

Website installation and configuration

For a Node.js/Express.js Website

This tutorial will explain how to setup your computer to develop a website in node.js/express.js. It will also explain how to deploy your site online using several free to use (some limitations apply) sites.

We will be using the following technologies and resources (just a summary, we will cover this in more detail later) :

- **HTML – HypertText Modeling Language** – Used to define the content of web pages
 - <https://www.w3schools.com/html/> (w3schools is your “go to” resource for websites)
- **CSS – Cascating Style Sheets** – Used to describe the style (presentation) of the web pages
 - <https://www.w3schools.com/css>
- **Javascript** – Programming language for web development – Used to make calculations and modifications to and for your web pages
 - <https://www.w3schools.com/js/>
- **Node.js and Express.js** – For creating a server that will manage the requests and responses of web pages and other data (we will only see what is essential to build a working site)
 - **Node.js** – Open source server environment – It is programmed in javascript
 - <https://nodejs.org/en/docs/guides/>
 - **Express.js** – Node.js web application framework (it is a Node.js module) – Used to create web applications more easily
 - **Express generator** – Tool to quickly create an application skeleton
 - <https://expressjs.com/en/starter/generator.html>
 - <https://expressjs.com/en/starter/basic-routing.html>
 - **API Reference:** <https://expressjs.com/en/4x/api.html>
- **REST – REpresentational State Transfer** (just a concept, nothing to install) - Defines a way to provide communication between computer systems
We will use the REST rules to define our server crating a RESTfull Application (or close)
 - <http://www.restapitutorial.com/lessons/whatisrest.html>
 - Documenting: <https://bocoup.com/blog/documenting-your-api>
- **AJAX and JQuery** – To allow web pages to get and send data to the server (server then communicates with database).
 - **jQuery** is a JavaScript Library to simplify Javascript programming.
We will use it only for AJAX requests: https://www.w3schools.com/jquery/ajax_ajax.asp
 - **AJAX - Asynchronous JavaScript And XML** (just a concept, nothing to install) - Allows the communication of web pages with the server and dynamically updating the page without reloading the whole page (much more convenient than the other option)
We will not use XML we will use JSON .
- **JSON - JavaScript Object Notation** (just a concept, nothing to install) – A syntax for storing and exchanging data – We will use it to send and receive data between web pages and server
 - https://www.w3schools.com/js/js_json_intro.asp
- **VSCode – Visual Studio Code** – Editor for code, includes all things web (HTML, CSS, Javascript, ...)
 - <https://code.visualstudio.com/>

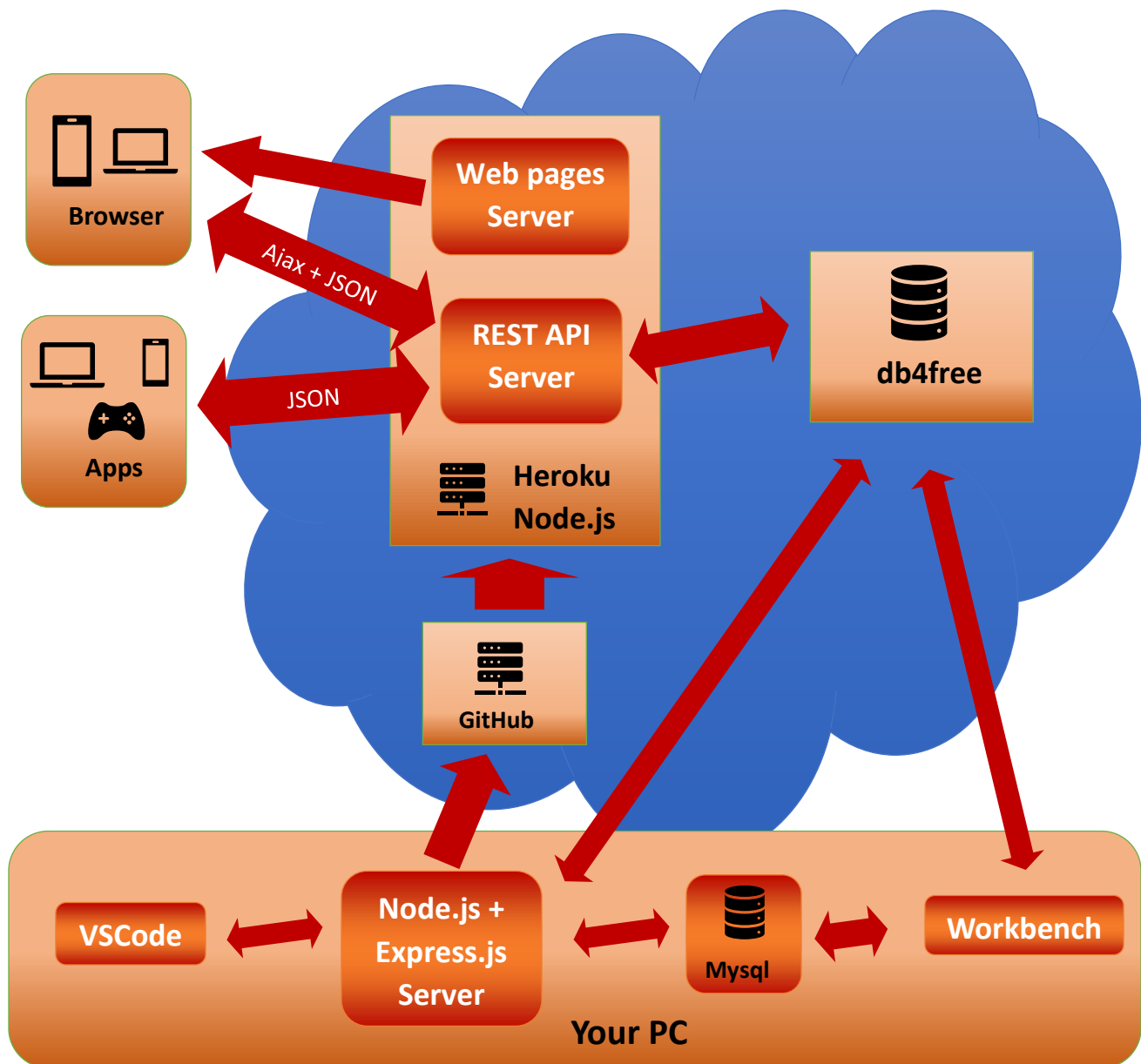
- You can also get Notepad++ - Also and editor for any type of text file – lighter and good for quick modifications
 - <https://notepad-plus-plus.org/downloads/>
- Mysql – For developing the database
 - Db4free – Free mysql hosting (some limitations: size, only one user and only one database)
 - <https://www.db4free.net/>
 - Make backups of database since host may delete at any time
 - Workbench + Mysql – For running and managing a database locally
May be needed to work locally
 - Mysql – It is the database itself
 - Workbench – Allows the design and management of databases (uses mysql)
 - Workbench installs mysql: <https://dev.mysql.com/downloads/workbench/>
 - Please check tutorial below before installing (if already installed a workaround is shown)
- Heroku and GitHub – For deploying your website
 - GitHub – It is a development platform – It is used to allow a community of developers to host and manage their code, including previous versions
 - <https://github.com/>
 - Heroku – It is a PaaS (platform as a service) – We will use it to host our website (needs GitHub to get the code)
 - <https://www.heroku.com/>
- Fork – A client for managing git and github projects: <https://git-fork.com/>
 - Currently you can use Fork as an evaluation freely, it will show you a pop up from time to time to get a paid license, but you can cancel and continue to use if freely
 - You can use another client but this is the one used in demonstrations
- Postman – Tool for testing REST APIs (**not mandatory but sometimes useful**)
 - <https://www.postman.com/>
 - <https://www.guru99.com/postman-tutorial.html>
- Chrome DevTools and equivalents – Tool for debugging web pages (**not mandatory but sometimes useful**)
 - <https://developers.google.com/web/tools/chrome-devtools>
 - We only need a small part of this to help on our page debugging

