Documentation

Tools and node packages

Tools and node packages used:

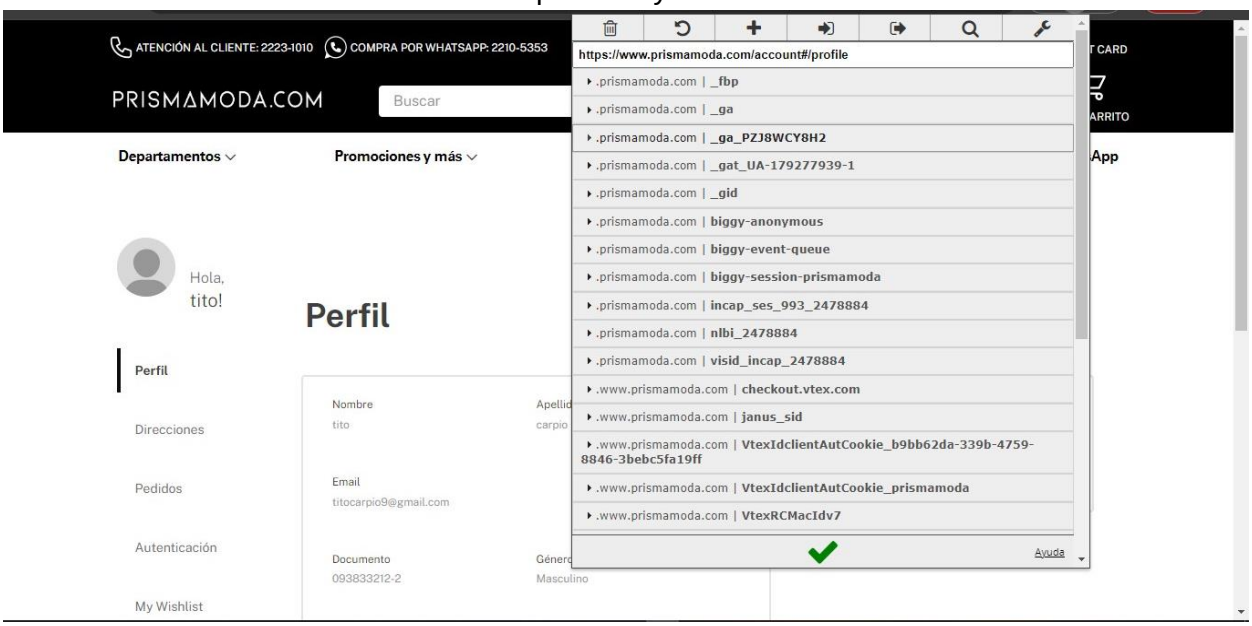
- **EditThisCookie**: browser extension to download cookies from a website visit.
- **Axios**: JavaScript library used to make HTTP requests from the browser or from a server using Node.js. visit.
- **fs**: file system operation module visit.

Scraping

The merchant to which scraping was performed is Primas Moda, from which the user's account information is obtained.

