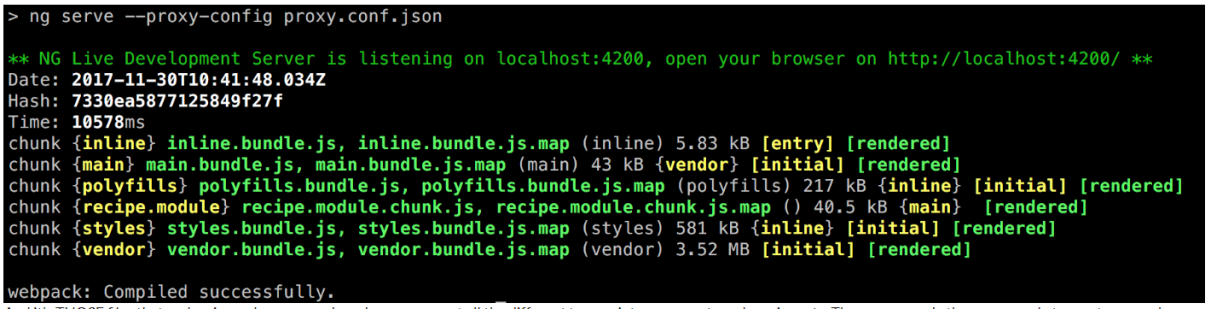
put your app on heroku

So we decided to put our app on Heroku (because heroku is free and we're cheap), until now our app consisted of two parts, the Angular frontend, and the NodeJs backend. We basically have two servers: one, obviously the NodeJS server, which serves our REST API; but also the Angular frontend. What we're really accessing is a compiled version of our app, served by Webpack.

When we start a 'ng serve' the first thing we always see is a compile step.



And it's THOSE files that end up in our browser and render our app, not all the different typescript components and services etc. Those are merely the source code to create our webapp, which consists of one html file and a few plain javascript files.

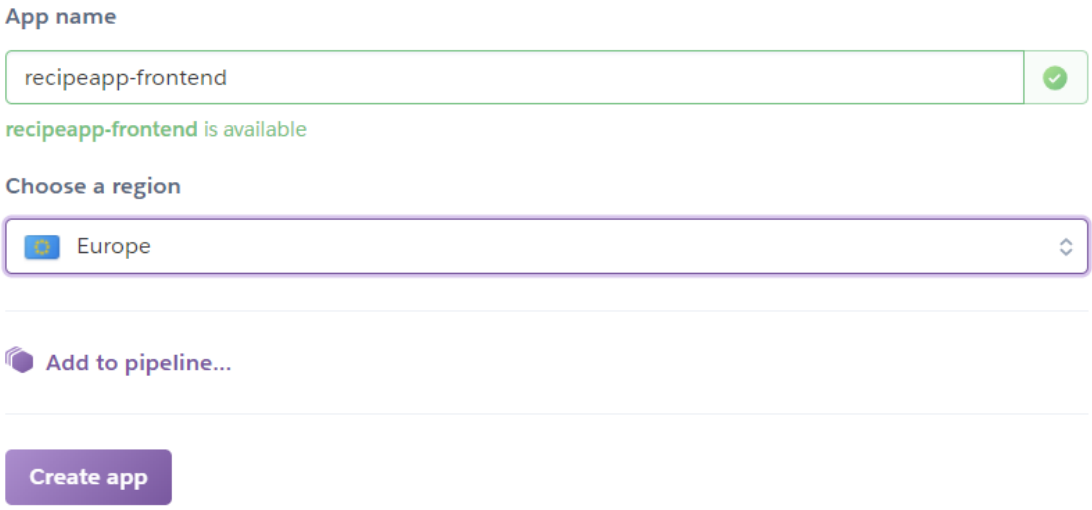
While these files are what we want to serve, they are nowhere to be found on our drive. When you do a ng serve, these files get built but only exist in memory. In order to create them you have to explicitly call 'ng build' 'ng build' creates a dist/ folder with all the files we need to serve our app, adding these files to the public/ folder of our nodejs app and adding a route so every request ends up in the (one) index.html file we have would actually be sufficient.

For this tutorial, the root is recipeapp, containing the folders frontend and backend and being a git repository.

Create 2 heroku apps, one for the frontend and one for the backend

You first have to register on heroku.com. Then create 2 new applications, one for the frontend and one for the backend.

- Create a new application for the frontend with as name e.g. “**recipeapp-frontend**”. Therefor click the new button at the top right side and select “new app”.



- Create a new application for the backend with as name e.g. "**recipeapp-backend**".

- Heroku presents the deploy tab for each of the applications. At the bottom you'll find the instruction to deploy an existing app on Heroku Git. You'll need this instruction later on.

