МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
Мегафакультет трансляционных информационных технологий
Факультет информационных технологий и программирования

Лабораторная работа №1
По дисциплине «Web-программирование»
Хостинг веб-приложения на сервисе Heroku

Выполнили студент группы M33081
Аль Даббагх Харит Хуссейн

Проверил
Приискалов Роман Андреевич

САНКТ-ПЕТЕРБУРГ

2022

СОДЕРЖАНИЕ

Installation 2

Installing NodeJS 2

Installing NestJS CLI.....2

Registration on Heroku 3

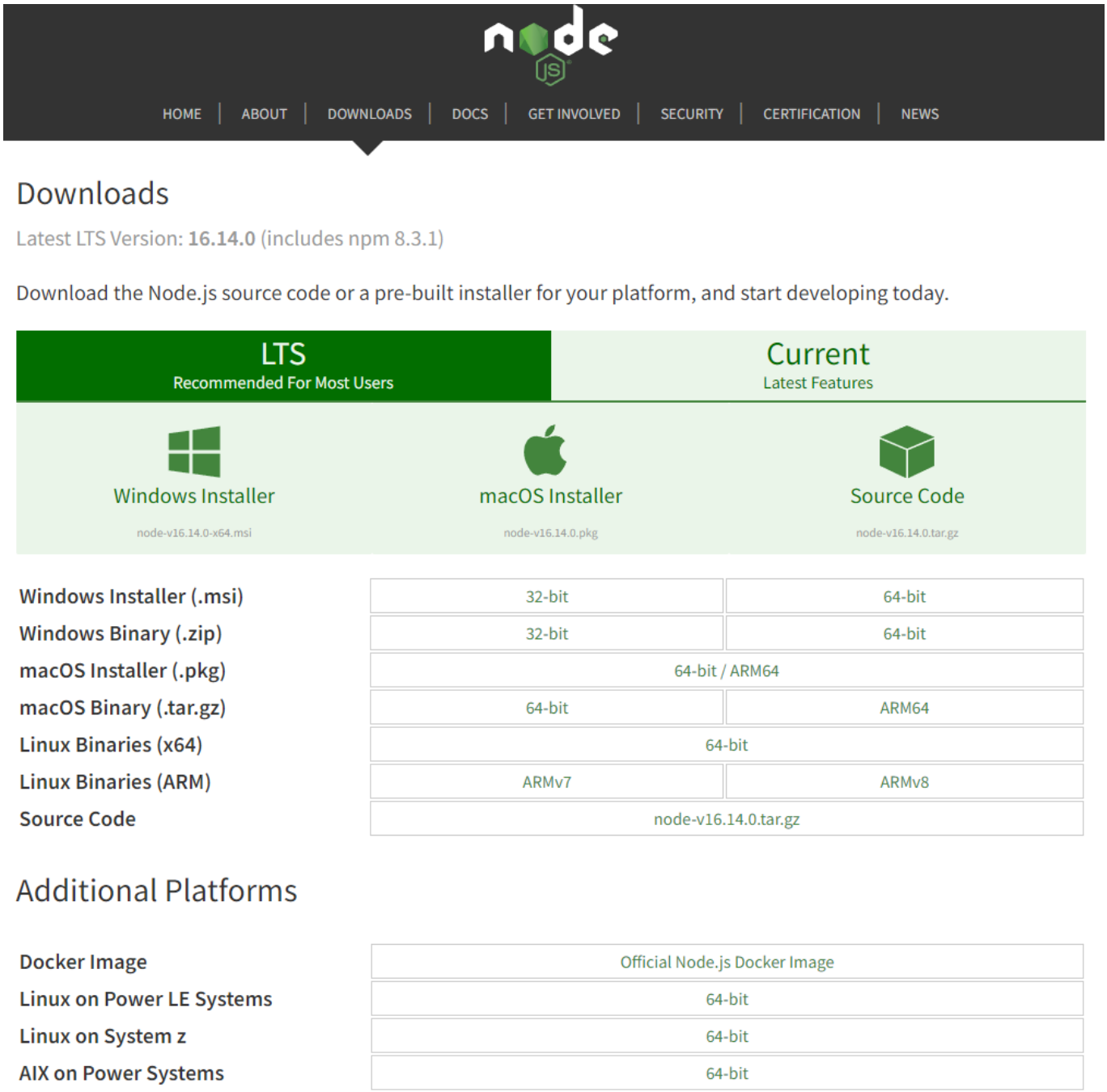
Installing Heroku CLI 4

Project Creation 5

INSTALLATION

Installing NodeJS

To install node we can easily do that from their [website download page]():



Installing NestJS CLI

After installing NodeJS we're gonna get the option to install other packages inside our app and/or globally.