Steps for data recovery

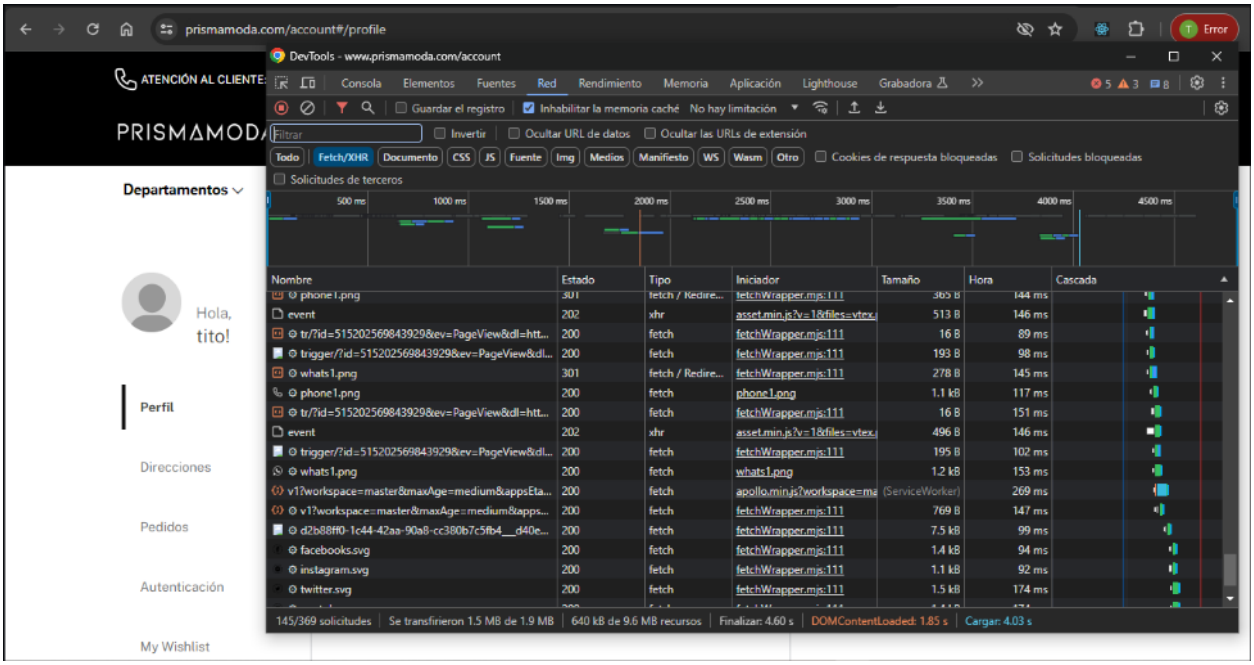
- Install the **EditThisCookie** browser extion that allows you to download cookies from the website.
- Log in to an account associated with the merchant or create an account, using the following link click here.
- Once you access the Prisma Moda website, cookies are downloaded from the site through the use of the extension that was previously installed.



- Clicking on the button below will copy the cookies to the computer clipboard in JSON

format.

- A text file called cookies.txt is created and the cookies extracted from the site are pasted into it.
- This file is added to a folder where a node js project is initialized.
- The requests made by the Prisma Moda website are identified through the use of the page inspection tools that browsers have. In the network section we can see the different requests which were validated using the HTTP Postman client.



Code JavaScript

Script Content

- Function that checks the existence of a file named cookies.txt in the root of the project and that it is not empty. It also formats the content of the file so that it can be used in the requests that are made.

```
function getCookies() {
  const cookiesFilePath = "cookies.txt";

  // Check if the cookie file exists
  if (!fs.existsSync(cookiesFilePath)) {
    console.error(`The file '${cookiesFilePath}' does not exist.`);
    return null;
  }

  // Check if the cookie file is empty
  const stats = fs.statSync(cookiesFilePath);
  if (stats.size === 0) {
    console.error(`The file '${cookiesFilePath}' is empty.`);
    return null;
  }

  // Load and parse the cookie JSON file
  const cookiesJSON = fs.readFileSync(cookiesFilePath, "utf8");
  const cookies = JSON.parse(cookiesJSON);

  // Converts JSON cookies to a valid cookie string for the request
  const cookiesString = cookies
    .map((cookie) => `${cookie.name}=${cookie.value}`)
    .join("; ");

  return cookiesString;
}
```

- Then we continue with the two functions that work with the responses of the requests to format them in the report displayed in the console.

User Data

```
//function to print user data on console
function printUserData(userData) {
  console.log(
    "------------------------------------------------------------------"
  );
  console.log(
    "                            USER DATA:                            "
  );
  if (userData.data.profile == null) {
    console.error("Cookies expired");
    return;
  }

  // Extract user data
  const {
    firstName,
    lastName,
    birthDate,
    email,
    gender,
    homePhone,
    businessPhone,
    document,
  } = userData.data.profile;

  console.log("User Data:", {
    Name: firstName,
    Lastname: lastName,
    Birthdate: birthDate,
    Email: email,
    Gener: gender,
    HomePhone: homePhone,
    BusinessPhone: businessPhone,
    Document: document,
  });
}
```

User addresses

```
//function that prints the user's addresses
function printUserAddress(userAddress) {
  console.log(
    "---------------------------------------------------------------"
  );
  console.log(
    "                          USER ADDRESS:                        "
  );
  // validation of the response of the request comes empty due to expired cookies
  if (userAddress.data.profile == null) {
    console.error("Cookies expired");
    return;
  }

    // the validation of the request response is empty because the user has no greyed out addresses
  if (userAddress.data.profile.addresses.length == 0 ) {
    console.error("User has no address record");
    return;
  }

  // Extract address data
  userAddress.data.profile.addresses.forEach((address, index) => {
    index++;
    console.log("Address: " + index, {
      Country: address.country,
      Postal_Code: address.postalCode,
      State: address.state,
      City: address.city,
      Street: address.street,
    });
  });
}
```

- An async function is created which through the use of Axios makes a request to the API of the web page. To the request we add the body that was identified using Postman and in the header we add the cookies that were previously saved and formatted.

Example of one of the requests

```
// Making the POST request with axios that will allow me to get the user's data
const response = await axios.post(
  "https://www.prismamoda.com/_v/private/graphql/v1?workspace=master&maxAge=long&appsEtag=remove&domain=store&locale=es-SV",
  {
    // fields in the body of the request
    operationName: "Profile",
    variables: {},
    extensions: {
      persistedQuery: {
        version: 1,
        sha256Hash:
          "c85be2148d672083db2e34fef20a4b38887fc827cf0546741d0926cef8c2ef58",
        sender: "vtex.my-account^@1.x",
        provider: "vtex.store-graphql^@2.x",
      },
    },
  },
  {
    headers: {
      Cookie: cookiesString, // Set cookies in the header
    },
  }
);
```

Code TypeScript

Script Content

- Creation of interfaces to define custom types to be used to validate the shape of the objects of the request responses and cookies.
- Function that checks the existence of a file named cookies.txt in the root of the project and that it is not empty. It also formats the content of the file so that it can be used in the requests that are made.

```
function getCookies(): string | null {
  const cookiesFilePath = "cookies.txt";

  // Check if the cookie file exists
  if (!fs.existsSync(cookiesFilePath)) {
    console.error(`The file '${cookiesFilePath}' does not exist.`);
    return null;
  }
  // Check if the cookie file is empty
  const stats = fs.statSync(cookiesFilePath);
  if (stats.size === 0) {
    console.error(`The file '${cookiesFilePath}' is empty.`);
    return null;
  }

  // Load and parse the cookie JSON file
  const cookiesJSON = fs.readFileSync(cookiesFilePath, "utf8");
  const cookies: Cookie[] = JSON.parse(cookiesJSON);

  // Converts JSON cookies to a valid cookie string for the request
  const cookiesString = cookies
    .map((cookie) => `${cookie.name}=${cookie.value}`)
    .join("; ");

  return cookiesString;
}
```

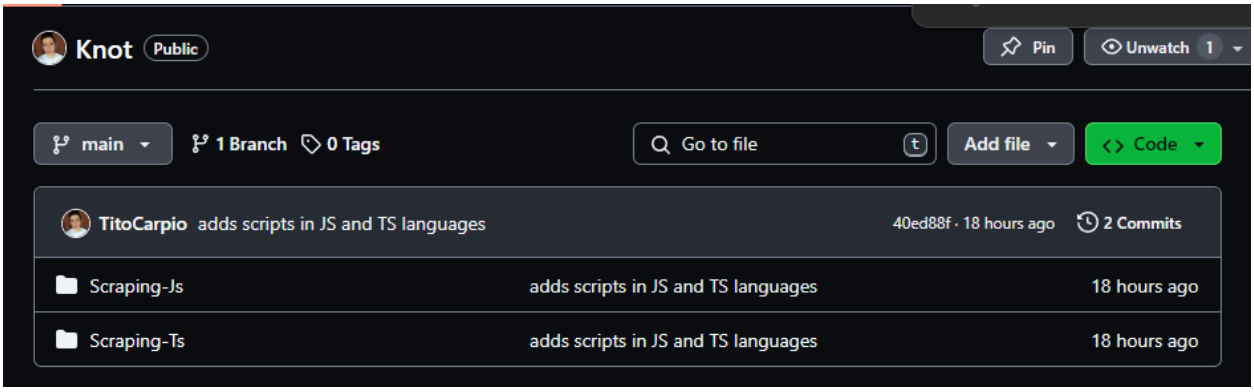
- Then we continue with the two functions that work with the request responses to format them in the report displayed in the console. Identical to those shown above, with the only change being the Typescript syntax.
- An asynchronous function is created using Axios to make a request to the API of the web page. To the request is added the body that has been identified by Postman and in the header are added the cookies that have been previously saved and formatted.