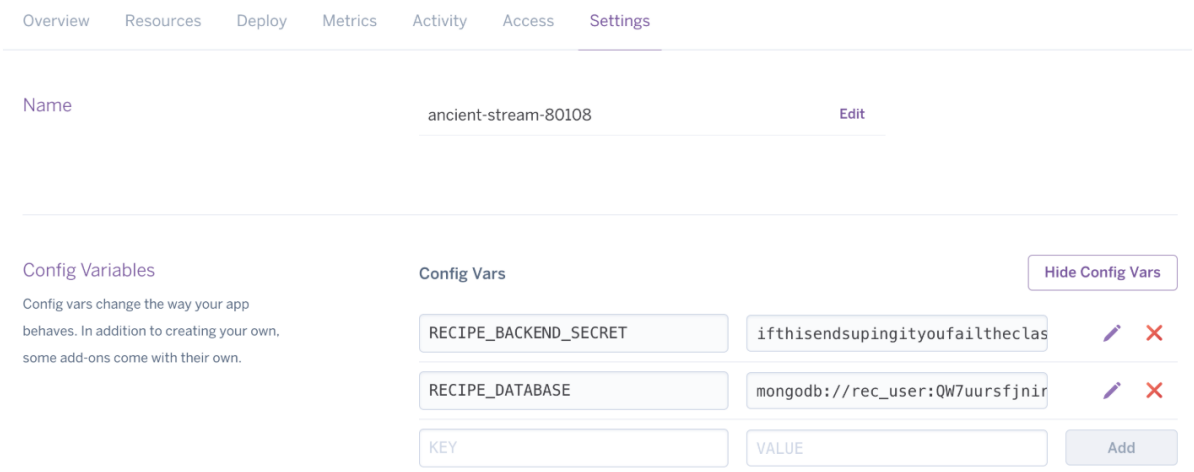
Create mongoose database on mlab.com

We still need a database, you can link mongodb plugins to your heroku app and get a (free) database that way, but you always have to add a credit card if you want to use plugins (even if you would never get charged). Since many students don't have credit cards, we'll link a database manually. Go to mlab.com, make an account, create a database for your app, and **add a database user** to it, in the end you should be able to access it through a url, something like:

```
mongodb://myuser:mypassword@somerandomstring12.mlab.com
```

Environment variables

We rely on a few environment variables to set our backend secret to secure our token, and to know where our database is. Go to the settings part of your app and the heroku website and set the environment variables there.



In Visual Studio Code : terminal Window

Install Heroku and log in

- Install Heroku CLI : see https://devcenter.heroku.com/articles/heroku-cli. You can install the application or use npm

```
$npm install -g heroku-cli
```
Verify your installation : `$heroku –v`

- Log in to heroku. Enter your heroku credentials

```
$heroku login
```

Create serve-frontend folder to deploy angular app on heroku

- Create a folder **serve-frontend** in the root. This folder will be deployed on heroku and contains the build version of the angular app. Heroku must be able to install the dependencies and run the app, therefor add a package.json file, an index.js file and a www folder. De www folder will contain the build angular files needed for the frontend

  - Code for package.json

```
{
  "name": "recipe-app-frontend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "gulp": "3.9.1",
    "gulp-rename": "^1.2.2",
    "express": "^4.14.0",
    "express-http-proxy": "0.10.1"
  }
}
```

  - Code for index.js.

```
'use strict';

var express = require('express');
var app = express();
var fs = require("fs");
var config;

app.listen(process.env.PORT, function() {
    console.log("Server running on port 8080.");
});

//Serve www folder
app.use(express.static(__dirname + '/www'));
app.use('*', express.static(__dirname + '/www/index.html'));
```

Configure the different remotes in the root folder.

- You'll need one remote for your backend application (see your instructions on the deploy tab of heroku for this app, section existing Git repository for the command)
  `$heroku git:remote -a recipeapp-backend`

- By default, the Heroku CLI names all of the Heroku remotes it creates for your app heroku. You can rename your remotes with the git remote rename command:

  `$git remote rename heroku recipeapp-backend`

- Configure another remote for your frontend application using (see deploy tab frontend repository)

  ```
  $heroku git:remote -a recipeapp-frontend
  $git remote rename heroku recipeapp-frontend
  ```

- You can query the remotes as follows. You should at least have those 4 lines (heroku, recipeapp-backend)

  ```
  PS C:\temp\recipeapp> git remote -v
  heroku  https://git.heroku.com/recipeapp-frontend.git (fetch)
  heroku  https://git.heroku.com/recipeapp-frontend.git (push)
  origin  https://github.com/Web-IV/recipeapp.git (fetch)
  origin  https://github.com/Web-IV/recipeapp.git (push)
  recipeapp-backend       https://git.heroku.com/recipeapp-backend.git (fetch)
  recipeapp-backend       https://git.heroku.com/recipeapp-backend.git (push)
  ```

