



Web Scraping System

03.14.2023 - 03.21.2023

Antonio Martínez

Qualex Consulting Services, INC

antonio.martinez@qlx.com

Software Developer

Overview

This is a web application that aims to collect information from other pages, to offer market analysis services. This web application is based on React for the front-end and nodeJS for the data extraction in the backend. The database used for the demo is mongoDB, but it is easily reproducible in relational databases such as MySQL. The really important thing is how the information should be received and delivered to the API.

Goals

1. Collect real-time data from the web pages of competitors of a certain brand.
2. Show and compare the same products, or similar ones, to establish relationships.
3. Filter the data under specific parameters such as: quantity, size, product.
4. Store the data of the searches and/or scans of web pages
5. Generate Reports
6. Make the system scalable, so that new products from the pages to be reviewed can be added.

Front-end Specifications

A brief list of the technical specifications:

Node JS v.16.15.0

It is necessary that the **node js version used is exactly 16.15.0**, due to the compatibility of the libraries used in the project. Any other later version will cause problems. A solution is to use the NVM version manager if you want to have several versions of node js installed. Or simply install the required version globally.

React v.17.0.2

React 17 enables gradual React upgrades, which means you can update parts of your app that use React without affecting the other parts. Supports Fast Refresh, which is a new feature that lets you edit React components in a running application without losing their state or unmounting them

Special libraries:

```
npm install chart.js react-chartjs-2
```

```
npm install chartjs-plugin-datalabels --save
```

```
npm install --save react-html-table-to-excel --legacy-peer-deps
```

Backend-end with Node, Specifications

Node JS v.16.15.0

Node Express v.4.18.2:

Node express is a web application framework for Node.js that allows you to create APIs and web servers in a simple and clean way. It is a lightweight package that does not hide the core Node.js features. It offers various features such as routing, middleware, template engines, and performance optimization. It is the most popular Node web framework and is used by many other Node web frameworks.

Cheerio v.1.0.0-rc.12 and Puppeteer v.19.7.3:

These are the libraries that make web scraping possible, one of them works for static pages or pages generated from a backend server, and the other one for pages that are generated dynamically with javascript code.

Special libraries:

```
npm install -g nodemon
```

```
npm install express cors mongoose dotenv
```

```
npm install mongoose-bcrypt
```

```
npm install express body-parser mongoose --save
```

```
npm install accounting-js
```

```
npm install flat
```

```
npm install --save request request-promise cheerio puppeteer
```

Backend-end with Python, Specifications

Python v.3.10.2

Flask v.2.1.2:

Flask 2.1.2 is a lightweight web application framework for Python that allows you to create quick and easy web applications that can scale up to complex ones. It uses Werkzeug and Jinja as its core components and offers various features such as routing, templating, testing, debugging, and more.

Special libraries:

```
pip install --user pipenv
```

MongoDB (cloud version) as Database for demo:

MongoDB Cloud Atlas is a fully managed, multi-cloud database service that lets you run MongoDB applications anywhere in the world. It offers a flexible JSON-like data model, rich querying and indexing, and elastic scalability. It also automates database administration tasks such as backup, security, and monitoring. You can deploy MongoDB Cloud Atlas on AWS, Azure, or Google Cloud and enjoy the benefits of a cloud-native database platform.

Installation Front-end:

Clone or download this private repository:

[metantonio/mikes-cigar: frontend \(github.com\)](https://github.com/metantonio/mikes-cigar)

Requirements:

Make sure you are using **node v.16.15.0**

Go to the root directory.

1. Install the packages:

```
npm ci
```

```
npm install chart.js react-chartjs-2
```

```
npm install chartjs-plugin-datalabels --save
```

```
npm install --save react-html-table-to-excel --legacy-peer-deps
```

2. Create a .env file:

```
cp .env.example .env
```

3. Configure the .env file:

You must configure "BASE_URL" that is the variable for backend IP and PORT with node js.

You must configure "BASE_URL2" that is the variable for backend IP and PORT with python. (Depending what you are doing, this API could be optional).

You must configure "BASE_URL_PUBLIC" that is variable to frontend server IP and PORT. If you only need work on your own pc, you can use localhost, instead of IP.

Example:

```
BASENAME=/
BASE_URL = "http://10.1.10.144:5001"
BASE_URL2 = "http://localhost:3340"
BASE_URL_PUBLIC = "10.1.10.144:3001"
```

If for some reason you must change PORT of **front-end (default is 3001)**, you have to change it at the following files:

.env: change BASE_URL_PUBLIC

Webpack.dev.js: change variable port.

Package.json: at "scripts" change the "start" command port argument.

4. Start the webpack dev server with live reload, for windows, mac, linux or Gitpod:

npm run start

Installation Back-end:

Clone or download this private repository:

[metantonio/backend-webscraping \(github.com\)](https://github.com/metantonio/backend-webscraping)

Requirements for Node JS Backend:

Make sure you are using **node v.16.15.0**: [node 16.15.0](#)

Go to the root directory.

1. Install the packages:
 - 1.1.1. **npm ci**
 - 1.1.2. **npm install -g nodemon**
 - 1.1.3. **npm install express cors mongoose dotenv**
 - 1.1.4. **npm install mongoose-bcrypt**
 - 1.1.5. **npm install express body-parser mongoose --save**
 - 1.1.6. **npm install accounting-js**
 - 1.1.7. **npm install flat**
 - 1.1.8. **npm install --save request request-promise cheerio puppeteer**
 - 1.1.9.
2. Create a .env file:
 - 2.1.1. **cp .env.example .env**
3. Configure the .env file:

You can configure "SALT " and "SEED " that are the variables for password encryption, but I recommend **changing it after configuring backend basic functionality**.

You must configure "BASE_URL" that is the variable for backend IP and PORT with node js. Must of the time it should be with localhost, if you don't want the API be exposed to world

You must configure "TOKEN_EXPIRE ", which is the timer in seconds for a user's session. Defaults are 36000 seconds, that is 10 hours.

You must configure "ATLAS_URI", which is the string connection to mongoDB cloud, you must provide username, database auto generated password (not the same as your login password), cluster key, and collection name.

You must configure "RESPALDO", which is the variable that refers to the absolute PATH to folder BD_Prueba. This folder contains the necessary files to populate the database of mongoDB to have an administrator user able to modify the system for the first time.

You must configure "FLASK_APP_KEY" that is the variable for JSON WEB TOKEN encryption, I recommend changing it for a **hash-256 string in deployment**.

You must have a GMAIL account with "password for applications" option available. In order to send emails (necessary for create new users in the system and change password), you have to configure "EMAIL_USERNAME" (example: antonio.martinez@qlx.com), and "EMAIL_PASSWORD". This password is not the same as a gmail account, is generated only for mail applications and only will be able to send emails not read inbox).

Example:

```

.env
1 ATLAS_URI=mongodb+srv://antonio:antonio@cluster0.mongodb.net/?retryWrites=true&w=majority
2 PORT = 5001
3 BASE_URL = "http://localhost:5001"
4 SALT = "aquiVaElSalt"
5 TOKEN_EXPIRE= 36000
6 SEED = "EcoGreen-Desarrollo"
7
8 EMAIL_HOST = "smtp.gmail.com"
9 EMAIL_USERNAME = "antonio.martinez@qlx.com"
10 EMAIL_PORT = 465
11 EMAIL_PASSWORD = " "
12 FECHA_MINIMA = "01/01/1970"
13
14 DB_CONNECTION_STRING=mysql+mysqlconnector://root@localhost/example
15 FLASK_APP_KEY="any key works"
16 FLASK_APP=src/main.py
17 FLASK_ENV=development
18 RESPALDO = "C:/Users/antonio/Documents/Antonio/Boilerplates/BD_Prueba"

