

Add to your ClientApp/components folder 4 *tsx* files and call them MoviesManager, Movie, ActorsComponent, and Actor. And add a component to each of these files with the following structure:

- MoviesManager

```
import * as React from 'react';
import { RouteComponentProps } from 'react-router';
import { Route, NavLink, Link } from 'react-router-dom'
import * as Models from "../Model"
import {Actor} from "./Actor"


async function loadMovies():Promise<Models.Movie[]>{
    let res = await fetch(`./Movies/GetAll`, { method: 'get',  credentials: 'include', headers: { 'content-type': 'application/json' } })
    let res1 = await res.json()
    return res1
}

export class MoviesManager extends React.Component<RouteComponentProps<{}>, {movies:Models.Movie[] | "loading"}> {
    constructor(props:RouteComponentProps<{}>){
        super(props)
        this.state = {movies:"loading"}
    }
    componentWillMount(){
        loadMovies().then(ms => this.setState({...this.state, movies : ms}))
    }
    public render() {
        if(this.state.movies == "loading") return <div>Loading...</div>
        return <div className="movies">
                    {this.state.movies.map(m => <div className="movies-movie">
                        {m.title}
                        <Link to={"/movie/false/" + m.id}>
                            <button>Expand movie</button>
                        </Link>
                        <div className="movies-movie-actors">
                            {m.actors.map(a => <div className="movies-movie-actors-actor">
                                <Actor actor={a} preview={true}/>
                                <Link to={"/actor/false/" + a.id}>
                                    <button>Expand actor</button>
                                </Link>
                            </div>)}
                        </div>
                    </div>)}
                </div>
    }
}
```

- Movie

```
import * as React from 'react';
import { RouteComponentProps } from 'react-router';
import { Route, NavLink, Link } from 'react-router-dom'
import * as Models from "../Model"
import {Actor} from "./Actor"


async function loadMovie(id:number):Promise<Models.Movie>{
    let res = await fetch(`./Movies/GetMovie/${id}`, { method: 'get',  credentials: 'include', headers: { 'content-type': 'application/json' } })
    let res1 = await res.json()
    return res1
}
export class MovieRoute extends React.Component<RouteComponentProps<{preview:any, movie:any}>, {movie:Models.Movie | "loading"}> {
    constructor(props:RouteComponentProps<{preview:boolean, movie:number}>){
        super(props)
        this.state = {movie:"loading"}
    }
    componentWillMount(){
        loadMovie(this.props.match.params.movie).then(m => this.setState({...this.state, movie : m}))
    }

    public render() {
        if(this.state.movie == "loading") return <div>Loading...</div>
        return <Movie preview={this.props.match.params.preview == 'true'} movie={this.state.movie}/>
    }
}

export class Movie extends React.Component<{preview:boolean, movie:Models.Movie}, {}> {
    constructor(props:{preview:boolean, movie:Models.Movie}){
        super(props)
        this.state = {}
    }

    public render() {
        if(this.props.preview){
            return <div> {this.props.movie.title} </div>
        }
        else{
            return <div>
                <div> Movie title: {this.props.movie.title} </div>
                <div> Movie genre: {this.props.movie.genre} </div>
                <div className="movie-actors">
                    {this.props.movie.actors.map(a => <div className="movie-actors-actor">
                        <Actor actor={a} preview={true}/>
                        <Link to={"/actor/false/" + a.id}>
                            <button>Expand actor</button>
                        </Link>
                    </div>)}
                </div>
            </div>
        }
    }
}
```

- ActorsManager

```
import * as React from 'react';
import { RouteComponentProps } from 'react-router';
import { Route, NavLink, Link } from 'react-router-dom'
import * as Models from "../Model"
import {Actor} from "./Actor"


async function loadActors():Promise<Models.Actor[]>{
    let res = await fetch(`./Actors/GetAll`, { method: 'get',  credentials: 'include', headers: { 'content-type': 'application/json' } })
    let res1 = await res.json()
    return res1
}

export class ActorsManager extends React.Component<RouteComponentProps<{}>, {actors:Models.Actor[] | "loading"}> {
    constructor(props:RouteComponentProps<{}>){
        super(props)
        this.state = {actors:"loading"}
    }
    componentWillMount(){
        loadActors().then(as => this.setState({ ...this.state, actors : as}))
    }
    public render() {
        if(this.state.actors == "loading") return <div>Loading ...</div>
        return <div className="actors">
                    {this.state.actors.map(a => <div className="actors-actor">
                            {<Actor actor={a} preview={true}/>}
                            <Link to={"/actor/false/" + a.id}>
                                    <button>Expand actor</button>
                            </Link>
                        </div>)}
                </div>
    }
}
```