Next topics:

Architecture of a website.....	3
Installing node.js and express.js	4
Installing and editor: VSCode.....	7
Install the module to work with the MySQL database.....	10
Setting up a website: Github and Heroku.....	11
Setting up a GitHub repository	12
Setting up a Heroku application.....	14

Architecture of a website

The following image will try to summarize all the components of a web application, their relations, and the corresponding resources/concepts.



Summarizing the image:

On "Your PC":

- You will use VSCoDe to develop the Node.js/Express.js server and all the web pages (with HTML, CSS and javascript).

- Your server can use the local mysql database or the db4free database (if the network configuration allows). You can use workbench or another client (db4free has phpadmin) to manage the database.
- You can upload your code to GitHub that, after configuration, will deploy the application on Heroku.
- The structure of the Node.js server on Heroku is the same as the server on your PC:
 - A web pages server that will serve HTML, CSS, Javascript and other files (ex: images) to the browsers
 - A REST API that will serve data requested by web pages on the browser (using AJAX and JSON) and could also serve data to other applications (using JSON).
 - Other application may also use the web page server to get files like images but that was not represented (only browsers use HTML, CSS and Javascript).
 - Since the REST API is responsible to manage data, only this server will communicate with the database. In the case of Heroku that database must be online (remotemysql)

You can also do everything using only your PC. In that case the PC will have the browser, the Node.js server (both servers for web pages and REST API) and the mysql database.

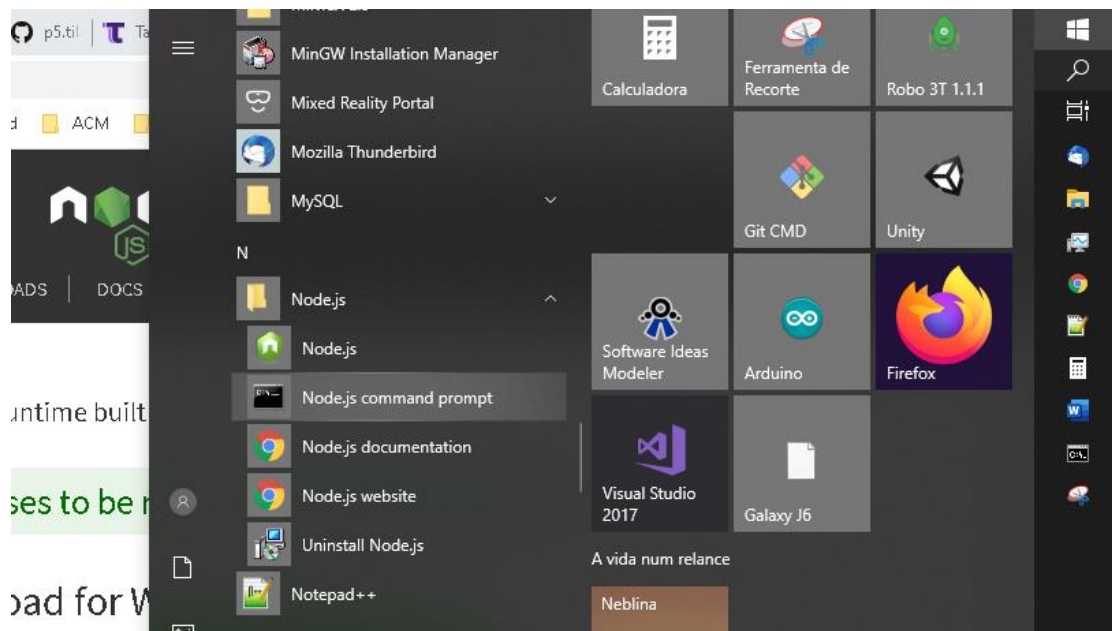
Communication between the modules will be the same, but all done inside your PC (access from other PCs is possible but depends on settings).

Installing node.js and express.js

Go to the site: <https://nodejs.org/en/>

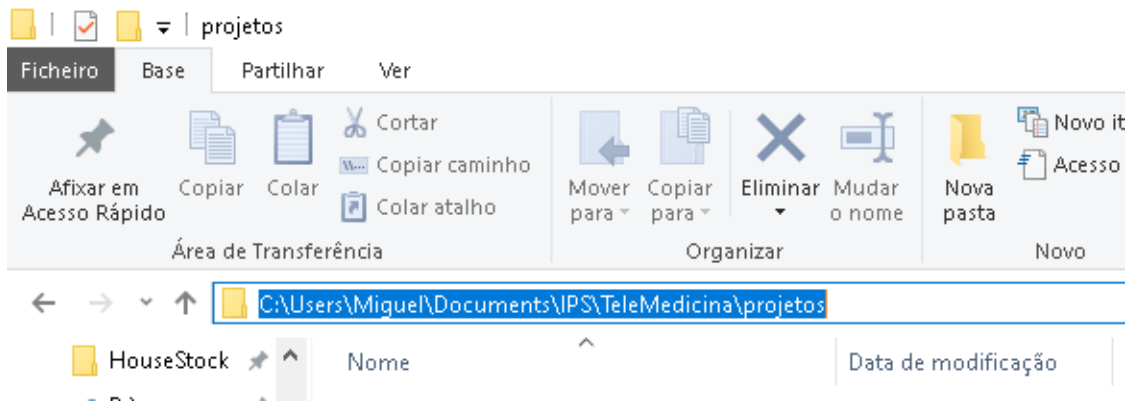
Choose the LTS version (recommended for most users). Download and install Node.js.