Example of one of the requests

```
// Making the POST request with axios that will allow me to obtain the user's addresses
const response = await axios.post<UserData>(
  "https://www.prismamoda.com/_v/private/graphql/v1?workspace=master&maxAge=long&appsEtag=remove&domain=store&locale=es-SV",
  {
    // fields in the body of the request
    operationName: "Addresses",
    variables: {},
    extensions: {
      persistedQuery: {
        version: 1,
        sha256Hash:
          "0aa6dc896b55a617e32a6c42a58dccd2e016c6278243b236d595991732f85b40",
        sender: "vtex.my-account^@1.x",
        provider: "vtex.store-graphql^@2.x",
      },
    },
  },
  {
    headers: {
      Cookie: cookiesString, // Set cookies in the header
    },
  }
);
```

Script Execution

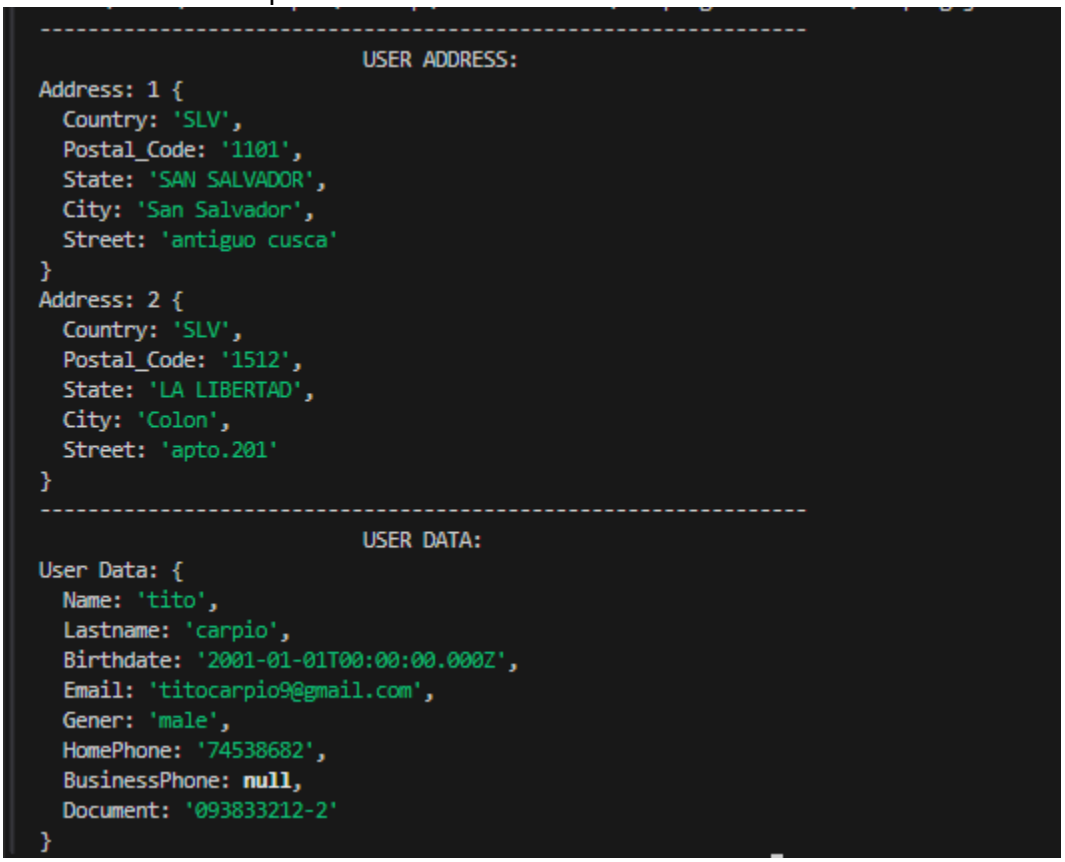
- The scripts can be found in the following repository: Knot.
- In the repository you will find two folders in which one has the script written in JavaScript language and another folder that will contain the script in TypeScript.



- Clone the repository
- Verify that you have Node.js installed on your computer.
- Verify that you have TypeScript installed on your computer (TypeScript)
- Install the **axios** and **fs** packages for each of the scripts.
- Copy cookies from the Prima Moda website with the browser extension once logged in. click here
- Paste the cookies in the cookies.txt file found in each of the scripts.

Script execution in JavaScript language

- Go to the Scrapin-Js folder and open a terminal of your choice in that location
- Execute the command *node .\scraping.js*
- The result of the script should look like this



Script execution in TypeScript language

- Go to the Scrapin-Ts folder and open a terminal of your choice in that location
- Execute the command *ts-node .\scraping.ts*
- The result of the script should show a result like the one above.