```

4. Start the webpack dev server with live reload, for windows, mac, linux or Gitpod:

npm run start

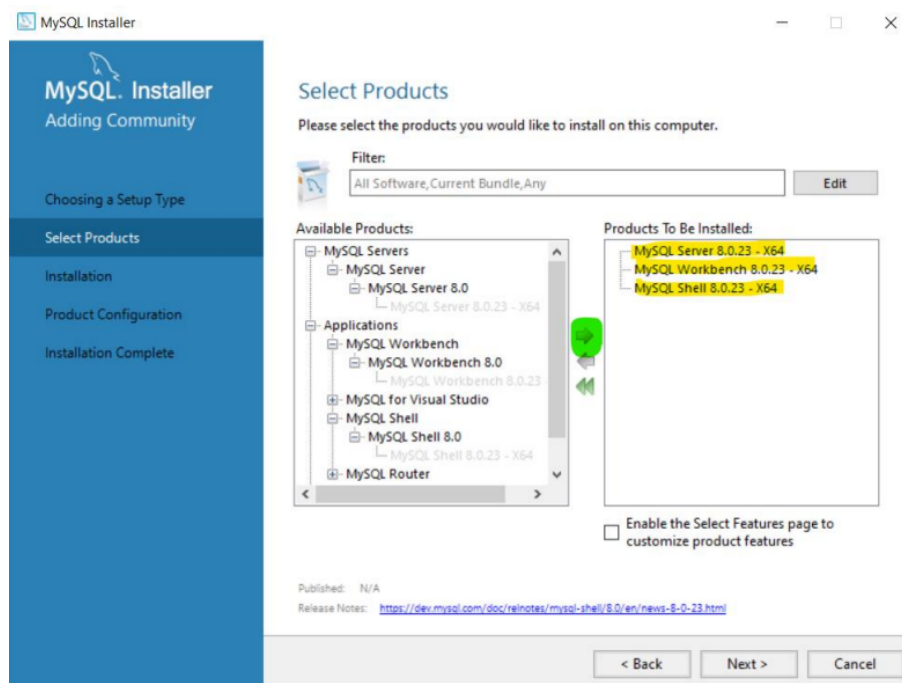
Requirements for Python - Flask Backend:

Everything inside the ./src/ folder is associated with the backend models and routes with Flask.

Make sure you are using **Python 3.10.2** : [Official Releases](#)

Go to the root directory.

1. Install pipenv:
 - a. **pip install --user pipenv**
2. Installation of relational databases for demo, so here i will explain installation for MySQL, but it's similar for PostgreSQL, etc.
 - a. Download MySQL: [MySQL :: Download MySQL Installer](#) and choose the full version of **430MB**. **Open the installer.**
 - b. Choosing a setup type: **Custom**. Next
 - c. Select the following products and click Next:



- d. If there is another requirement, the installer will try to download it. Click **EXECUTE** to install MySQL selected products.

- e. Click Next until you get to the “**Type and Networking**” step. Select **development computer**.

MySQL Installer

MySQL Server 8.0.23

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

☒ TCP/IP Port: 3306 X Protocol Port: 33060

☒ Open Windows Firewall ports for network access

☐ Named Pipe Pipe Name: MYSQL

☐ Shared Memory Memory Name: MYSQL

Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

☐ Show Advanced and Logging Options

Next > Cancel

- f. Choose the **Strong Password** option, and click Next.
- g. At this point we must create 2 types of users. One of them will be the **ROOT** user and we must provide a password (save it!!!):

MySQL Installer

MySQL Server 8.0.32

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Server File Permissions

Apply Configuration

Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password strength: Weak

MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

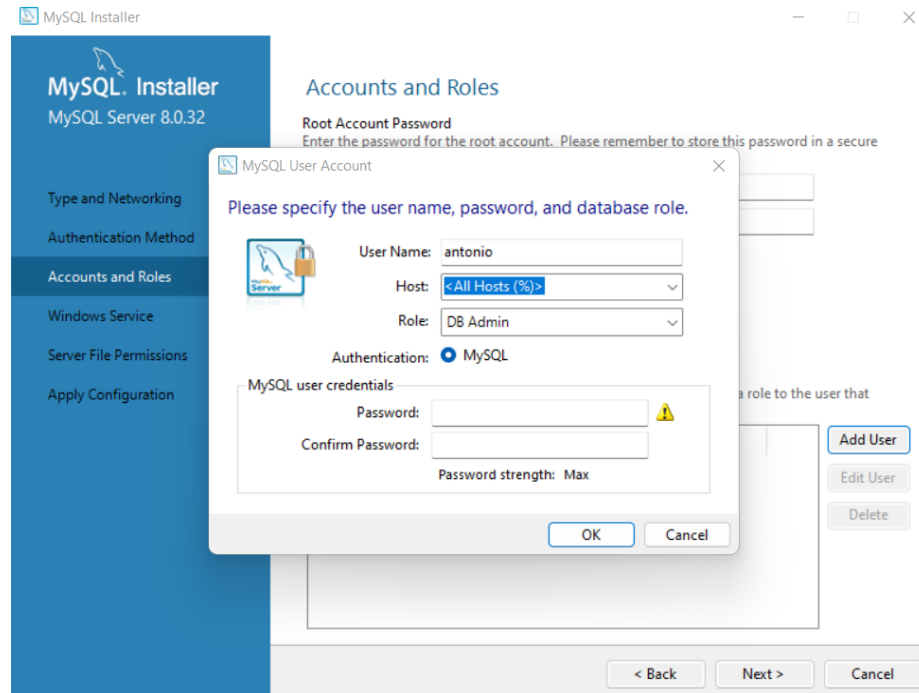
Add User

Edit User

Delete

< Back Next > Cancel

- h. Now click “Add User” and add a user without password:



- i. Click Next, and then Execute to finish installation.
 j. Close all terminals and MySQL Workbench.
 k. Open a new terminal and navigate to the installation folder:
cd C:\Program Files\MySQL\MySQL Server 8.0\bin
 l. Now let's configure user, change “user” by your username used on installation:

mysql -h localhost -u user

- m. If everything is correct you will enter into mysql command line. It's time to configure environment variables and add to Path the installation folder, to make this easier on Windows.
 n. Every time that you want enter with your “user” you will have to do:
mysql -h localhost -u user
 o. Every time that you want enter with root user, you will have to do:
mysql -h localhost -u root -p
 And provide a root password.
 p. At the root folder of this project, configure “DB_CONNECTION_STRING” variable inside of .env specify if user if root or another, and database name, by default database name is “example”:

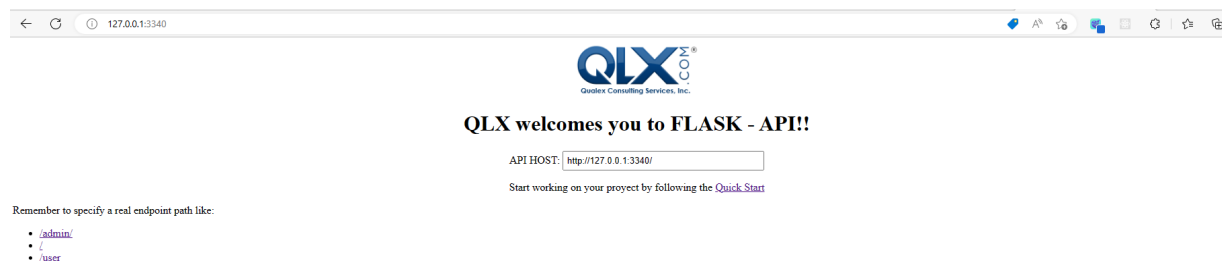
DB_CONNECTION_STRING=mysql+mysqlconnector://root:password@localhost/example

3. Enable in the Windows environment variables, the pipenv command. To do this, open the windows runner (Run) and place: **%AppData%**

- a. Once the AppData folder is opened, navigate to:
\Roaming\Python\Python310\Scripts, where we will see the **pipenv.py**
 - b. Open the Environment Variables window, in the user environment variables we must look for the Path property, and then add the address of the folder where the Python scripts are located. Close every opened command line and Visual Studio Code, open command line again and this command should be recognize: **pipenv**
4. At the root of this backend project, install all the python libraries into the virtual environment with:
- a. **pipenv install**
5. I created 3 scripts to **reset the database** from scratch for different databases. For MySQL database (it will ask for password):
- a. With Bash: **pipenv run reset_db_sql**
 - b. With command line: **pipenv run reset_db_sql_win**
 - c. Or do it manually if don't have bash, cmd or wsl installed:
 - i. **pipenv shell**
 - ii. Delete "migrations" folder if it exists.
 - iii. On the terminal: **pipenv run init**
 - iv. **mysql -u root -p -e "DROP DATABASE example;"**
 - v. **mysql -u root -p -e "CREATE DATABASE example;"**
 - vi. **pipenv run migrate**
 - vii. **pipenv run upgrade**
6. Execute Flask-Python server from virtual environment:
- a. **pipenv shell**
 - b. **pipenv run start**
7. You can check database live at:

<http://localhost:3340/>

Remember that every time that you change models definition (add, edit, remove), you must stop the server and repeat step 5. **IT WILL ERASE DATABASE.**

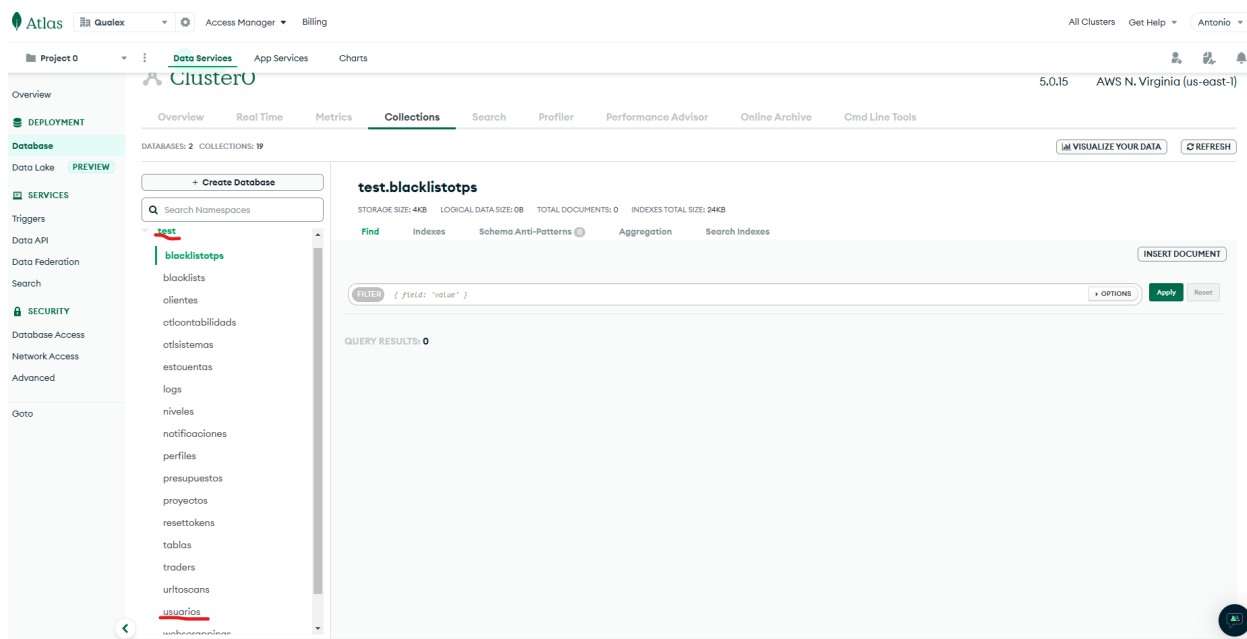


Configuration for first time use:

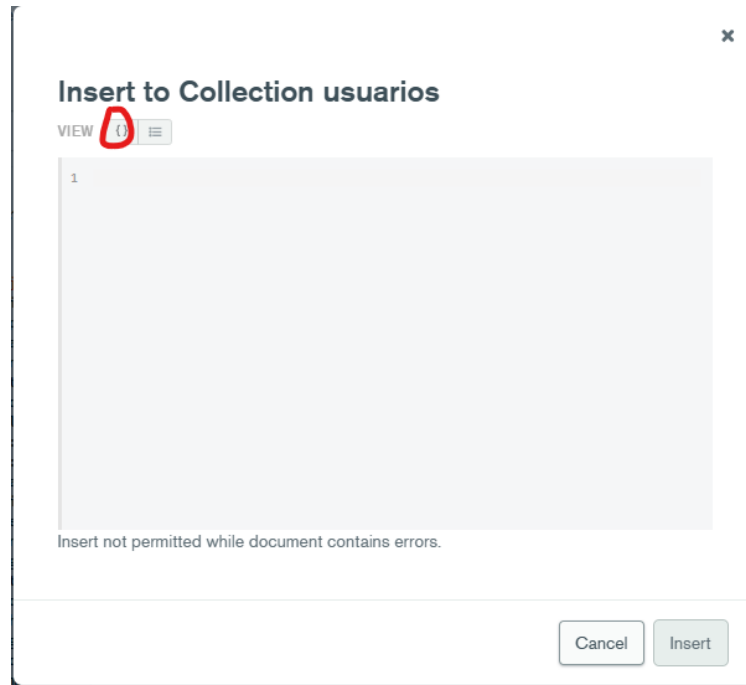
If you want to connect to **an existing mongoDB**, you don't need to do any of the **following steps**. Just ask for the **ATLAS_URI** string and copy it into the **.env** of the backend.

Configure the first valid user:

Once you have a database created (in the example you can see my database name is "test"), you can check your collections (like tables in SQL), if a collection named **"usuarios"** (referred to users) is not created, **add it**.

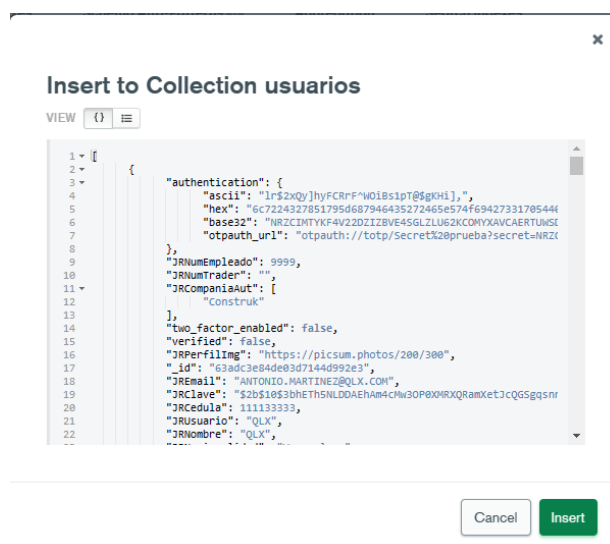


Enter to "usuarios" collection, and press the button "Insert Document", click into json view, and erase everything:



You must enter the first valid user in the system, for that, check the root folder of the backend, and go to the file at: **./BD_PRUEBA/UsuarioPrueba123.json**

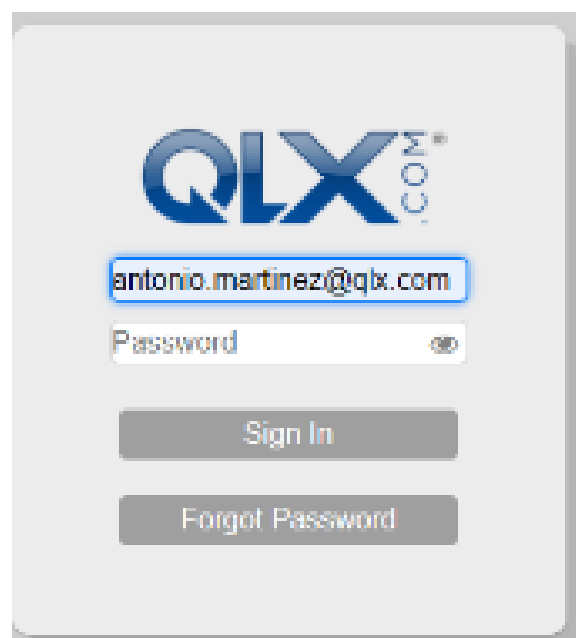
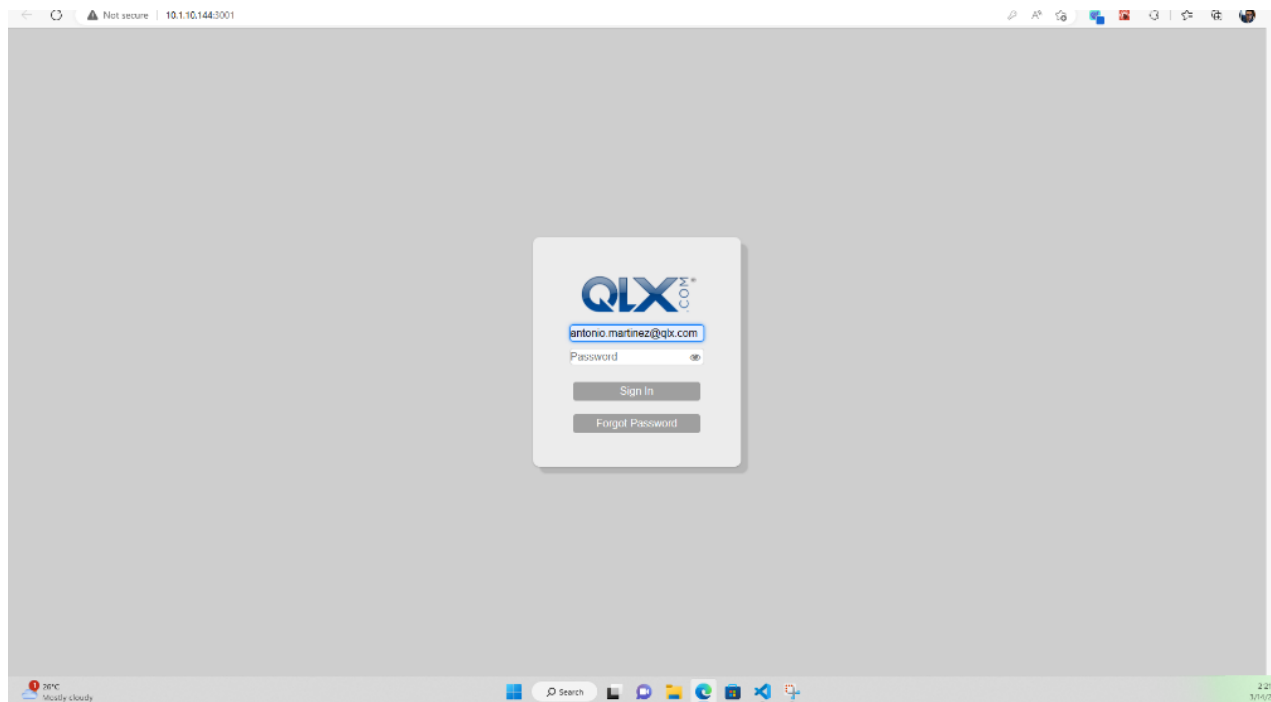
Once the json file is opened, copy all the content, and paste it into the mongoDB Atlas collection, and change **JREmail** if is needed:



The user will be **QLX** and the password will be **123**

The password is encrypted with the **default SEED and SALT** that comes with the .env.example file.

Still, if SEED and SALT environment variables were changed at this moment, and if email variables with Gmail are configured, you can ask **"forgot password"** using the email address, and not the username, that were used on JREmail field:



This will return an email to your inbox with the new password, encrypted with custom SALT and SEED.

Note that if you are using MongoDB Community to have a local mongoDB in your PC, and MongoDB Compass to manage it and visualize, you can create an "ATLAS_URI" here, and insert documents to collections in the same way as atlas cloud.

Configure the system tables and users profiles:

Repeat the steps used to configure the first user but, in this case, check for collections named "ctlsistemas", "tablas" and "perfiles", if they do not exist then create them.

Insert document as json for collection "ctlsistemas" with the file at:

./BD_PRUEBA/Sistema.json

Insert document as json for collection "tablas" with the file at: **./BD_PRUEBA/Tablas.json**

Insert document as json for collection "perfiles" with the file at: **./BD_PRUEBA/Perfiles.json**

Navigating through the system:

Open website

The frontend server should be running at port 3001 by default. Just navigate to the BASE_URL_PUBLIC on any browser:

Example: <http://10.1.10.144:3001>

Login

If everything went perfect, you should be able to log in into the website.

Username: **QLX**

Password: **123**

Basic Features:

Some of the basic features of this internal system are:

Creation of Users

Some user's profiles will have the ability to create new users. People can't come and create a user by themselves. **For this to work, SMTP should be configured in backend**, it means that GMAIL account and GMAIL password for applications should be configured in the **.env** file

Navigate to: **System > Security > Register users**

Log of users

System is able to register the users movement through the system, always that middleware is configured on every request at the backend.

Example of middleware in the **node backend** to register when user log out of the system:

```
router.get('/logout', [verifyToken, logger], (req, res) => {
  let token = req.get('Authorization');
  //console.log('entró en el logout')
  let blacklist = new BlackList({
    token: token
  })
```

This registration can be used as metrics, to know how many requests are made to the API, in order to calculate prices of AWS, etc...

Navigate to: **System > Security > Log Admin**

usuario	fecha	mes	año	endpoint		
Usuario asc	Fecha asc	IP	Endpoint asc	Mes(número)	Año	
ninguno	9/3/2023, 12:16:08 p. m.	localhost	/usuarios/registerjwt	3	2023	
qlx	9/3/2023, 12:16:34 p. m.	10.1.10.140	/usuarios/login	3	2023	
QLX	9/3/2023, 12:16:39 p. m.	10.1.10.140	/usuarios/login	3	2023	
QLX	9/3/2023, 12:43:33 p. m.	10.1.10.140	/usuarios/login	3	2023	
QLX	9/3/2023, 12:43:34 p. m.	10.1.10.140	/usuarios/check	3	2023	
QLX	9/3/2023, 12:43:34 p. m.	10.1.10.140	/usuarios/check	3	2023	

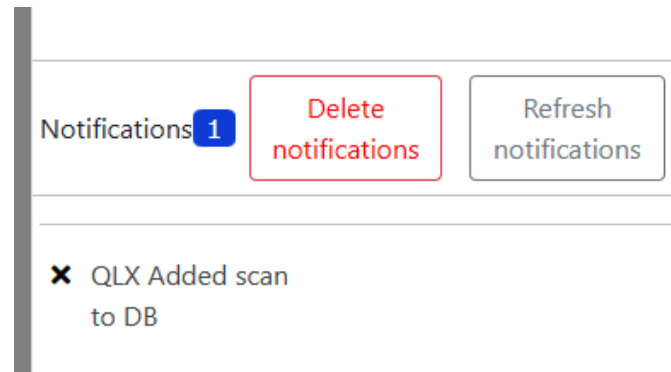
Change password and email it to the user

System is able to change the password with the existing **SEED** and **SALT**, and mail it to the previous user email registered in the system.

Notification System

System is capable of showing notifications to specific users, or just alerting any user of changes in the system that you want to be alerted to.

Users with Administrator profiles are **able to see every notification in the system**.



Permission System

You can generate any number of permissions, with description. But still they have to be added to one or many profiles. Then, you can add a profile to an user, and custom permissions for that user.

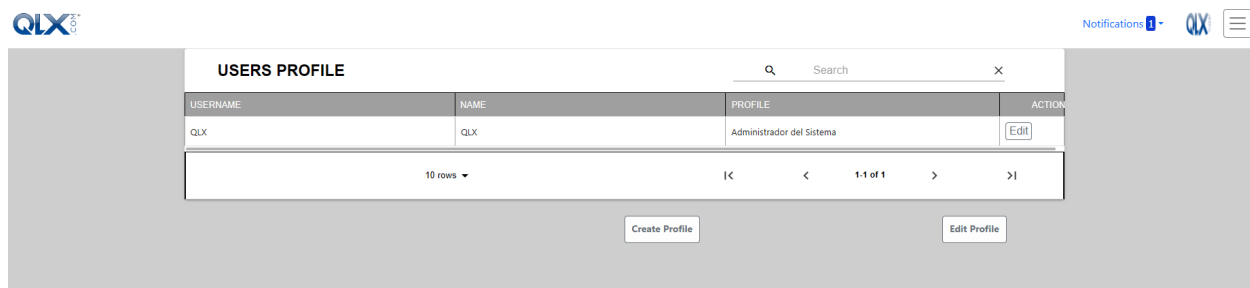
Still, permission creation must have support in the frontend code. So even if the permission creation can be parameterized in the backend, in the frontend you must ask for that permission at the time of user login and at the time to make a specific action. The backend will check if the user has permission and then it will send the requested data.