For example to install NestJS we can use:

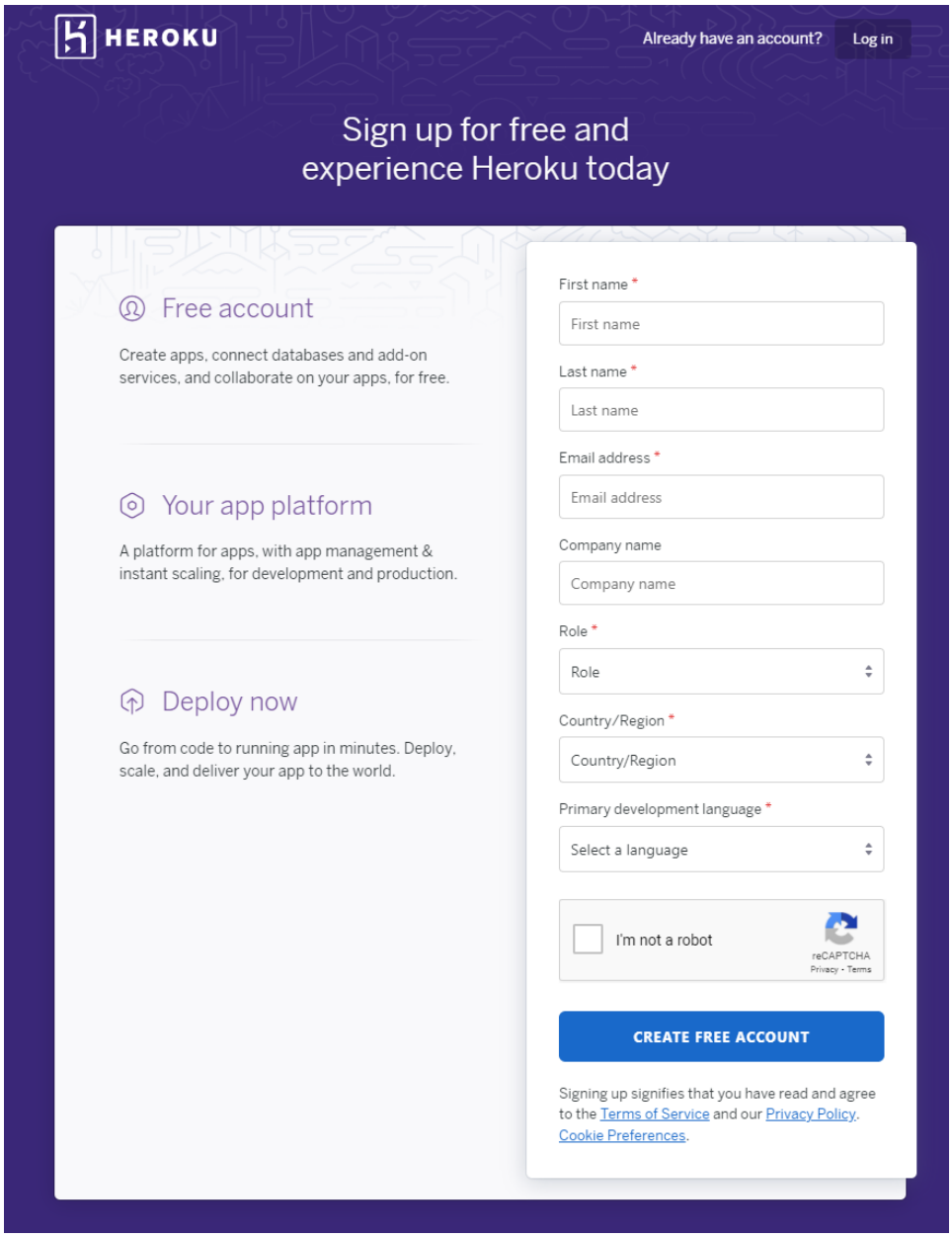
```
npm install -g @nestjs/cli
```

npm is the default package manager for node, there are also other package managers that can be installed and used like yarn and pnpm.

-g flag means that we want to install globally.