Run the command line terminal for node.js:

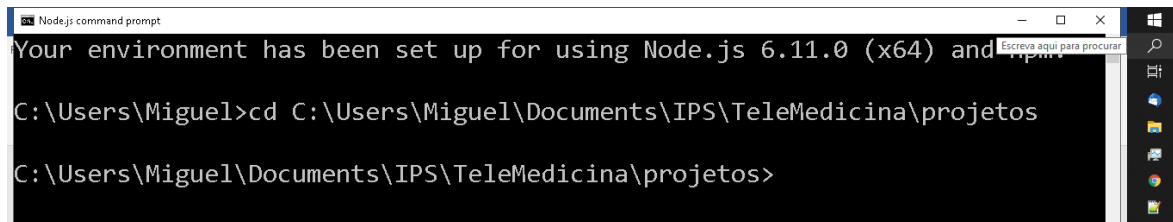


Change to your working directory (for the project you will need to have a directory with the code subdirectory and other directories for other deliveries).

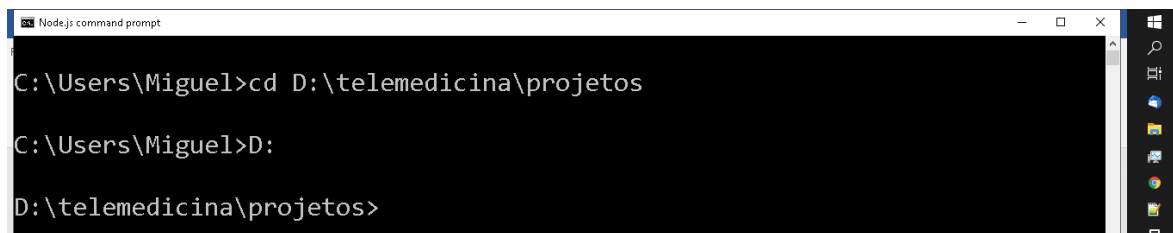
You can copy the path from windows explorer:



On the command line console write “cd ” (space after cd) and then left click with the mouse:

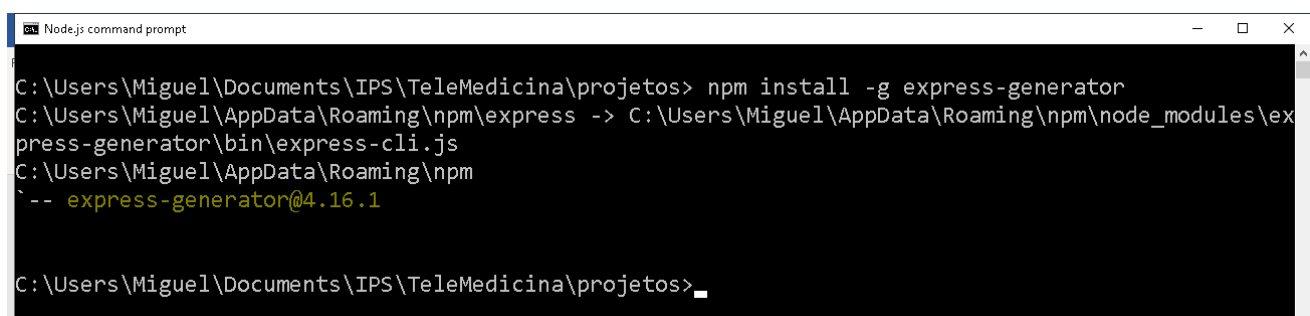


If your directory is on another drive you need to change drive. If the drive is D you write “D:”.



Lets install the express generator, this will allow you to create a project much faster. Run the command:
npm install -g express-generator

NPM is the Node Package Manager (used to install new modules and manage node projects)



The generator has many options, we can see them running “express -h”

```
C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos>express -h

Usage: express [options] [dir]

Options:

  --version          output the version number
  -e, --ejs          add ejs engine support
  --pug             add pug engine support
  --hbs             add handlebars engine support
  -H, --hogan       add hogan.js engine support
  -v, --view <engine> add view <engine> support (dust|ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
  --no-view         use static html instead of view engine
  -c, --css <engine> add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
  --git            add .gitignore
  -f, --force       force on non-empty directory
  -h, --help        output usage information

C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos>
```

We will now create a project called “alunos” by running
express --git --no-view alunos

A new directory called alunos will be created (you can also call the project code and get a folder named code, and do that inside a “alunos” directory so you can add other folders to the project):

```
C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos>express --git --no-view alunos

create : alunos\
create : alunos\public\
create : alunos\public\javascripts\
create : alunos\public\images\
create : alunos\public\stylesheets\
create : alunos\public\stylesheets\style.css
create : alunos\routes\
create : alunos\routes\index.js
create : alunos\routes\users.js
create : alunos\public\index.html
create : alunos\.gitignore
create : alunos\app.js
create : alunos\package.json
create : alunos\bin\
create : alunos\bin\www

change directory:
> cd alunos

install dependencies:
> npm install

run the app:
> SET DEBUG=alunos:* & npm start

C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos>
```

We will now work with the project using an editor.

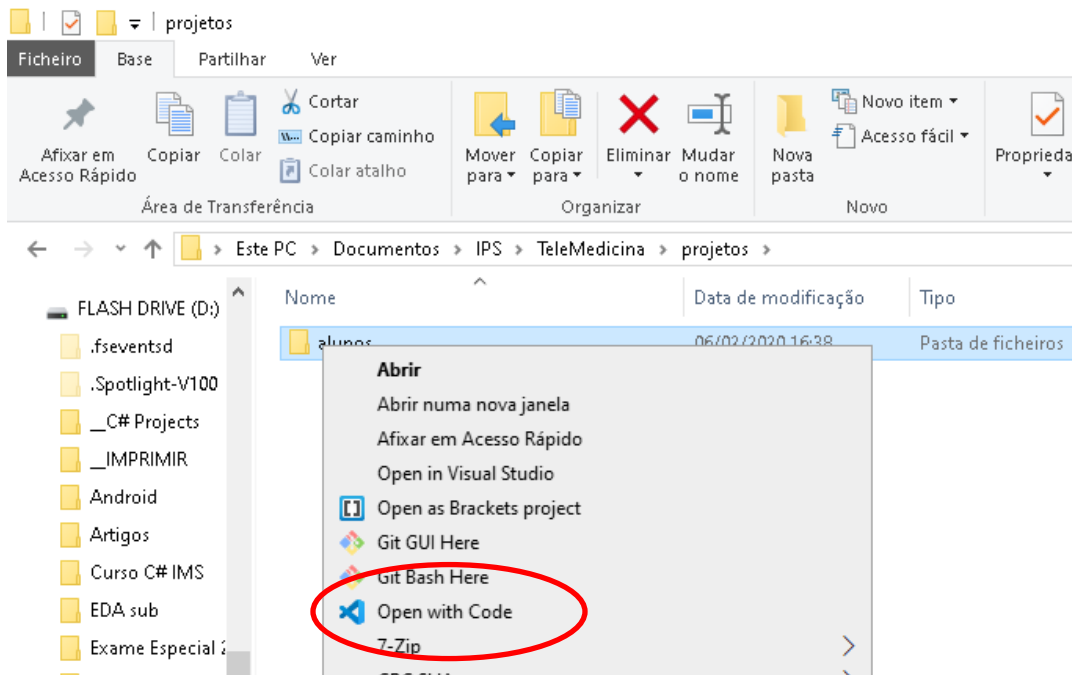
Installing and editor: VSCode

After creating the project, we only need to use an editor to make changes to it. The editor will include a terminal to run any commands we need.

Go to the site <https://code.visualstudio.com/>

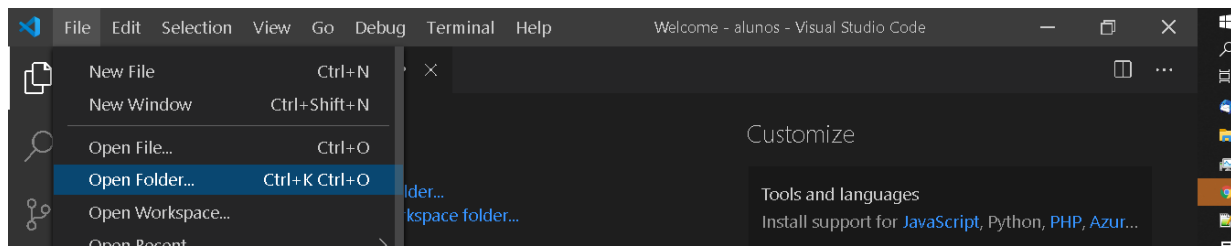
Download and install the file.

When you right click the directory, you should now have an option to open the directory with VSCode:



if the option to “Open with Code” does not show

Open VSCode from start menu and in the file menu choose “Open Folder”:

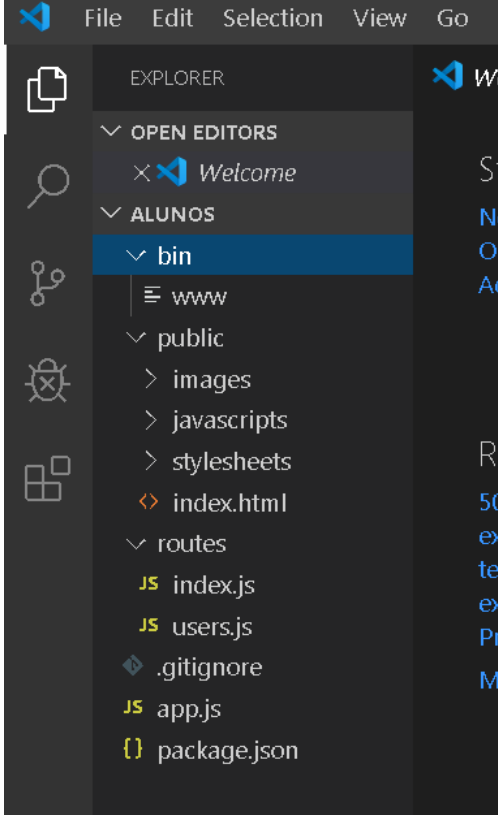


Then choose the folder of the project

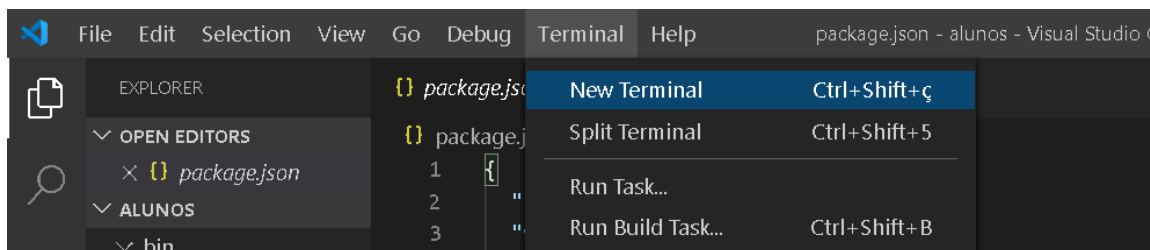
You can now see all the files and directories created for the project.

When clicked files will open on the right and you can edit them

The next table summarizes the main files and directories

	<p>The www file on the bin directory is the <u>script to start the server</u>. You should not change this file, although you can do it, for instance to directly change the default port where the server will be run (you will see in class).</p> <p>The app.js file is the file where all initial configurations are made, we will need to change this file for instance to include more files to process data (ex: students or classes data)</p> <p>The routes directory is where all the files to process data will be. This will correspond to the REST API Server described in the architecture.</p> <p>The public directory will hold all the HTML, CSS, javascript and other files (ex: images) for the web pages. This will correspond to the Web Pages Server described in the architecture.</p> <p>The package.json is where we save settings for our server like module dependencies or scripts (usually we do not need to change it directly)</p>
--	---

We will also need to use the terminal:



A Windows PowerShell or Cmd terminal will open on the bottom inside the directory of the project.

