I build a simple Angular 6 frontend application. As there were no specific instructions how application should look I created layout by myself. I used simple solution so application consist with: header with application name, search interface section, search results and footer with disclaimer.

Regarding Search options, I gave user option to search: by name, by ingredients or combine both. For query search I used classic text input and for ingredients I chose to use select with multiple options as I found this better than user must type ingredients comma separated in to text input and also to show that I can work with more complex elements than just classic text input. Also I implemented pagination, as API is returning 10 results per page. For this reason, I also made grid on desktop with 5 columns, and on responsive mobile grid falls to 2 columns.

Technology used:

- Node.js – is webserver for running angular app locally for development. It contains NPM, which handle our dependencies and give us angular-cli.
- Angular 6 – as it latest version of Angular

Library used:

- Angular material – Used for frontend elements and their styling. I decided to use Angular Material because it has already styled elements (such as buttons, inputs, selects). Also provides us grid. Also Material for me is more „angular way“ than using Bootstrap or other Library, and preserve time of writing css.
- Rxjs – I use rxjs for asynchronous operations. It is used it for communication between components as Rxjs provide module BehaviorSubject, which offers us communication between components, even if they are not related (e.g.: parent->children).

Application architecture (description of all important parts of application).

- In root directory is package.json file all depdendcy information that is required for web application.
- Then in src/app are mian files.
  - Main component (app.component)
  - Other components
  - Services
- For all stylesheet in changed from CSS to SCSS. So we use precompiler for all styles: which allows us write styles quicker, use mixins and variables and etc.

Detailed description how web application is tied together:

Main component
Is our default root *component* that connects another components hierarchy inside this component.

Preloder component
I implemented preloader to make nice loading effect when service calls are executed. It is nice touch to show preloader while service calls are being executed. I implemented preloader with http_interceptor, so it automatically shows preloader on all requests.

RecipeSearch Component
This component contains search elements such as input, select, button and also pagination. Component calls RecipeService when user perform new search query and then new data (search results) are injected into RecipeList Component.

RecipesList component
This component is used for showing search results. We use material grid for displaying results. On load, we show default results (api call without params). On User search action, we get new data for search result list. Commutation between RecipeSearch component is done with rxjs BehavourSubject via message over Service.

Recipe service (services/recipe, service)
This is our service in application. Its purpose is to communicate with backend API via simple get requests. Our Api accepts 3 params (query, ingredients, page) and return 10 results per page. We get data in JSON, and then save each recipe in IRecipe Object.

Disclaimer 1:

Backend on server from Recipe Puppy isn't set up for Access-Control-Allow-Origin headers. This means that for all request from frontend to API we get CORS error. Ideally we could solve this on backend, but it's not our API and not part of this task.

So in this case I used simple solution and serve all request via simple proxy.

Procedure:

1. Client calls server-side proxy for information.
2. Proxy calls Recipe Puppy's API and translates or passes through information as needed.
3. Proxy relays that information to the client-side code for further processing.

On development:

I used proxy.conf.json file, and then serve app locally with command: npm start: "**ng serve –proxy config=proxy.conf.json**".

On production

Since I had Apache server available. I created proxy with few simple PHP lines.

File: proxy.php (is in root folder of the repository)

Code:

```
<?php
   $page = $_GET['p'];
        if($_GET['p'] == "" || $_GET['p'] == null) {
            $page = "1";
         }
   echo file_get_contents('http://www.recipepuppy.com/api/?q=' . $_GET['q'] . '&i=' . $_GET['i'] . '&p=' . $page );
?>
```

***Since I created this proxy and is available on my server, we can also use this proxy for local development requests. There in no need for local proxy: proxy.conf.json.**

Dejan Lužnic, 18.10.2018