Registration on Heroku

You can register to heroku easily using their form:



Installing Heroku CLI

Heroku CLI can be installed depending on our system using their page:

Install the Heroku CLI

Pre-requisites

The Heroku CLI requires **Git**, the popular version control system. If you don't already have Git installed, complete the following:

- Git installation
- First-time Git setup

Install with an Installer

The Windows installers display a warning titled "Windows protected your PC" to some users. To run the installation when this warning shows, click "More info", verify the publisher as "salesforce.com, inc", then click the "Run anyway" button.

macOS

```
$ brew tap heroku/brew && brew install heroku
```

Windows

Download the appropriate installer for your Windows installation:

64-bit installer

32-bit installer

Ubuntu 16+

Run the following from your terminal:

```
$ sudo snap install --classic heroku
```

Snap is available on other Linux OS's as well.

In my case using windows 64-bit installer.

PROJECT CREATION

To create a NestJS app with boilerplate we use the following command:

```
nest new web-6th-sem
```

Within the boilerplate there is a package.json file. We changed some of it's contents according to the lab requirements:

```
{
  "name": "web-6th-sem",
  "version": "0.0.1",
  "description": "",
  "author": "Harith Al-Dabbagh <messi4ever500@gmail.com> (https://harith.me/)",
  "private": true,
  "license": "UNLICENSED",
  ▷ Debug
  "scripts": {
    "prebuild": "rimraf dist",
    "build": "nest build",
    "format": "prettier --write \"src/**/*.ts\" \"test/**/*.ts\"",
    "start": "nest start",
    "start:dev": "nest start --watch",
    "start:debug": "nest start --debug --watch",
    "start:prod": "node dist/main",
    "lint": "eslint \"{src,apps,libs,test}/**/*.ts\" --fix",
    "test": "jest",
    "test:watch": "jest --watch",
    "test:cov": "jest --coverage",
    "test:debug": "node --inspect-brk -r tsconfig-paths/register -r ts-node/register node_modules/.bin/jest --runInBand",
    "test:e2e": "jest --config ./test/jest-e2e.json"
  },
  "dependencies": {
    "@nestjs/common": "^8.0.0",
    "@nestjs/config": "^1.1.6",
    "@nestjs/core": "^8.0.0",
    "@nestjs/platform-express": "^8.0.0",
    "hbs": "^4.2.0",
    "reflect-metadata": "^0.1.13",
    "rimraf": "^3.0.2",
    "rxjs": "^7.2.0"
  },
```

During the lab work we are required to use environment variables for our app. To do this we installed the configuration module for nest.

The variable we're going to define is the port number.

Made some changes to the already created src/main.ts file to check if the variable is being used or undefined. If it's undefined then it's gonna use port 3000.

```
main.ts
src > main.ts > ...
import { NestFactory } from '@nestjs/core';
import { NestExpressApplication } from '@nestjs/platform-express';
import { join } from 'path';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);
  app.useStaticAssets(join(__dirname, '..', 'public'));
  app.setBaseViewsDir(join(__dirname, '..', 'views'));
  app.setViewEngine('hbs');
  const port = process.env.PORT || 3000;
  console.log(`App listening on port ${port}`);
  await app.listen(port);
}
bootstrap();
```

We check the app if it's working locally using the command:

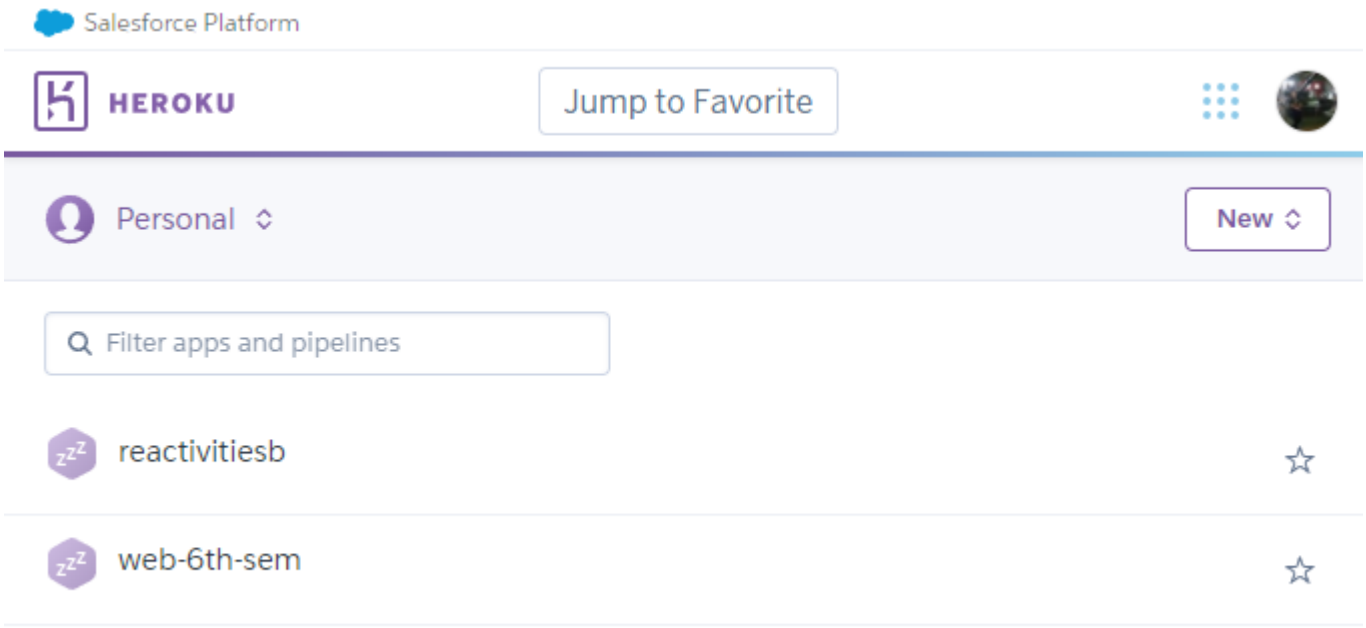
```
npm run start:dev
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

[2:54:19 PM] Starting compilation in watch mode...

[2:54:21 PM] Found 0 errors. Watching for file changes.

[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [NestFactory] Starting Nest application...
[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [InstanceLoader] ConfigHostModule dependencies initialized +20ms
[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [InstanceLoader] AppModule dependencies initialized +0ms
[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [InstanceLoader] ConfigModule dependencies initialized +0ms
App listening on port 3000
[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [RoutesResolver] AppController {/}: +5ms
[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [RouterExplorer] Mapped {/, GET} route +1ms
[Nest] 26084  - 02/24/2022, 2:54:22 PM     LOG [NestApplication] Nest application successfully started +2ms
```

Next we can create an App on Heroku using it's installed CLI:

```
heroku apps:create web-6th-sem
```

And check on the website if the app is created:



For Heroku to be able to start the app, we need to specify the command a file called Procfile, the contents of which is this single line:

```
web: npm run start:prod
```

NOW, this has all been on the backend side of the application. Now we need to bring our frontend that we created in the last semester.

To do that we created a MVC app inside our project, but we also need a template engine render our HTML views, for example Handlebars. To install Handlebars:

```
npm install --save hbs
```

And as before. We need to add the following to our main.ts file:

```
main.ts
src > main.ts > ...
import { NestFactory } from '@nestjs/core';
import { NestExpressApplication } from '@nestjs/platform-express';
import { join } from 'path';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);
  app.useStaticAssets(join(__dirname, '..', 'public'));
  app.setBaseViewsDir(join(__dirname, '..', 'views'));
  app.setViewEngine('hbs');
  const port = process.env.PORT || 3000;
  console.log(`App listening on port ${port}`);
  await app.listen(port);
}
bootstrap();
```

This means that the public directory will be used for storing static assets. Now we just need to copy our assets from the last semester to the public folder and restart the app.

Now we can commit and push to Heroku using the commands:

```
git add .
git commit -m "MESSAGE"
git push heroku master
```

```
BL4CK@DESKTOP-QEDDTPO MINGW64 ~/Desktop/6th Semester/Web/web-6th-sem (master)
$ git push heroku master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/nodejs
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment
remote:
remote:        NPM_CONFIG_LOGLEVEL=error
remote:        NODE_VERBOSE=false
remote:        NODE_ENV=production
remote:        NODE_MODULES_CACHE=true
remote:
remote: -----> Installing binaries
remote:        engines.node (package.json):  unspecified
remote:        engines.npm (package.json):   unspecified (use default)
remote:
remote:        Resolving node version 16.x...
remote:        Downloading and installing node 16.14.0...
remote:        Using default npm version: 8.3.1
remote:
remote: -----> Restoring cache
remote:        Cached directories were not restored due to a change in version of node, npm, yarn or stack
remote:        Module installation may take longer for this build
remote:
remote: -----> Installing dependencies
remote:        Installing node modules
```

Heroku will build and deploy the app and host it on:

https://web-6th-sem.herokuapp.com/