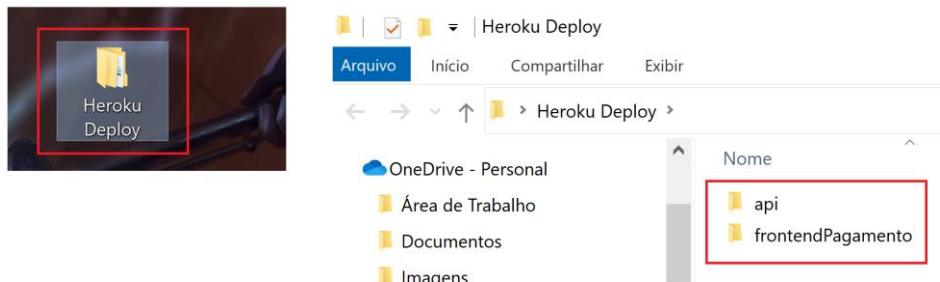


Processo de deploy

Primeiro de tudo eu crio uma pasta para onde eu vou COPIAR tanto o backend e o frontend da aplicação (*porque esses arquivos vão ser modificados*):



Build do projeto Angular:

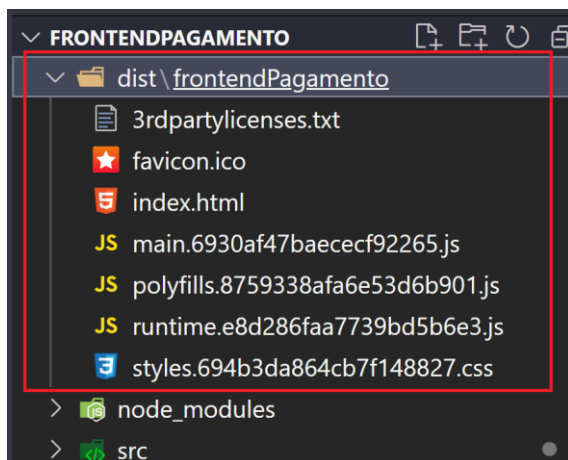
Os endpoints do front end da aplicação agora serão consumidos diretamente pela **própria url do heroku**, então, é necessário efetuar essa mudança:

```
export class FuncionariosService {  
  readonly API = "https://nome-da-aplicacao.herokuapp.com/api/empresas";  
  constructor(private http:HttpClient) { }  
  getFuncionarios() : Observable<IFuncionario[]> {  
    return this.http.get<IFuncionario[]>(this.API + "/funcionarios");  
  }  
}  
  
export class HoleritesService {  
  readonly API = "https://nome-da-aplicacao.herokuapp.com/api/holerites";  
  constructor(private http:HttpClient) { }  
  getHoleriteById(idHolerite: number) : Observable<IHolerite> {  
    return this.http.get<IHolerite>(`${this.API}/${idHolerite}`);  
  }  
}
```

O próximo passo é fazer build do projeto Angular com o comando **"ng build --prod"**

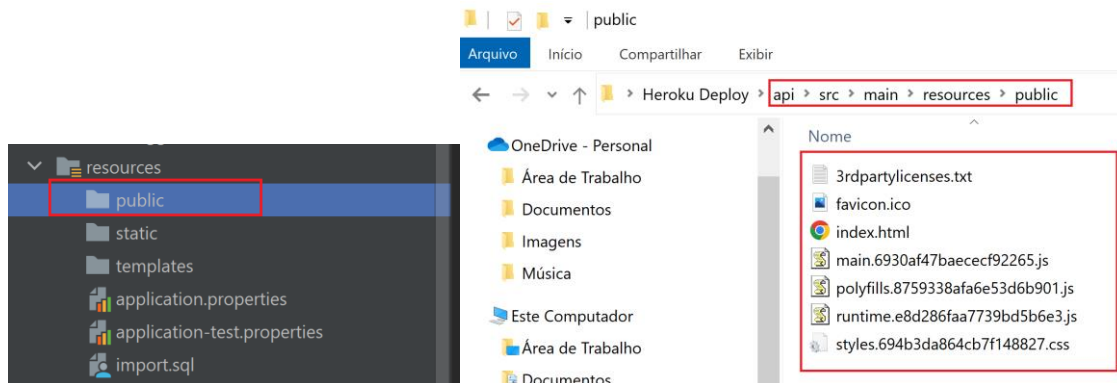
```
Microsoft Windows [versão 10.0.19044.1645]  
(c) Microsoft Corporation. Todos os direitos reservados.  
  
C:\Users\lucas\Desktop\Heroku Deploy\frontendPagamento>ng build --prod
```

A ação do build do projeto angular cria uma pasta chamada **"dist"**, essa pasta contém o **index.html** da aplicação e todos os **componentes já transformados em javascript**. Porém esse index não roda localmente, ele precisa de um node server:

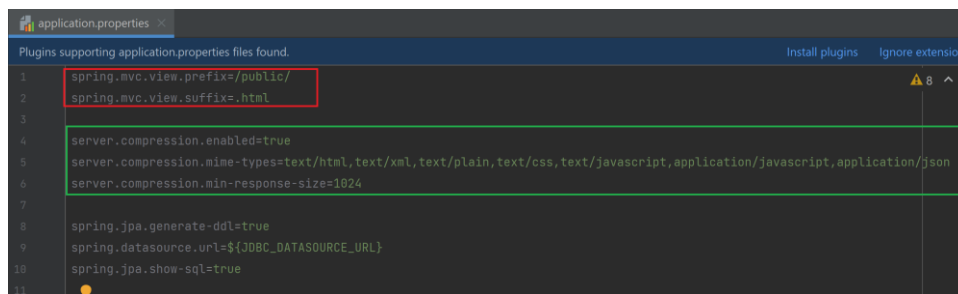


Linkando o Front End buildado com o Backend e tendo finalmente o view

Agora temos a camada de view do nosso MVC (os arquivos gerados dentro de “dist” no build do projeto Angular), precisamos **adiciona-lo ao projeto java**. Então na pasta de resources vamos criar uma pasta “public” e copiar todos os arquivos da “dist” :



Agora é preciso mudar as configurações no application properties para indicar **aonde está o view do projeto**, algumas configurações de **compressão para performance**:



Outra mudança necessária é a dependência do banco de dados, o Heroku só fornece gratuidade se o banco de dados for postgres:



Configurando o Spring

Colocamos o frontend na pasta public, agora é necessário adicionar uma configuração para que o Heroku assuma as rotas do Angular e para que o Spring fale pro Heroku qual o nosso front end:

```

@Configuration
public class MyAppWebMvcConfigurer implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/public/**")
            .addResourceLocations("classpath:/public/")
            .resourceChain(cacheResources: true)
            .addResolver(new PathResourceResolver() {
                @Override
                protected Resource getResource(String resourcePath, Resource location) throws IOException {
                    Resource requestedResource = location.createRelative(resourcePath);
                    return requestedResource.exists() && requestedResource.isReadable() ?
                        requestedResource : new ClassPathResource("/public/index.html");
                }
            });
    }
}

```

Então essa configuração basicamente indica aonde está o front end da aplicação.

Deploy no Heroku

Agora que o backend possui o view da aplicação já dá pra subir no heroku. Essa pasta que tem o backend vai virar um repositório, só dar um “git init” nela:

```

C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.19044.1645]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\lucas\Desktop\Heroku Deploy\api>git init
Initialized empty Git repository in C:/Users/lucas/Desktop/Heroku Deploy/api/.git/

```

Depois disso a gente faz login no heroku com “heroku login” (precisa ter instalado na máquina):

```

C:\Windows\System32\cmd.exe
C:\Users\lucas\Desktop\Heroku Deploy\api>heroku login
Warning: heroku update available from 7.54.1 to 7.60.0.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/c78af084-b6c3-4594-b50b-ea65d9eed53f?requestor=SFMyNTY.g
2gDbQAAAA4xNzAuMTUwLjc0LjIzNG4GABDTz3qAAWIAAVGA.39hkq59yHD-dcL_kudJU2df04sQZxcXptgBxB8Bdw0Y
Logging in... done
Logged in as lucasevizar@hotmail.com

```

Depois disso temos que criar o projeto, com o nome que a gente definiu nos endpoints do angular lá trás:

```

C:\Users\lucas\Desktop\Heroku Deploy\api>heroku create nome-da-aplicacao
Warning: heroku update available from 7.54.1 to 7.60.0.
Creating nome-da-aplicacao... done
https://nome-da-aplicacao.herokuapp.com/ | https://git.heroku.com/nome-da-aplicacao.git

```

Agora tem que adicionar o postgres na máquina virtual do heroku:

```
C:\Windows\System32\cmd.exe
C:\Users\lucas\Desktop\Heroku Deploy\api>heroku addons:create heroku-postgresql:hobby-dev
» Warning: heroku update available from 7.54.1 to 7.60.0.
Creating heroku-postgresql:hobby-dev on nome-da-aplicacao... free
Database has been created and is available
! This database is empty. If upgrading, you can transfer
! data from another database with pg:copy
Created postgresql-polished-81352 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
```

Agora só dar um git push pra branch do Heroku que se tiver tudo certo a aplicação vai subir e ficar disponível:

```
C:\Users\lucas\Desktop\Heroku Deploy\api>git push heroku master
Enumerating objects: 77, done.
Counting objects: 100% (77/77), done.
Delta compression using up to 12 threads
Compressing objects: 100% (66/66), done.
Writing objects: 100% (77/77), 175.14 KiB | 8.34 MiB/s, done.
Total 77 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Java app detected
remote: -----> Installing JDK 17.0.2... done
remote: -----> Executing Maven
remote:      $ ./mvnw -DskipTests clean dependency:list install
```