To create permissions go to:

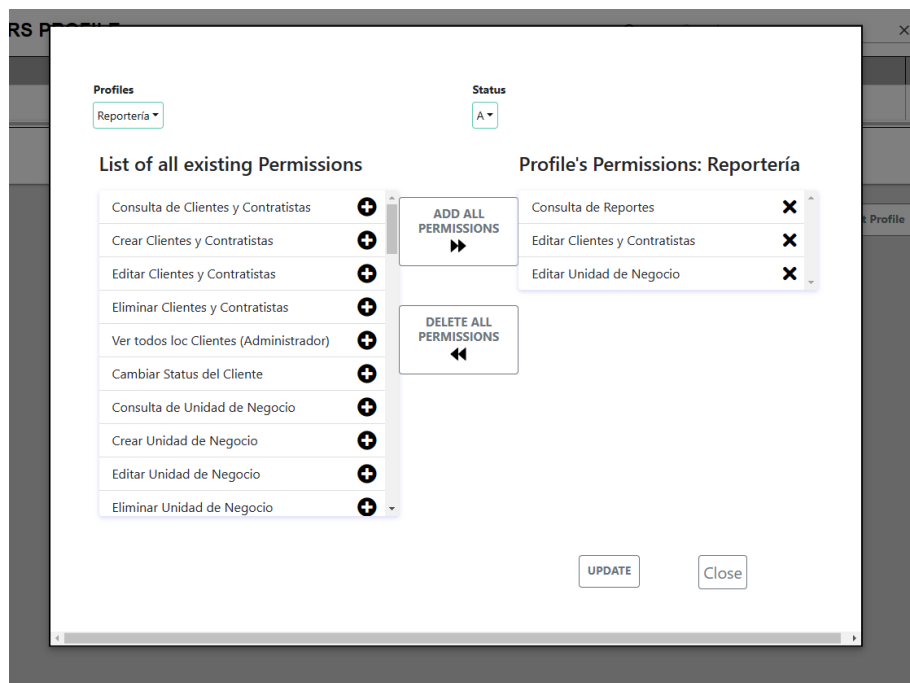
System > Security > Permission Table

System for User's profiles permission

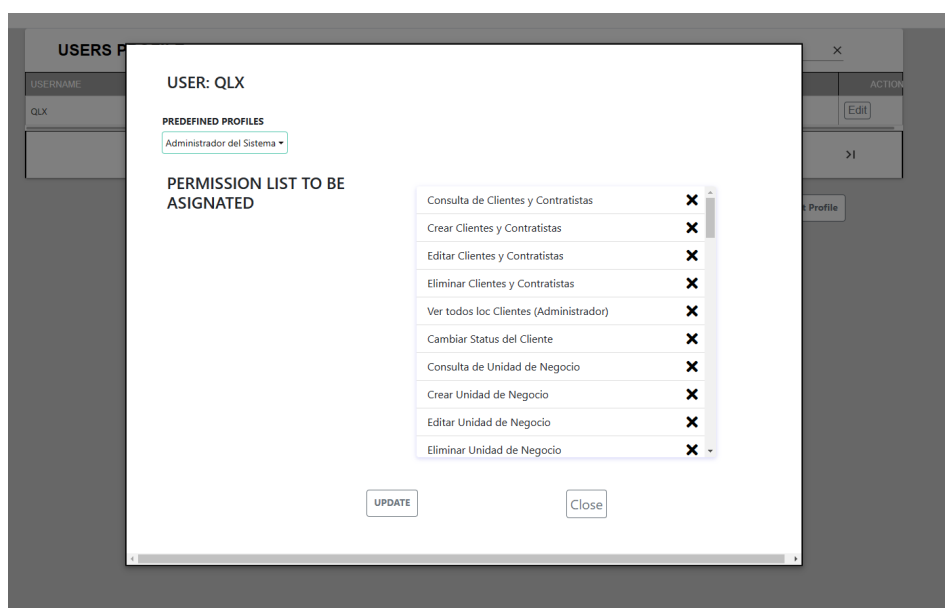
In this section you can create a new profile. It will be without permissions.



Then you can edit the created profile, adding to it the necessary permissions. Remember me change the descriptions of every permission to english.



Finally, you can add a profile to a user. **But, if you are testing the system, don't change the profile of QLX username.** It's better to create a new user for testing.



To edit profiles or user's profile go to:

System > Security > User Profiles

List of Users

If you have Administrator Privileges, then you will be able to see the list of users (password will not be visible, even if that would be the case, they are encrypted with SEED and mixed with SALT, so are pretty useless in that form without SEED, SALT and know how reverse the encryption algorithm).

The list can be seen as a table view and as a card view.

To see users list go to:

System > Security > List of Users

Emergency Button

The Emergency button is silenced, only users with administrator privileges can use it. When it is activated, there's a view where you can erase all permissions of every registered user. This is in case that you need to stop the system to prevent abuse.

The only way to restart the system is to manually insert into the database an Administrator user.

To see users list go to:

System > Security > Emergency

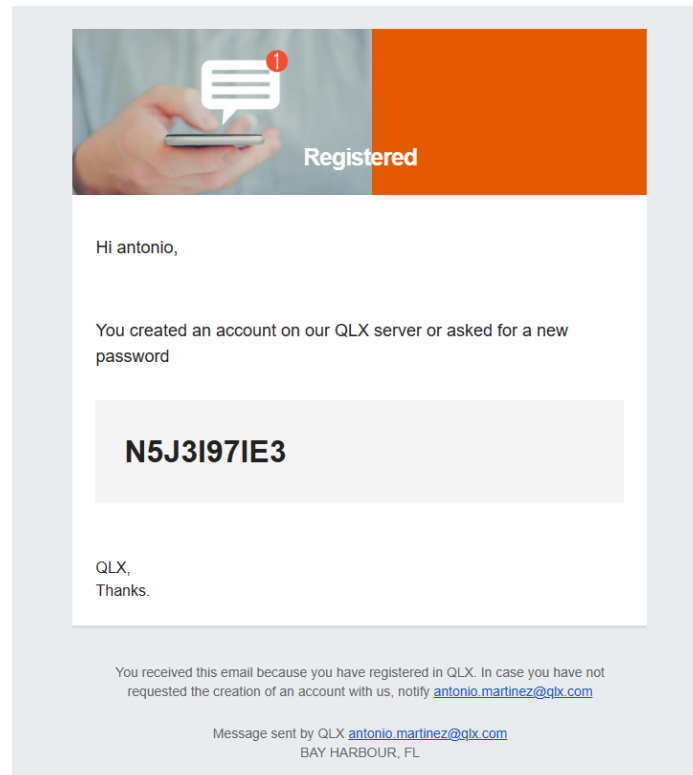
To be able to view this button, you must uncomment it from frontend. Still, there is a backdoor to stop the system directly in the backend if it is needed.

Login & Recover Password

The login works like in most websites. It works with JSON WEB TOKEN (**JWT**) protocol. Email, Password, and timestamp of session travels encrypted into the web.

To login, you have to use your username and password.

To recover your password, you have to use the email related to your username, and click "Forgot Password". System will send you mail with a new password.



Example

Parametrization of System's Tables

The main idea here is that you can easily add parameters to the system, and avoid hard coding directly into the frontend. Example: If there's a new country, you just add a new country to tables and it will appear automatically in every dropdown or list that needs it.

To see tables go to:

System > System Parameterization> System's Tables

Web Scrapping:

Web scraping system demo was made thinking of Mike's Cigars competitors, in order to have business intelligence over them and offer competitive prices, discounts, promotions, etc. To achieve that, the first step is to collect data from competitor's websites.

Products and their URL Registration

In this demo we created a manual system to Register a Product (the name of the product will be registered automatically in Upper case) and the url of the competitor's websites.

To register product go to:

Register > Register URL to scan



Register URL for a product

Product**Mike's Cigar****Famous Smoke****Cigars International****JR Cigars****Atlantic Cigar****Neptune Cigar****Thompson Cigar**





For this demo, we determined and effectively extracted the data from **Mike's Cigar**, **Famous Smoke** and **Cigars International** websites. The URL pattern has to follow the logic in the example to work with this software.

Once we know the pattern of the URL, we can scan all website urls and only keep those we need.

Check and edit Products names and their URL

Like urls could change in the time, there's a page where you can look at every registered product and it will be possible to edit them. You will be able too, look at raw urls that are registered on MySQL database and filter them.

Example, in the 2nd table, filter URL by **"1994"**, you should get coincidences with the url that contains that word. To Be more detailed, you will notice that the product name is: **"La Flor Dominicana 1994"**. So, you could register this product's URL on every company where it was found.