- Actor

```
import * as React from 'react';
import { RouteComponentProps } from 'react-router';
import { Route, NavLink } from 'react-router-dom'
import * as Models from "../Model"


async function loadActor(id:number):Promise<Models.Actor>{
    let res = await fetch(`Actors/GetActor/${id}`, { method: 'get',  credentials: 'include', headers: { 'content-type': 'application/json' } })
    let res1= await res.json()
    return res1
}

export class ActorRoute extends React.Component<RouteComponentProps<{preview:any, actor:any}>, {actor:Models.Actor | "loading"}> {
    constructor(props:RouteComponentProps<{preview:boolean, actor:number}>){
        super(props)
        console.log("props", props)
        this.state = {actor:"loading"}
    }
    componentWillMount(){
        loadActor(this.props.match.params.actor).then(a => this.setState({ ...this.state, actor : a}))
    }

    public render() {
        if(this.state.actor == "loading") return <div>Loading ...</div>
        return <Actor preview={this.props.match.params.preview == 'true'} actor={this.state.actor}/>
    }
}

export class Actor extends React.Component<{preview:boolean, actor:Models.Actor}, {}> {
    constructor(props:{preview:boolean, actor:Models.Actor}){
        super(props)
        console.log("props1", props)
        this.state = {}
    }

    public render() {
        if(this.props.preview){
            return <div> {this.props.actor.name}</div>
        }
        else{
            return <div>
                <div> Actor name: {this.props.actor.name} </div>
            </div>
        }
    }
}
```

STEP 2.3 Routing/Navigation Link

React Router is a collection of navigational components that compose declaratively with your application. Whether you want to have bookmarkable URLs for your web app or a composable way to navigate in React Native (https://reacttraining.com/react-router/).

- We use routers to determine the page to visualize on the browser at a time. See code below to understand how routes are manipulated in our project (routes.tsx) to address each of our components.

```
import * as React from 'react';
import { Route } from 'react-router-dom';
import { Layout } from './components/Layout';
import { Home } from './components/Home';
import { FetchData } from './components/FetchData';
import { Counter } from './components/Counter';
import { MoviesManager } from './components/MoviesManager';
import { ActorsManager } from './components/ActorsManager';
import { ActorRoute } from './components/Actor';
import { MovieRoute } from './components/Movie';

//export let path = "http://localhost:5000"
export const routes = <Layout>
    <Route exact path='/' component={ Home } />
    <Route path='/counter/:count?' component={ Counter } />
    <Route path='/fetchdata' component={ FetchData } />
    <Route path='/moviesmanager' component={ MoviesManager } />
    <Route path='/actorsmanager' component={ ActorsManager } />
    <Route path='/actor/:preview/:actor' component={ ActorRoute } />
    <Route path='/movie/:preview/:movie' component={ MovieRoute } />
</Layout>;
```

-We also use navigation links in the navigation menu (NavMenu.tex) to make our MoviesManager and ActorsManager available from the *quick* menu on the left of the page.

```
<li>
    <NavLink to={ '/moviesmanager' } activeClassName='active'>
        <span className='glyphicon glyphicon-th-list'></span> Movies
    </NavLink>
</li>
<li>
    <NavLink to={ '/actorsmanager' } activeClassName='active'>
        <span className='glyphicon glyphicon-th-list'></span> Actors
    </NavLink>
</li>
```

STEP 2.4 Life cycle

See componentWillMount in each component

[ADD MORE]

STEP 2.5 Promise/fetch (api calls)

See fetch in loadActor/loadActors and loadMovie/loadMovies. Each fetch makes a call to a specific controller in the back-end. Check STEP 3 to see how controllers are implemented.

[ADD MORE]

STEP 3 Controller

-Movie controller

```
namespace reac_mvc_movie_app.Controllers
{
    [Route("[controller]")]
    public class MoviesController : Controller
    {
        private readonly MovieContext _context;

        public MoviesController(MovieContext context)
        {
            _context = context;
        }
        [HttpGet("GetAll")]
        public Movie[] GetAll(){
            var movies = from m in _context.Movies
                         let actors = _context.Actors.Where(a => a.MovieId == m.Id)
                         select new Movie(){Id=m.Id, Actors = actors.ToList(), Title = m.Title, Genre = m.Genre};
            return movies.ToArray();
        }
        [HttpGet("GetMovie/{id}")]
        public IActionResult Get(int id){
            var movies = from m in _context.Movies
                         where m.Id == id
                         let actors = _context.Actors.Where(a => a.MovieId == m.Id)
                         select new Movie(){Id=m.Id, Actors = actors.ToList(), Title = m.Title, Genre = m.Genre};
            var movie = movies.FirstOrDefault();
            if(movie == null) return NotFound();
            return Ok(movie);
        }
    }
}
```

-Actor controller

```
namespace reac_mvc_movie_app.Controllers
{
    [Route("[controller]")]
    public class ActorsController : Controller
    {
        private readonly MovieContext _context;

        public ActorsController(MovieContext context)
        {
            _context = context;
        }
        [HttpGet("GetAll")]
        public Actor[] GetAll(){
            return _context.Actors.ToArray();
        }

        [HttpGet("GetActor/{id}")]
        public IActionResult GetActor(int id){
            var actors = from a in _context.Actors
                         where a.Id == id
                         select a;
            var actor = actors.FirstOrDefault();
            if(actor == null) return NotFound();
            return Ok(actor);
        }
    }
}
```