# WebApiExample

This is a small solution to show how to connect a react stand-alone client project to a .Net Core Web Api. I’m using react, but any client project can make a web requisition.

\*\*Client\*\*<br />  
1. Technologies: <br />

React version 16.13 with hooks / Typescript 3.7.2 / SASS to preprocess the CSS <br />  
2. What this project does? <br />

The employees page displays a table with a list of employees from the server, you can filter the   
table results by any field in the table header.

\*\*Back-End\*\*<br />  
1. Technologies: <br />

C# Web Api with .Net Core 3.102<br />

2. What this project does? <br />

The External project receive all web requisitions and it communicates with the Persistence   
project that connect with the repository.

## Executing the project step-by-step:  
  
\*\*Installing all you need<br />\*\*

- I’m using Visual Studio 2019 community to work with server side (External and Persistence projects): <br />

<https://visualstudio.microsoft.com/pt-br/vs/?rr=https%3A%2F%2Fwww.google.com%2F><br />  
  
- You can download version 3.102 of .Net core here: <br />

<https://dotnet.microsoft.com/download/dotnet-core/3.0><br />  
  
- And Visual Studio Code for the client (Webapp project), if you like: <br />  
<https://code.visualstudio.com/Download><br />  
  
- You’ll also need Node.js and npm to execute react and to download all packages with npm instruction: <br />

<https://nodejs.org/en/><br />

\*\*Executing server side project<br />\*\*

- Let’s execute the project WebApiExample.External. Click in the name of the project with the right button and select “set as a startup project”. Now click in the play button in the top menu.

Once the service is up (see the prompt window that will open), you can access the employees

GET api through the url: <https://localhost:5001/api/employees/get>  
  
- This will return a list of employees in json format.

\*\*Executing client project<br />\*\*

- Before starting the react project, you’ll need to download the project packages. If you are using Visual Studio Code, please confirm that you have packages-lock.json in the WebApiExample.App folder, if you have, just type the command “npm ci” in the terminal, if not, first execute “npm install”, and then “npm ci”.

You should now have the packages-lock.json file and a folder called node\_modules. This folder

contains all our plugins for the front-end, the folder, and its contents should not be altered

manually. And you don’t need to upload it to your repository or when you publish your site.

- Then, start the react project typing “npm start” in the terminal  
  
## Project Structure <br />  
A short explanation of what each section of each project does:  
<br /> <br />

\*\*WebApiExample.Webapp:\*\* <br />

api <br />

|\_Constants

|&nbsp; &nbsp;|\_Https header config for GET/POST/DELETE calls<br />

|&nbsp; &nbsp;|\_Define base web api route: https://localhost:5001/api<br />

|\_Services <br />

|&nbsp; &nbsp; |\_Service methods for GET/POST/DELETE and default methods using fetch <br />

|&nbsp; &nbsp; |\_Common CRUD functions <br />

|\_Src<br />

&nbsp; &nbsp;|\_Components<br />

&nbsp; &nbsp;| &nbsp; &nbsp;|\_A folder for each component<br />

&nbsp; &nbsp;| &nbsp; &nbsp;|\_Table component used in employees.tsx<br />

&nbsp; &nbsp;|\_Pages <br />

&nbsp; &nbsp; &nbsp; &nbsp; |\_A folder for each page<br />

&nbsp; &nbsp; &nbsp; &nbsp; |\_Employees page tsx and style files<br />

&nbsp; &nbsp; &nbsp; &nbsp; |\_Plugins <br />   
&nbsp; &nbsp; &nbsp; &nbsp; |\_SCSS with general class styles  
<br /><br />  
\*\*WebApiExample.External:\*\*<br />

|\_Controllers

|&nbsp; &nbsp; |\_Http methods for web requisitions<br />

|\_Injection <br />

|&nbsp; &nbsp; |\_Dependency injection for service classes <br />

|\_Properties <br />

|&nbsp; &nbsp; |\_IIS configuration (applicationUrl port when executing External project)<br />

|\_Services <br />

|&nbsp; &nbsp; |\_Conection with repository layer.<br />

|\_Program.cs <br />

|&nbsp; &nbsp; |\_First class to be initialized when External application start<br />

|\_Startup.cs <br />

&nbsp; &nbsp; |\_Initialize cors policy for web requistions<br />

&nbsp; &nbsp; |\_Allow specific connection route (localhost:3000)<br />

&nbsp; &nbsp; |\_Added controllers to service collection

<br /> <br />  
\*\*WebApiExample.Persistence:\*\*<br />

|\_Injection<br />

|&nbsp; &nbsp; |\_Persistence injecton for factory class and model repositories <br />

|\_Models <br />

|&nbsp; &nbsp; |\_Database models, Entity Framework Core data context <br />

|\_Repository <br />

|&nbsp; &nbsp; |\_Database Repository for each model, with linq to sql interactions <br />

|\_DbContextSettings <br />

&nbsp; &nbsp; |\_Store database connection string from appsettings. The connection string is used in DbContextFactory