Notifications 



Go to Scan webs

Registered URL to scan on MongoDB				
Product	Mike's Cigars	Famous Smoke	Cigar International	JR Cigars
DAVIDOFF NICARAGUA	https://mikes cigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua	https://www.famous-smoke.com/brand/davidoff-nicaragua-cigars	https://www.cigarsinternational.com/p/davidoff-nicaragua-cigars/1474796/	
DAVIDOFF PRIMEROS	https://mikes cigars.com/catalogsearch/result/index/?product_list_mode=list&q=davidoff+primeros&product_list_limit=25	https://www.famous-smoke.com/brand/davidoff-primeros-cigars	https://www.cigarsinternational.com/p/davidoff-primeros-cigars/1484890/	

10 rows

Find URL

Registered RAW URL on MySQL Database	
Company	URL
famous smoke	https://www.famous-smoke.com/brand/1994-by-la-flor-dominicana-cigars
mikes cigar	https://mikes cigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=1994+la
famous smoke	https://www.famous-smoke.com/brand/20-acre-farm-cigars
mikes cigar	https://mikes cigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=20+acre
famous smoke	https://www.famous-smoke.com/brand/2012-by-oscar-cigars
mikes cigar	https://mikes cigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=2012+oscar
famous smoke	https://www.famous-smoke.com/brand/22-minutes-to-midnight-connecticut-radiante-cigars
mikes cigar	https://mikes cigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=22+minutes
famous smoke	https://www.famous-smoke.com/brand/22-minutes-to-midnight-habano-de-oro-cigars
mikes cigar	https://mikes cigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=22+minutes

10 rows

1-10 of 3666

To see products go to:

Queries > Registered URL

Scan websites

At this point it is very straightforward. You select the websites that you want to scan, select a product (dropdown will increase automatically if you register more products) or write it (you have a predictive text feature), and then press the button "Scan Website"



Select Website(s)

- ☒ Mikes cigars.com
- ☐ CigarsInternational.com
- ☐ Famous-Smoke.com
- ☐ JRCigars.com
- ☐ AtlanticCigar.com
- ☐ NeptuneCigar.com
- ☐ Thompsoncigar.com

Select an option



daviDOFF NICARAGUA

Add another Website for Scan

After the scanning process, you will have a table with all the scanned data. You will be able to filter it by size, and by product's name (names are generated dynamically).

Add another Website for Scan

Rotate View Export to Excel Update Price Save scan in DB Go to History of Scans

Filters
[Clear Filter](#)

Quantity
 Single Pack of 4 Pack of 5 Box of 12
 Box of 14 Pack of 20 Tin of 6 Pack of 30

Product [Info](#)
 Short Corona Toro Robusto Diadema

Set Price
 Amount Below

Report
 Average Lowest

54 coincidence(s)

WEBSITE	NAME	QUANTITY	PRICE	STOCK	URL	Actions
Mikes cigars.com	Davidoff Nicaragua Diadema Natural 6 1/2 x 50	BOX OF 12	285.60	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Diadema Pack Natural 6 1/2 x 50	PACK OF 4	95.50	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Primeros 6 Maduro 4 1/8 x 34	UNIT OF 30 TIN OF 6	170.95	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Robusto Box Pressed Natural 5 x 48	BOX OF 12	217.50	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Robusto Box Pressed Pack Natural 5 x 48	PACK OF 4	72.50	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Robusto Tubos Natural 5 x 50	BOX OF 12	223.50	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Robusto Tubos Pack Natural 5 x 50	PACK OF 4	74.50	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Short Corona Natural 3 3/4 x 46	BOX OF 14	191.95	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Short Corona Pack Natural 3 3/4 x 46	PACK OF 5	68.50	IN STOCK	LINK	Compare
Mikes cigars.com	Davidoff Nicaragua Toro Natural 5 1/2 x 54	BOX OF 12	265.50	IN STOCK	LINK	Compare

10 rows 1-10 of 54

You can Rotate view of the table that will help you to compare items side-by-side:

[Rotate View](#)
[Export to Excel](#)
[Update Price](#)
[Save scan in DB](#)
[Go to History of Scans](#)

Filters

[Clear Filter](#)

Quantity

[Single](#)
[Pack of 4](#)
[Pack of 5](#)
[Box of 12](#)
[Box of 14](#)
[Pack of 20](#)
[Tin of 6](#)
[Pack of 30](#)

Product [Info](#)

[Short Corona](#)
[Toro](#)
[Robusto](#)
[Diadema](#)

Set Price

[Amount](#)
[Below](#)

Report

[Average](#)
[Lowest](#)

54 coincidence(s) Q Search X

WEBSITE	Mikescigars.com	Mikescigars.com	Mikescigars.com	Mikescigars.com
NAME	Davidoff Nicaragua Diadema Natural 6 1/2 x 50	Davidoff Nicaragua Diadema Pack Natural 6 1/2 x 50	Davidoff Nicaragua Primeros 6 Maduro 4 1/8 x 34	Davidoff Nicaragua Robusto Box Press
QUANTITY	BOX OF 12	PACK OF 4	UNIT OF 30 TIN OF 6	BOX OF 12
PRICE	285.60	95.50	170.95	217.50
STOCK	IN STOCK	IN STOCK	IN STOCK	IN STOCK
URL	LINK	LINK	LINK	LINK
Actions	Compare	Compare	Compare	Compare

If you click on the button “Compare”, it will show you a modal with the products that their name and quantity coincide with the one that was clicked. This view will be showing brands side-by-side.

Notifications

Select Website(s)

☒ Mikescigars.com

☒ Cigarsinternational.com

Select an option

[DAVIDOFF NICARAGUA](#)

Details: [Close](#)

Comparison

Company	Mikescigars.com	Cigarsinternational.com	Famous-smoke.com
Product	DIADEMA NATURAL 6 1/2 X 50	DIADEMA (SALOMON) (6.5"X50)	DIADEMA 6 1/2 X 50- NATURAL FIGURADO
Qty	BOX OF 12	BOX OF 12	BOX OF 12
PRICE	285.60	285.60	285.60

WEBSITE	NAME	QUANTITY	PRICE	STOCK	URL	Actions
Mikescigars.com	DIADEMA NATURAL 6 1/2 X 50	BOX OF 12	285.60	IN STOCK	LINK	Compare
Mikescigars.com	Davidoff Nicaragua Diadema Pack Natural 6 1/2 x 50	PACK OF 4	95.50	IN STOCK	LINK	Compare
Mikescigars.com	Davidoff Nicaragua Primeros 6 Maduro 4 1/8 x 34	UNIT OF 30 TIN OF 6	170.95	IN STOCK	LINK	Compare

In the future, the idea is to show with colors and/or indicators when price has risen or diminished, and send notifications to the user when this happens.

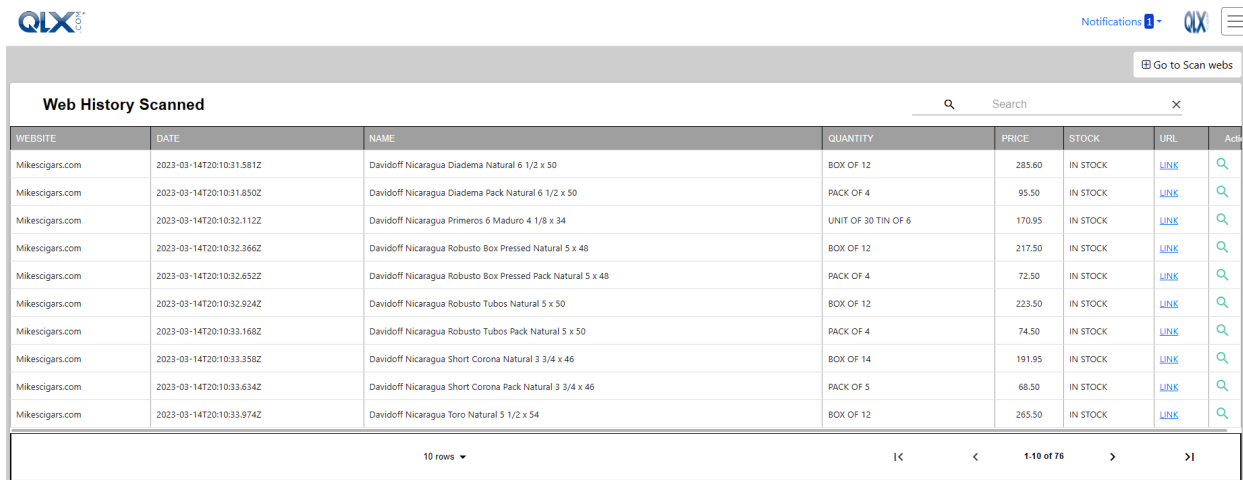
For this demo, you will be able to export to Excel spreadsheet, save the scanned websites to the Database and go to the full history of scanned websites.