- Add the following package.json to the root folder. Change the file where needed. The package contains different scripts : deploy-backend to push the changes in the backend to heroku, build-frontend to run ng build and commit the changes(commit-build-changes) and build the frontend, copy the static files to serve-frontend/www (copy command) to commit and push the changes to heroku(deploy-frontend)

```
{
  "name": "recipeapp",
  "version": "1.0.0",
  "description": "Recipe demo app",
  "homepage": "http://www.github.com/pieter-hogent/",
  "scripts": {
    "start": "node server",
    "deploy-backend": "git subtree push --prefix backend recipeapp-backend
master",
    "build-frontend": "cd frontend && ng build --prod",
    "commit-build-changes":
      "git add . && git commit -m 'chore(build): build files'",
    "deploy-frontend":
      "npm run build-frontend && npm run copy && npm run commit-build-changes
&& git subtree push --prefix serve-frontend recipeapp-frontend master",
    "copy": "cpx 'frontend/dist/**' serve-frontend/www"
  },
  "author": {
    "name": "Pieter Van Der Helst",
    "email": "pieter.vanderhelst@hogent.be",
    "web": "http://www.github.com/pieter-hogent/"
```

```
  },
  "repository": {
    "type": "git",
    "url": "git://github.com/pieter-hogent/webapps/"
  },
  "engines": {
    "node": ">=6.0.0"
  },
  "dependencies": {
    "connect": "^3.6.5",
    "serve-static": "^1.13.1"
  },
  "license": "MIT",
  "devDependencies": {
    "copyfiles": "^2.0.0",
    "cpx": "^1.5.0"
  }
}
```

Run npm install in the root folder

Change the backend code

app.js : connect to the mlab database when running in production

```
mongoose.connect(process.env.MONGOLAB_URI || 'mongodb://localhost/recipedb');
```

app.js : use cors for cross origin requests

```
var users = require('./routes/users');

var app = express();

let cors = require('cors');

app.use(cors({origin: "*"}));
```

package.json: run application local or remote

$npm install --save cors

```
"scripts": {
…
    "start-local": "nodemon ./bin/www",
     "start": "node ./bin/www"
…
```

```
  },
```

To run the backend locally, you now must type : npm run start-local

Heroku will use the start script : npm start

Deploy backend on Heroku

You can still commit your changes to your git repository

To deplay on heroku :

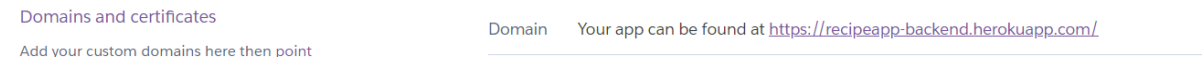
$npm run deploy-backend

You'll find the script in the package.json in the root folder. The backend will be deployed on heroku. You can test with insomnia. The url can be find on the settings tab.

Change the frontend

Define the url of the backend application.

You can find the url in heroku, select the recipeapp-backend and go to the settings tab



Open environments.prod.ts, and add the backend_url

```
export const environment = {
  production: true
};

export const BACKEND_URL = "https://recipeapp-backend.herokuapp.com";
```

in environment.ts voeg je toe

```
export const BACKEND_URL = null;
```

Add an interceptors to change the url to the need to be changed for

- Add in the interceptors folder file base-url.interceptors.ts

```
import { Injectable } from '@angular/core';
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from
'@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { BACKEND_URL } from '../../environments/environment';

@Injectable()
export class BaseUrlInterceptor implements HttpInterceptor {
  constructor() {
      }
```

```
  intercept(req: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
    if(BACKEND_URL) {
      req = req.clone({
        url: `${BACKEND_URL}${req.url}`
      });
      // console.log('set to ', `${BACKEND_URL}${req.url}`);
    }
    return next.handle(req);
  }
}
```

- Change index.ts in the interceptors folder
  Voeg toe

```
export const basehttpInterceptorProviders = [
  {
    provide: HTTP_INTERCEPTORS,
    useClass: BaseUrlInterceptor,
    multi: true
  }
];
```

- Change recipe.module.ts : add basehttpInterceptorProvider

```
providers: [basehttpInterceptorProviders, httpInterceptorProviders,
RecipeDataService, RecipeResolver]
```

- Change users.module.ts : add

```
providers: [basehttpInterceptorProviders, AuthenticationService,
AuthGuardService ],
```

- RecipeDataService : Updated route

```
private readonly _appUrl = '/API';
```

Deploy frontend on Heroku

First commit your changes to your git repository. Then run the script (see package.json in the root)

$npm run deploy-frontend

If you do this the first time it's not unlikely you'll bump into some errors when building the application (ng build command), the two most common ones are:

- using private i.s.o. public for properties you use outside your class (private variables are aggressively minified, so they can't be found anymore), this is fixed by making those properties public
- accessing FormArray's as AbstractControls (which doesn't have a .controls property), this is fixed by adding a getter to the .ts file which explicitly casts.

Solve the errors and try again.

Surf to your website on heroku

To run the website locally : npm start

The code can be found on repository tvh : https://github.com/tvh-parts/hogent-recipe-app