UNIVERSITÉ DE RENNES 1

ESIR ECOLE SUPERIEURE D'INGENIEURS DE RENNES

16/01/2019

# TD 3

## Galerie de photos

> *« J'atteste que ce travail est original, qu'il indique de façon appropriée tous les emprunts, et qu'il fait référence de façon appropriée à chaque source utilisée »*

Léo Guilpain

# Table des matières

# Partie I - "Hello World" Node.js + Express avec Docker

## Création du "Hello World"

### Question n° 1 et 2 :

Après avoir initialisé le projet, nous pouvons tester sur l'adresse « http://localhost:3000 » . On obtient ceci :

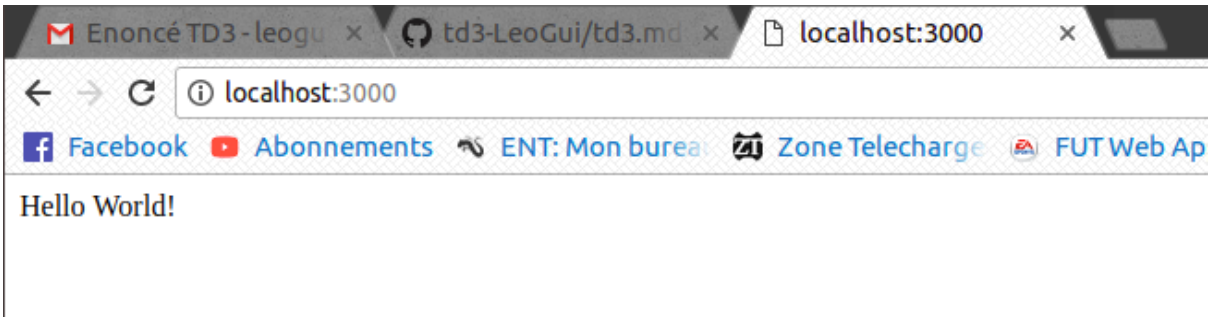


*Figure 1 : Hello World*

### Question n°3 :

On modifie le package-json comme ceci :

```
{
  "name": "gallery",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "express": "^4.16.4"
  },
  "devDependencies": {},
  "scripts": {
    "start": "node index.js"
  },
  "author": "",
  "license": "ISC"
}
```

*Figure 2 : package-json*

Lorsque l'on lance la commande « npm start », cela signifie que l'on fait « node index.js ».

## Conteneurisation de notre app

### Question n° 1

La taille de l'image est de 674 MB. En faisant « docker pull node » on récupère la dernière image qui est de 898 MB.

### Question n°2

Pour générer le DockerFile, je me suis aidé de ce site :
https://nodejs.org/en/docs/guides/nodejs-docker-webapp/

On obtient donc ceci pour le Dockerfile :

```
FROM node:8

# Create app directory
WORKDIR /usr/src/app

RUN npm install

COPY . .

CMD [ "npm", "start" ]
```

*Figure 3 : Dockerfile*

On exécute ensuite la commande suivante :

```
leo@leo:~/ESIR3/Cloud/TP3/td3-LeoGui/td3/gallery$ docker run --rm --name mynodeapp -p 3000:3000 mynodeimage

> gallery@1.0.0 start /usr/src/app
> node index.js

App listening on http://localhost:3000!
Client requested /
Client requested /
```

*Figure 4 : docker run*

En exécutant la commande "docker images", on voit bien que l'image a été créée.

```
leo@leo:~/ESIR3/Cloud/TP3/td3-LeoGui/td3/gallery$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mynodeimage         latest              d0c40f48bfc5        2 minutes ago       897MB
```

*Figure 5 : docker images*

### Question n°3

Pour que les modules soient installés lors du build de l'image Docker il faut rajouter ceci dans le fichier DockerFile :

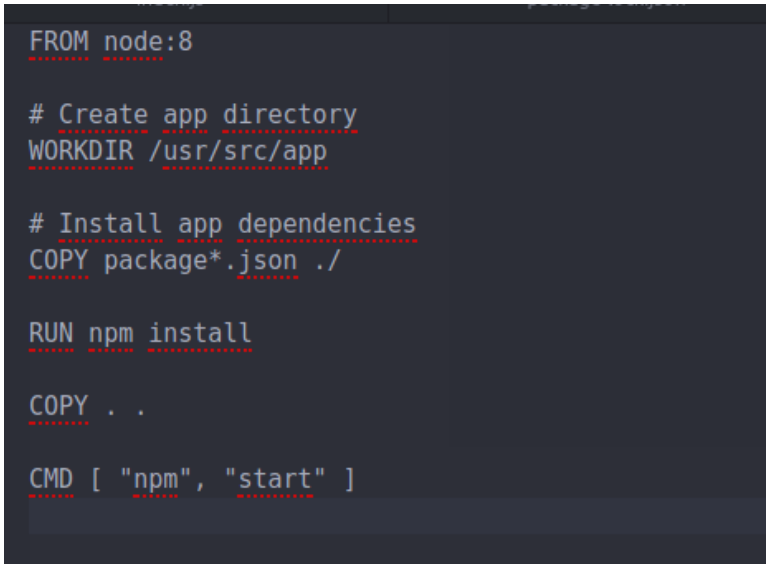


*Figure 6 : Dockerfile*

Je me suis servi du site suivant :
https://www.sitepoint.com/create-new-express-js-apps-with-express-generator/

## Partie II - Création d'une galerie d'images statique

La difficulté de cette partie a été de rendre dynamique l'affichage des albums et des photos
J'ai utilisé l'arborescence des images pour gérer leur titre ainsi que les albums. J'obtiens donc ceci :

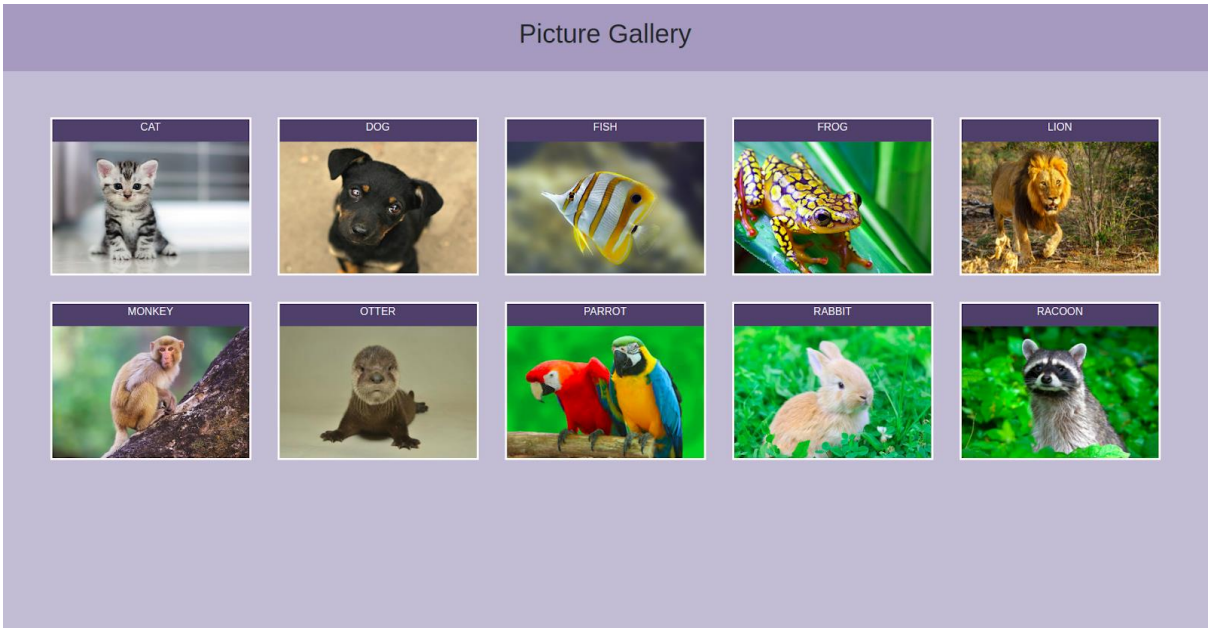


*Figure 7 : page d'accueil / index.js*

Ici on peut voir 4 albums possédant une miniature. Si on ajoute ou supprime un album cela l'ajoutera ou le supprimera automatiquement sur cette page.

Ensuite lorsque l'on clique sur une des miniatures, on arrive sur la page correspondant à l'album (celle-ci est dynamique et s'adapte aux photos de l'album). Un carrousel se trouve sur cette page permettant de faire défiler les photos.

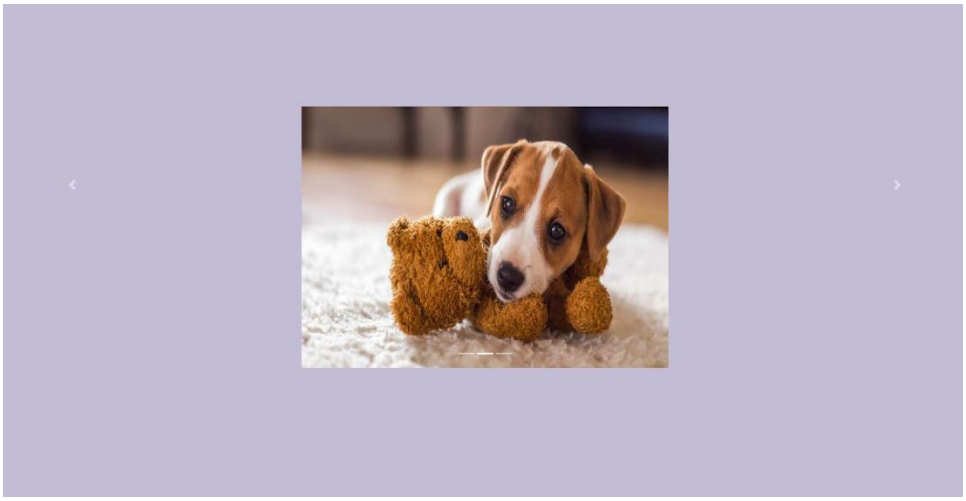

*Figure 8 : Carrousel*

Ensuite pour déployer la galerie avec Docker, j'ai créé un Dockerfile dans mon projet. Je l'ai ensuite complété comme ceci :

```
FROM node:8

# Create app directory
WORKDIR /usr/src/myappgallery

# Install app dependencies
COPY package*.json ./

VOLUME ./public/images /usr/src/myappgallery/public/images

RUN npm install

# Bundle app source
COPY . .

CMD [ "npm", "start" ]
```

*Figure 9 : Dockerfile*

Après avoir build, on run l'image avec la commande suivante :

```
leo@leo:~/ESIR3/Cloud/TP3/td3-LeoGui/td3/gallery/myappgallery$ docker run --rm --name myapp -p 3000:3000 myappimage

> myappgallery@0.0.0 start /usr/src/myappgallery
> node ./bin/www
```

*Figure 10 : docker run*