We can now install and run the server by using the Node Package Manager. Run the commands

npm install

npm start

install will install all modules defined in the **package.json** dependencies and sub-modules they need. A new directory called **node_modules** will be created that holds all the modules and sub-modules.

start will start the server using the script identified as start in **package.json** (that will call the **www** file that in turn will load all settings defined in the **app.js** file)

You can see the results of the commands in the next images

```
PS C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos> npm install
alunos@0.0.0 C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos
+-- cookie-parser@1.4.4
| +-- cookie@0.3.1
| `-- cookie-signature@1.0.6
+-- debug@2.6.9
| `-- ms@2.0.0
+-- express@4.16.4
| +-- accepts@1.3.7
| | +-- mime-types@2.1.26
| | | `-- mime-db@1.43.0
```

...

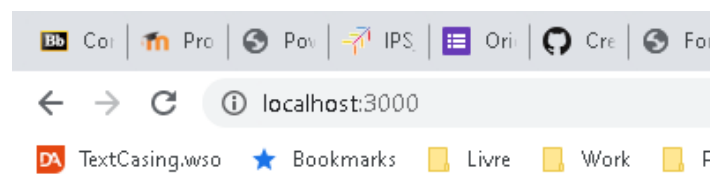
```
PS C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos> npm start

> alunos@0.0.0 start C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos
> node ./bin/www
```

The server is now running. You can see the result by writing localhost:3000 in a browser

localhost means you are running the server locally

3000 is the port where the server is running, you can change this by going to the www file and changing the default port (is not the correct way, but it is the easiest way).

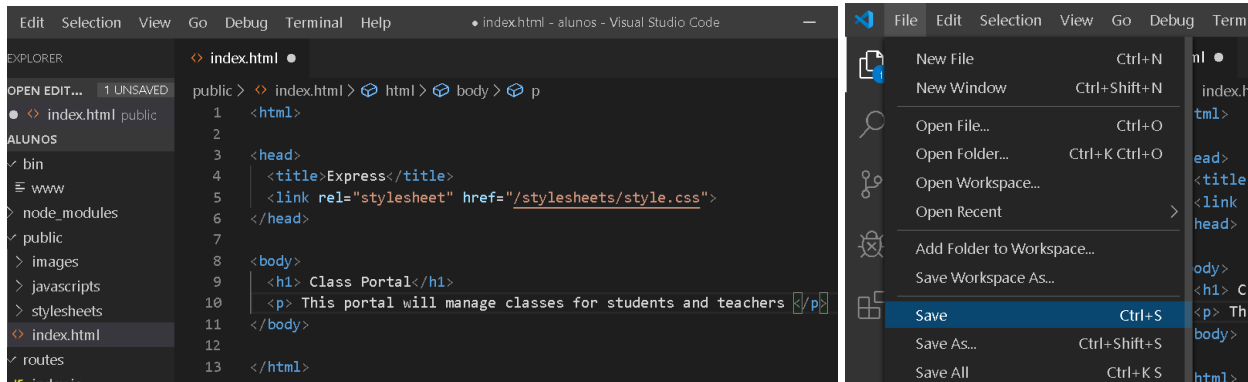


Express

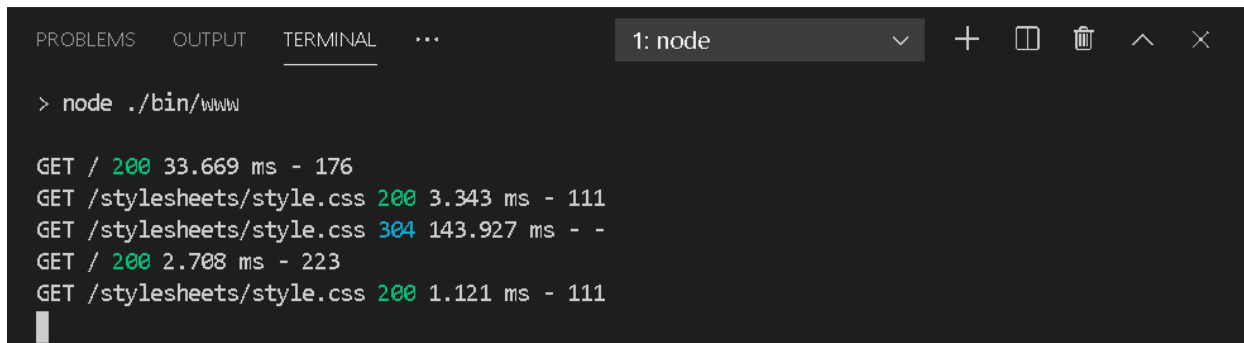
Welcome to Express

Now let's try to change something. Click on the `index.html` file in the `public` directory and change the text.

Save the changes and reload the `localhost:3000`




Notice the terminal below:



Each time you open or reload the site you will see lines being printing.

GET / means a request was made for the root of the website, that will correspond to the **index.html** (HTML files named `index` will be called when the root of the directory is written in the browser).

Since the HTML also refers to the **style.css** file this file will also be requested to the web server.

To stop the server you can use the kill terminal button  on the top right of the terminal, but that will close the terminal and you would need to open it again next time. You can also use **Ctrl+C** to stop the server (only works if you have the terminal window selected).

To start again you only need to run **npm start** again (no npm install, you only need to run it if you changed the dependencies on **package.json** or if you delete the **node_modules** directory, so for now you only run it once when you create the project).

Install the module to work with the MySQL database

Stop the server and run the following command:

npm install mysql --save --save-exact

This will install the module that will allow node.js to communicate with the database (we will see how this module works later).

The **--save** option will save the dependency on the **package.json** file, and it is indispensable to make sure this module is installed when we deploy to a hosting site.

The **--save-exact** option will make sure that the version of the module you use is not updated to a new version. This is not usually a problem for academic projects but if your web application code will be used for a long time the chances increase for a new version to crash your site. You can either correct it when it crashes or use this option and manage the versions updates yourself.

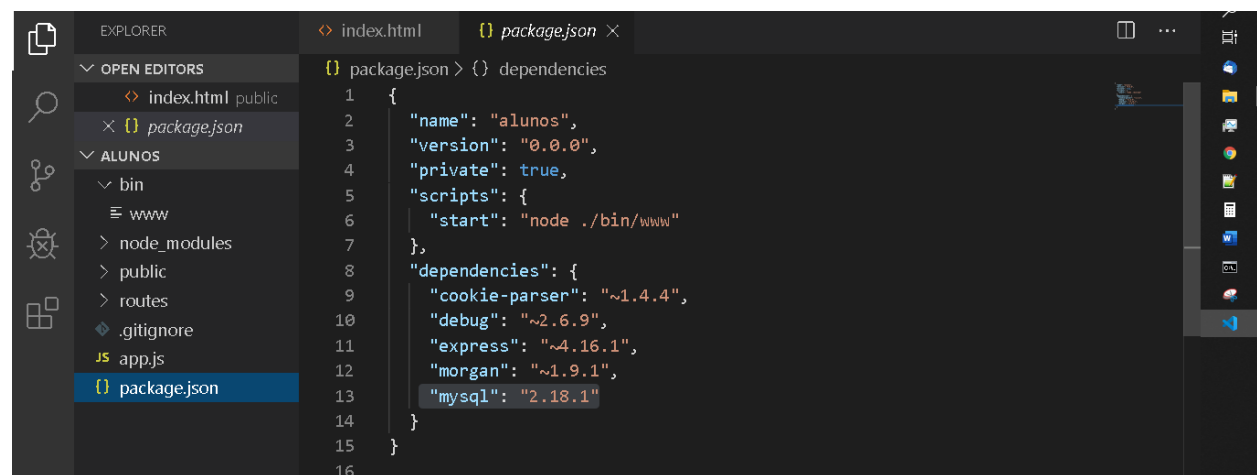
Result of installing the mysql module:

```
> node ./bin/www

Terminate batch job (Y/N)? y
PS C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos> npm install mysql --save --save-exact
alunos@0.0.0 C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos
`-- mysql@2.18.1
   +-- bignumber.js@9.0.0
   +-- readable-stream@2.3.7
      | +-- core-util-is@1.0.2
      | +-- isarray@1.0.0
      | +-- process-nextick-args@2.0.1
      | +-- string_decoder@1.1.1
      | `-- util-deprecate@1.0.2
   `-- sqlstring@2.3.1

PS C:\Users\Miguel\Documents\IPS\TeleMedicina\projetos\alunos> |
```

You can see that a new dependency was added to package.json



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows the project structure with 'package.json' selected. The main editor area displays the 'package.json' file. The 'dependencies' section has been updated to include 'mysql': '2.18.1'. The file content is as follows:

```
1 {
2   "name": "alunos",
3   "version": "0.0.0",
4   "private": true,
5   "scripts": {
6     "start": "node ./bin/www"
7   },
8   "dependencies": {
9     "cookie-parser": "~1.4.4",
10    "debug": "~2.6.9",
11    "express": "~4.16.1",
12    "morgan": "~1.9.1",
13    "mysql": "2.18.1"
14  }
15 }
```

We are now ready to place our web application online for the all World Wide Web to see.

Setting up a website: Github, Fork and Heroku

To host our web application on Heroku we first need to place our files on GitHub.

Setting up a GitHub repository

First create an account on GitHub: <https://github.com/join>

Fill the form (1st page) and on the next page (2nd page) choose the free plan. On the page after the plan choice (3rd page) you don't need to fill, you can just click to the next page.

You will need now to verify your email. Check your email and click on the Verify email link:

Almost done, @MiguelBugalho! To complete your GitHub sign up, we just need to verify your email address: miguel.bugalho.a@gmail.com.

[Verify email address](#)

It will open the page to create your first git project:

I will call it alunos, make it public and leave the other options empty:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

 MiguelBugalho ▾

Repository name *

alunos ✓

Great repository names are short and memorable. Need inspiration? How about [expert-octo-chainsaw?](#)

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

The repository is now created and has its own link (example: <https://github.com/MiguelBugalho/alunos>)

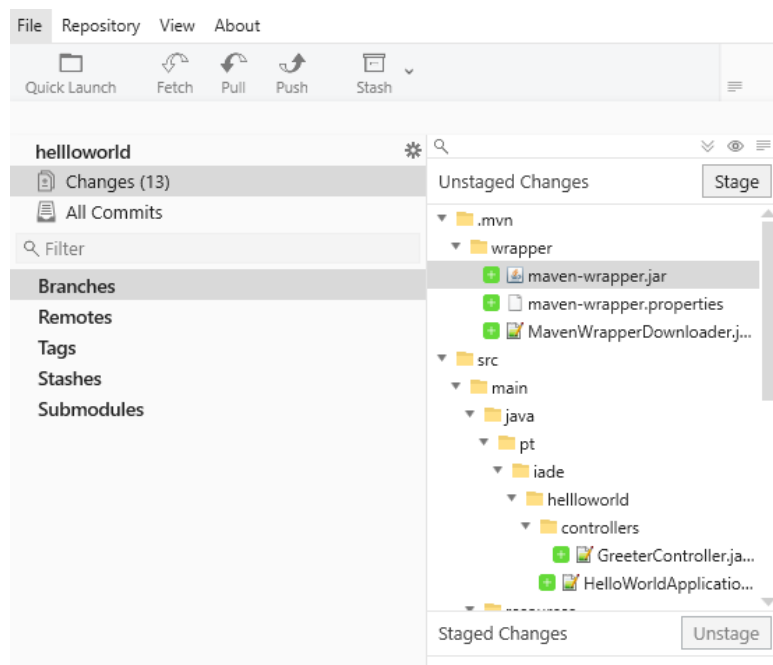
Using fork to manage git and github

Install fork from <https://git-fork.com/>

Now, open fork application. Go to **File/Init New Repository**. You can now choose the folder of your project.

NOTE: For the course project you will need to create a folder that will have the project application folder and other folders with other deliverables (ex: reports, documentation, database, etc), and should use this root folder instead of only the project code folder.

When the name of the project shows on the top left corner you can click on the option “Changes” bellow and will see something like what is shown bellow (the example is from another type of project, but the commands shown are valid for any source code):



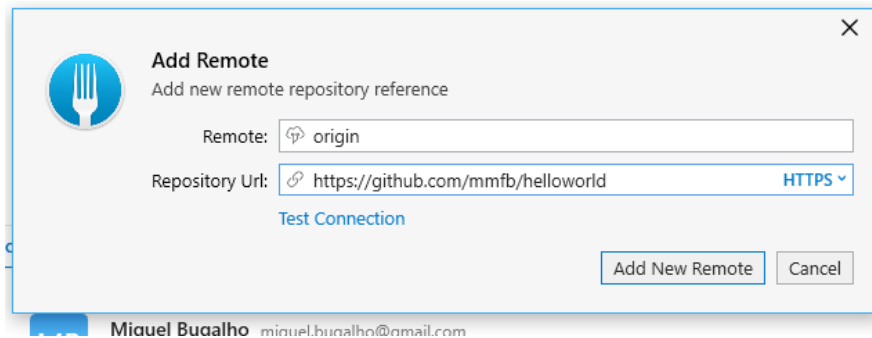
Click on each directory and file and on the **Stage** button above so that you pass everything from **Unstaged Changes** to **Staged Changes**.

Write a commit message like **Initial Commit** on the bottom right and click on **Commit X Files**.

You have now created your first git repository, but for now it is only local. On the left you can see a “master Branch” was created, and you can click there and track all the history of that repository branch.

We will talk more about Git later, but first we need to connect this local repository with our remote repository on GitHub.

Right click on **Remotes** (bellow **Branches**) and choose **Add Remote**. On the Repository Url you need to copy the URL of your GitHub repository and then click **Add New Repository** (see on next page an example). A new **Remote** was created with the name **origin**.



Now go to the Repository menu on the top bar and choose **Push** to send the files to the remote repository. It automatically will choose your only branch and your only remote. Click **Push**.

And that is it, if you refresh your GitHub remote repository your code should be there.

Any modifications you make you can go back to **Changes** and make a new **Commit**. When the modifications are already stable you can then **Push** those commits to the GitHub remote server. This not only serves as a backup, as it keeps all the versions of the code you did, making sure you can backtrack to a previous version if you need.

There is much more to know about Git and GitHub, but for an introductory tutorial we are now done, but you are welcome to make your own experiences.

Setting up a Heroku application

First you need to create an account on Heroku: <https://www.heroku.com/>

Choose signup for free and fill the form. Create the free account.

Go to your email to activate your account and click on the link to accept:



Thanks for signing up with Heroku! You must follow this link to activate your account:

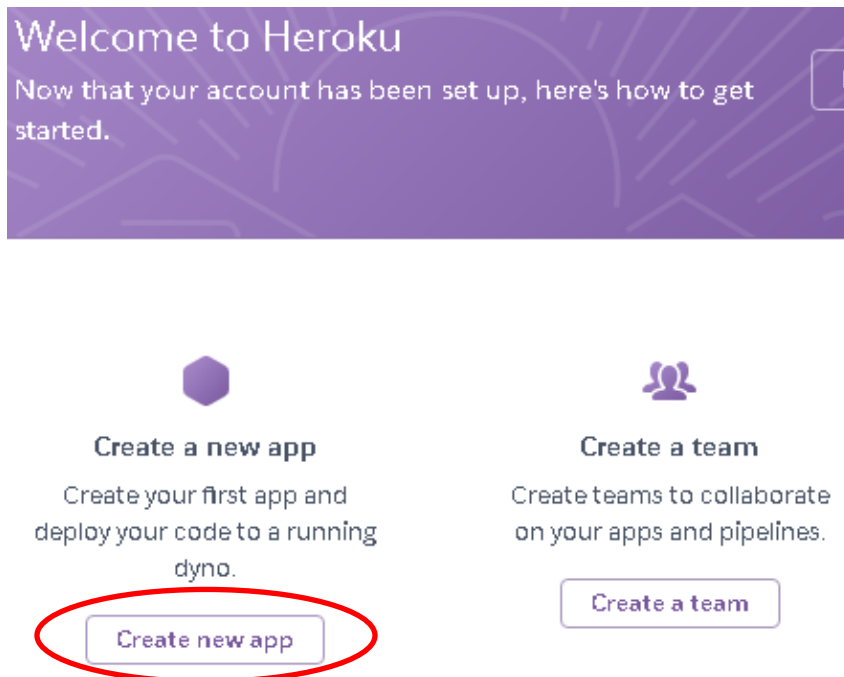
<https://id.heroku.com/account/accept/7291714/9be9c1f27ad08470b53d71664b3bdefb>

Set up your password and login.

Use your email and password to login

(if the login page does not open use this link: <https://id.heroku.com/login>)

Choose create a new app



Choose an app name. Each app as its own name since it will correspond to its own web site, so you cannot use the same name as your colleagues (or your teacher).

App name

✓

alunos-mb is available

Choose a region

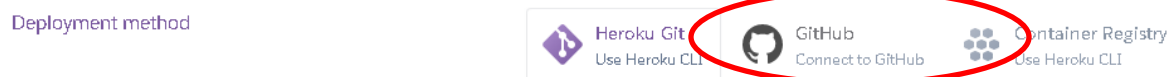
Europe

Add to pipeline...

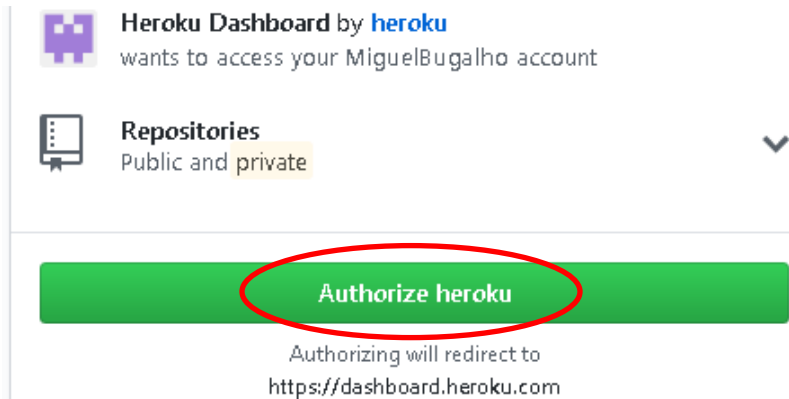
Create app

Now you are on the deploy page, you have other options (tabs above), but this is the only one we will really need for setting up the web application.

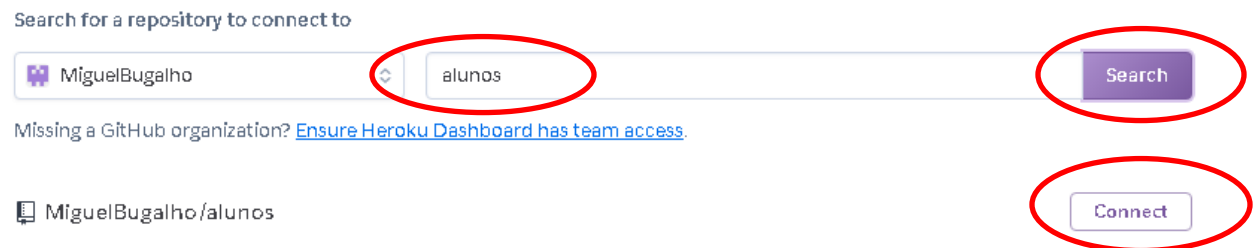
Ignore the pipeline configuration and on Deployment method choose GitHub



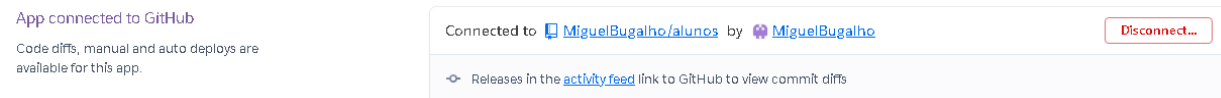
Click on connect to GitHub, if you are logged to your GitHub account it will only ask you to authorize Heroku, if not it will ask you to login and then ask you to authorize Heroku.



Fill the repository name click search and choose Connect on the repository



Your app is now connected to your GitHub repository.

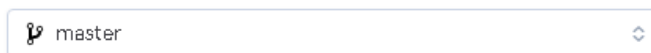


On Automatic deploys click on Enable Automatic Deploys. This way each time you update your repository Heroku will update your web application.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more.](#)

Choose a branch to deploy



☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Finally on Manual Deploy click on deploy branch, this is only needed the first time, after that the automatic deploy will take care of deploys, but you can always come here and make a manual deploy if you think something failed on the deploy.

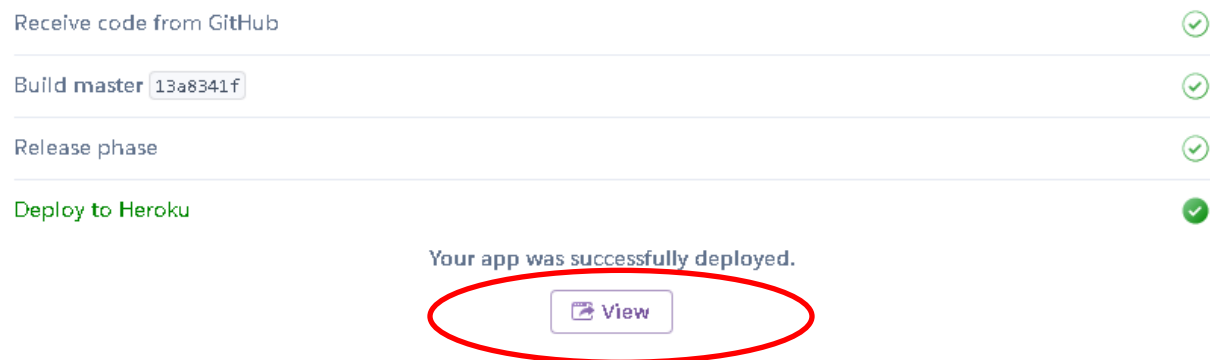
Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

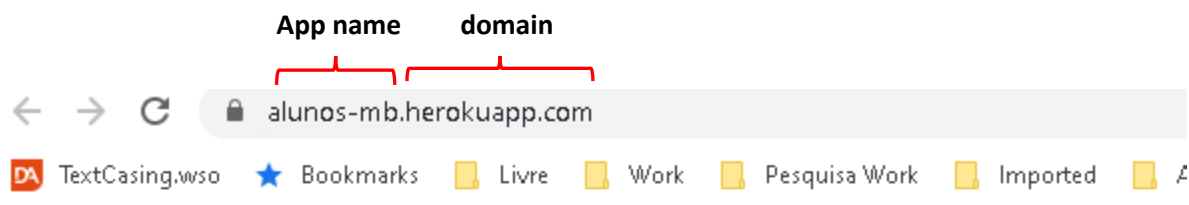
Choose a branch to deploy



It will take some seconds until you see the button to open your app:



Its done, your Web Application is online to everyone to see!



Class Portal

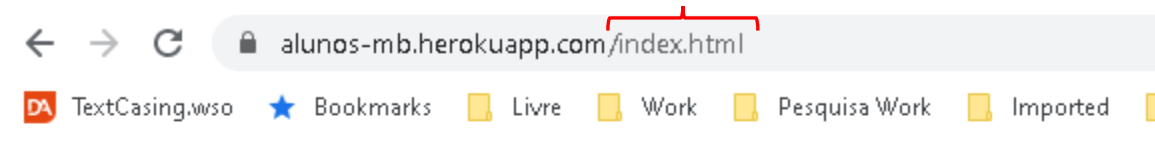
This portal will manage classes for students and teachers

Notice the URL on top. **herokuapp.com** is the domain name that Heroku provides for its clients, and the name of your application (in this case **alunos-mb**) that distinguishes it from any other.

Notice that the URL does not include the port (:3000) like in our localhost example, Heroku already configures the application to use the “normal” port (in this case is the HTTPs port, since the locker symbol is been shown)

The site is showing **the index.html** file like the example before but we can access more files by adding a path after the domain name (only the ones on **public** directory: not many files to access for now).

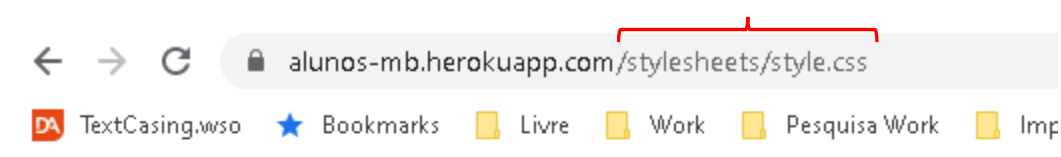
Path to index.html file



Class Portal

This portal will manage classes for students and teachers

Path to style.css file



```
body {  
  padding: 50px;  
  font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;  
}  
  
a {  
  color: #00B7FF;  
}
```