To scan websites go to:

Queries > Scan Webs

History of scanned websites

There is a page where you can find all the history of scanned products around competitors' websites.



WEBSITE	DATE	NAME	QUANTITY	PRICE	STOCK	URL	Action
Mikesigars.com	2023-03-14T20:10:31.581Z	Davidoff Nicaragua Diadema Natural 6 1/2 x 50	BOX OF 12	285.60	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:31.850Z	Davidoff Nicaragua Diadema Pack Natural 6 1/2 x 50	PACK OF 4	95.50	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:32.112Z	Davidoff Nicaragua Primeros 6 Maduro 4 1/8 x 34	UNIT OF 30 TIN OF 6	170.95	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:32.366Z	Davidoff Nicaragua Robusto Box Pressed Natural 5 x 48	BOX OF 12	217.50	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:32.652Z	Davidoff Nicaragua Robusto Box Pressed Pack Natural 5 x 48	PACK OF 4	72.50	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:32.924Z	Davidoff Nicaragua Robusto Tubos Natural 5 x 50	BOX OF 12	223.50	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:33.168Z	Davidoff Nicaragua Robusto Tubos Pack Natural 5 x 50	PACK OF 4	74.50	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:33.358Z	Davidoff Nicaragua Short Corona Natural 3 3/4 x 46	BOX OF 14	191.95	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:33.634Z	Davidoff Nicaragua Short Corona Pack Natural 3 3/4 x 46	PACK OF 5	68.50	IN STOCK	LINK	Q
Mikesigars.com	2023-03-14T20:10:33.974Z	Davidoff Nicaragua Toro Natural 5 1/2 x 54	BOX OF 12	265.50	IN STOCK	LINK	Q

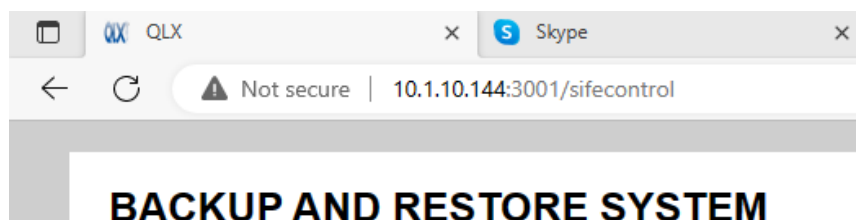
As we can know every detail of the products, it's possible to make more analysis that could be managed here or other page named reports.

Backup and Restore System:

We can make backups and restore them with one account with Administrator privileges, without touching the mongoDB database directly.

First, you must be logged in to the system with an administrator user.

In the browser add to the url: **“/sifecontrol”**



Then you will be able to touch some parts of the system, but still, if you are an outsider, even with an administrator account you won't be able to capture the data. You only will be able to do that if you are in the machine that is running the backend server.

 A screenshot of a web application titled "BACKUP AND RESTORE SYSTEM". The interface includes a form with the following sections:

- Endpoint:** A dropdown menu.
- Method:** A text input field with the value "ej: GET".
- Name of the file (upload or download):** A text input field with the value "ej: Data.json o Data.xls".
- Key 1:** A text input field with the value "key 1".
- Value 1:** A text input field with the value "Value 1".
- Key 2:** A text input field with the value "Key 2".
- Value 2:** A text input field with the value "Value 2".

 At the bottom of the form, there are five buttons: "Generate Backup in JSON", "RESTORE Backup-JSON to DB", "Transform EXCEL TO JSON", "TRANSFORM JSON TO EXCEL", and "Go Back to Menu".

To be able to work with this, you must know how endpoints are working in the backend, what data you need to send and what is being expected. Please check it out at Readme.md at the backend repository.

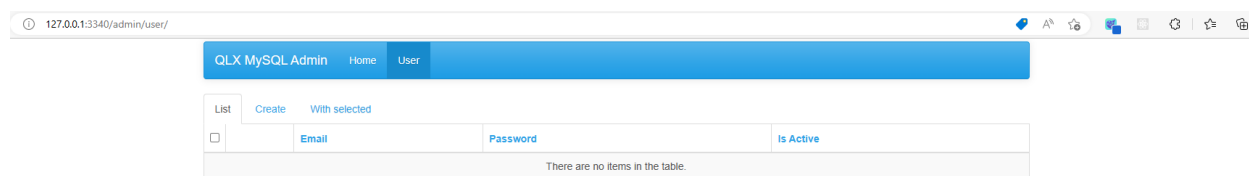
Backend API with Python FLASK for MySQL:

Live view of endpoints and database

Only for local machines to be able to watch, still, this is a different database from mongoDB, so you can have another layer of security to connect to MySQL database.

Go to: <http://localhost:3340/>

You will have an Admin endpoint that will take you to a live view of the database. It has a very basic USER model as an example. This model isn't used for login with the front-end.



The tables that you want have a view, must be added inside of the admin.py file.

QLX MySQL Admin Home User Scanned								
List (54)	Create	With selected						
<input type="checkbox"/>		Name	Web	Website	Stock	Price	Quantity	Date
<input type="checkbox"/>		Davidoff Nicaragua Diadema Natural 6 1/2 x 50	https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua	Mikescigars.com	IN STOCK	285.6	BOX OF 12	2023-03-16T13:03:27.97
<input type="checkbox"/>		Davidoff Nicaragua Diadema Pack Natural 6 1/2 x 50	https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua	Mikescigars.com	IN STOCK	95.5	PACK OF 4	2023-03-16T13:03:28.16
<input type="checkbox"/>		Davidoff Nicaragua Primeros 6 Maduro 4 1/8 x 34	https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua	Mikescigars.com	IN STOCK	170.95	UNIT OF 30 TIN OF 6	2023-03-16T13:03:28.34
<input type="checkbox"/>		Davidoff Nicaragua Robusto Box Pressed Natural 5 x 48	https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua	Mikescigars.com	IN STOCK	217.5	BOX OF 12	2023-03-16T13:03:28.60
<input type="checkbox"/>		Davidoff Nicaragua Robusto Box Pressed Pack Natural 5 x 48	https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua	Mikescigars.com	IN STOCK	72.5	PACK OF 4	2023-03-16T13:03:28.89
<input type="checkbox"/>		Davidoff	https://mikescigars.com/catalogsearch/result/index/?	Mikescigars.com	IN STOCK	223.5	BOX OF	2023-03-

Create endpoints

You can create endpoints to reach the database through this API, and obtain data in any format, in this case in JSON format. Note: you could create an endpoint to obtain data into .csv/.text file, etc...

/scan-url

This endpoint has GET and POST methods. With the GET method you will obtain all url for 3 websites of the demo, and with POST you will scan those websites (mike's cigars, famous smoke, cigar international)



QLX welcomes you to F

API HOST:

Start working on your project by follow:

Remember to specify a real endpoint path like:

- [/admin/](#)
- [/helloprotected](#)
- [/logout](#)
- [/lista-usuarios](#)
- [/allScans](#)
- [/](#)
- [/user](#)

For those endpoints that are related with the "GET" method you will have a list, and obtain whatever the API is programmed to return.

```

← ↻ ⓘ localhost:3340/allScans

{
  "list": [
    {
      "date": "2023-03-16T13:03:27.971Z",
      "id": 1,
      "name": " Davidoff Nicaragua Diadema Natural 6 1/2 x 50  ",
      "price": 285.6,
      "quantity": "BOX OF 12",
      "stock": "IN STOCK",
      "web": "https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua",
      "website": "Mikescigars.com"
    },
    {
      "date": "2023-03-16T13:03:28.160Z",
      "id": 2,
      "name": " Davidoff Nicaragua Diadema Pack Natural 6 1/2 x 50  ",
      "price": 95.5,
      "quantity": "PACK OF 4",
      "stock": "IN STOCK",
      "web": "https://mikescigars.com/catalogsearch/result/index/?product_list_limit=25&product_list_mode=list&q=davidoff+nicaragua",
      "website": "Mikescigars.com"
    },
    {
      "date": "2023-03-16T13:03:28.342Z",
      "id": 3,
      "name": " Davidoff Nicaragua Primeros 6 Maduro 4 1/8 x 34  "
    }
  ]
}

```

/allSortedUrl

This endpoint has GET and POST methods, and are a little tricky. With the GET method you will obtain all raw history data of every product, of every brand, per every website. With the POST method it will trigger the very slow process of web scraping for every registered URL with the previous endpoint.

The idea of this endpoint with the POST method is to be executed once per day. At the moment, it seems to be web scraping around **1800 url/hour**. In this demo with 3 websites as source data, that means around 120 brands per hour (consider that every brand is being searched on 3 websites, and every brand can have 5 products as average). **This process needs about 24 hours for 2808 brands.**

/allScans

This endpoint has GET and POST methods. With the GET method you will obtain a list of every live scan of certain products that were saved into the database. Remember that the frontend website has an option to scan in real-time a certain product in every selected website (at the moment there are 3 possible selections). For this to work properly, you should have been sorting and filtering every url of each of those 3 websites and linking them with just 1 brand. As this process is not viable in long term, the best approach is to obtain first the raw data and just request to the database (with **/allSortedUrl** endpoint)

Backend Notes:

The request limit handler of **node js** is set to a maximum of 10 requests per function. Normally, asynchronous functions would not have problems, but the problem is how node handles the resolution of rejected promises, since it waits to resolve all the rejected ones at the same time, and that causes the requests to accumulate until reaching the limit.

An initial solution was to increase the request limit per function to 70. Another measure that was taken is to ensure the closing of requests with a finally block, in the case of those made with **Puppeteer**. Also, for this last library, the instances of the default browser that it opens to scrape dynamic pages with javascript were closed. This slightly improves the efficiency and resource consumption of the device. For static pages, **Cheerio** is used to further increase efficiency.

A handler for rejected requests was also added to the node server, in order to prevent it from stopping when it does not know what to do with these rejected promises. At the moment, it seems to be handling approximately 120 brands per hour. Brands that are web

scraped on 3 websites, and where each of them can have a variable number of products (for an average total of **1800 urls scraped per hour**).

Since there is no other way to improve the speed at which web scraping is done, **the only thing left to do is to clean up those urls that have discontinued products**, and/or, return already non-existent products with a 404 error status; **and divide the process by website** (not run 3 websites at the same time).

Backend File Explorer:

Python - Flask

Root folder

-Pipfile:

This is the initial configuration of the virtual environment with **pipenv library** (you could use conda environment or pytorch, and try install with requirements.txt).

This file contains the necessary scripts to delete, create a database and run **Flask** server.

Besides, this is **one of the files** that you should change port from 3340 to another of your choices for Flask Backend.

-requirements.txt:

This file contains the necessary libraries (or almost, check Pipfile if there's one missing) that you would need if you want to use another virtual environment or make a global install.

-.env:

This file contains environment variables that are shared with **node js backend**.

Besides, this is **one of the files** that you should change port from 3340 to another of your choices, for Flask Backend.

./migration folder

This is the folder where the SQL database is created.

./docs/assets folder

This is the folder where the scripts to create/delete databases are.

./src folder

-main.py:

This is the main file that runs the server for the backend. Here you **must add** every **MODEL** and **ENDPOINT'S FUNCTION**, to be able to exist and work in the API.

Besides, this is **one of the files** that you should change port from 3340 to another of your choices, for Flask Backend.

-admin.py:

This is the admin View for Flask. Here you **must add** every model that you have created.

[./src/modelos folder](#)

This is the folder for our database's models.

- __init__.py:

This is an index file, it's necessary to make clean exports of the models of this folder to the outside folders. It's necessary too, to make clean imports of every model contained in every file of this folder.

-scanurl.py:

This file contains the model of the table in SQL, this model will store the data of every url that has to be scanned.

-scannedwebs.py:

This file contains the model of the table in SQL, this model will store the data resulting from the web scraping process, depending on the needs, it should be changed.

-sortedurls.py:

This file contains the model of the table in SQL, this model will store the data of one product and their different urls in every competitor's website. As this was un-viable in the long term, due to difficulty of matching product's names that are very variable on every website, this file as a database model should be discarded.

[./src/rutas folder](#)

This is the folder for our API's routes (endpoints).

- __init__.py:

This is an index file, it's necessary to make clean exports of the routes of this folder to the outside folders. It's necessary too, to make clean imports of every route contained in every file of this folder.

-scanurl.py:

This file contains the route to obtain (GET METHOD) a list of every possible URL to web scraping.

This file contains the route to store (POST METHOD) in the database, a list of every possible URL to web scraping. The process to **obtain every possible url is independent of the node's backend.**

-scanedwebs.py:

This file contains the route to obtain (GET METHOD) a list of every product obtained through the web scraping process.

This file contains the route to store (POST METHOD) in the database, a list of products obtained through the web scraping. The process to **obtain every product is Dependent of the node's backend.**

-sortedurls.py:

This file contains the route to obtain (GET METHOD) a list of every product with their respective urls in every website. But as mentioned before, the process to obtain it in this way was not viable in the long term.

This file contains the route to obtain (POST METHOD) in the database, the list of all possible urls, and **send them 1 by 1 to the node's backend** to start the web scraping process. This is the second most important process to **obtain every product and is Dependent of the node's backend.**

Javascript - Node

Root folder

-package.json:

This is the initial configuration of the environment with the **webpack library** (Node JS Framework). It contains the libraries that will be used and the scripts to run the node server.

Besides, this is **one of the files** that you should change port from 5001 to another of your choices for Node Backend.

-package-lock.json:

It contains the **exact** libraries that will be used. Just run: **npm ci**

-.env:

This file contains environment variables that are shared with the flask **backend**.

Besides, this is **one of the files** that you should change port from 5001 to another of your choices, for Node Backend.

-server.js:

This is the main file that runs the server for the backend. Here you **must add** every **ROUTE** and **MIDDLEWARE ROUTE**, to be able to exist and work in the API.

./middleware folder

This folder contains all the middleware files necessary to intercept requests to validate JWT, and register every move or requests that are made in the API.

./models folder

This folder contains all the models that are necessary to store data into the **MongoDB** database with help of the ORM library **mongoose**.

-urlToScan.js:

-webscrapping.js:

./routes folder

This folder contains all the routes that are used by this API. For this demo, the main purpose is all the functions inside of:

-urlToScan.js:

-webscrapping.js:

Probably the most important file in this demo. This file contains the functions and endpoints that are needed to webscrape, specifically:

```
router.post(`/find/generic`, async(req, res) => {...})
```

Note that with the middleware inside of server.js, the whole endpoint route will be:

```
"/webscrapping/find/generic"
```

The function above has 3 predetermined websites to web scrape (Cigars International, Mike's Cigars, Famous Smoke), always that initial url is given as is required (no problem, this will come automatically from flask backend). Beside, this endpoint function with post method, requires **3 important parameters** into the body with **JSON** format, that will be provided by Flask-Backend automatically; those parameters are:

```
{ item: "DAVIDOFF", web: "mikescigars", url: "www.example.com/brands/" }
```

Frontend File Explorer:

Javascript - Webpack

Root folder

-.env:

This file contains environment variables to connect to Node Backend and Flask Backend.

Besides, this is **one of the files** that you should change port from 3001 to another of your choices, for front-end.

-package.json:

This is the initial configuration of the environment with the **webpack library** (Node JS Framework). It contains the libraries that will be used and the scripts to run the node server.

Besides, this is **one of the files** that you should change port from 3001 to another of your choices for Frontend.

-package-lock.json:

It contains the **exact** libraries that will be used. Just run: **npm ci**

-webpack.dev.js:

It contains the developer server.

Besides, this is **one of the files** that you should change port from 3001 to another of your choices for Frontend, if it is not changed from the .env file.

./src folder

This is the folder that contains all necessary elements for our React app demo.

./src/js folder

-index.js:

This is the main file of our React app

-layout.js:

In this file you choose which component will behave as a page/views and control the frontend path to navigate to it.

[./src/js/store folder](#)

In this folder are all the global functions and global states of React.

[./src/js/component folder](#)

In this folder are all the components that can be reused on views of React. And special components to export to excel, pdf, etc...

[./src/js/views folder](#)

-registerUrl.js:

-registerUrlTable.js:

-registerUrl.js:

-mikescigar.js:

This file contains the live web scraping version, where you'll be able to compare all products of a specific brand.

-scannedWeb.js:

-webscrapedTable.js:

This is the most important view, it will go to the Flask Backend and ask for all the information stored over time, from the brands that have been scanned.

[./src/styles folder](#)

This folder contains all necessary stylesheets. Between them, most important for this demo are:

-index.css:

-predictive.css:

-rotateTable.css: