/

Universidad Europea Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

GRADO EN INGENIERÍA INFORMÁTICA

PROYECTO DE COMPUTACION III

JORGE DUMONT LORENZO , JOSE MANUEL RUBIO TALEGÓN E ISABEL SUTIL MARTÍN

ALFONSO VILCHEZ

CURSO 2020-2021

Índice

Antecedentes

Contexto y justificación

Actualmente existen numerosas páginas web que ofrecen información para organizar tus propios viajes. Se está viendo el decaimiento de agencias de viajes. La necesidad de este proyecto surge cuando, analizando el nicho de mercado, se puede ver una gran incertidumbre por parte de las personas al tener que buscar la información en numerosas páginas web. Es muy difícil sacar conclusiones ante una gran cantidad de información dispersa.

Planteamiento del problema

Este proyecto está destinado a la recuperación de datos acerca de diferentes localidades y sus características más destacadas mediante una página web. Se recuperan los datos mediante Web Scraping en “tiempo real” de las diferentes búsquedas por parte de los usuarios y se muestran los resultados mediante la estrategia del cruce de datos.

Por otro lado, todos los datos recuperados por los scrappers son almacenados en una BBDD compartida alojada en un servidor para su modificación y mantenimiento.

Existen dos modalidades de utilización de la solución software, la primera es acceso a información menos extensa por parte de los usuarios que no se registran y la segunda opción es el acceso a una información más detallada para los usuarios que se han registrado

La composición planteada del proyecto en su conjunto global es:

- Menu de navegacion
  La parte del front-end contará con un menú de navegación entre las diferentes interfaces de la web para asegurar la mejora del User Experience y anunciar en todo momento en que parte se encuentra el usuario, este menú te permitirá navegar entre el “Inicio” que muestra información poco detallada para los usuarios no registrados y más información para los usuarios registrados, la pestaña de “Información del usuario” donde muestra información básica de cada usuario registrado y la pestaña de acceso restringido para el administrador “Administración de Cuentas”. También se tendrá en cuenta de una manera visual si el usuario que está usando la herramienta está registrado o no.

- Buscador
  Funcionalidad que permite al usuario introducir el nombre de una localidad para obtener información sobre su búsqueda, esta funcionalidad también cuenta con un mensaje de error en el caso de que se haya introducido mal la búsqueda. Será la encargada de realizar peticiones a la BBDD para recuperar los datos relacionados con dicha búsqueda, la idea es buscar cruzando los datos (esta ultima subfuncionalidad no será prioritaria debido al desconocimiento sobre la realización de la misma).

Otra funcionalidad será implementar un histórico en el buscador, esto genera la posibilidad de que el usuario pueda registrarse y guardar su historial de búsqueda, permitiéndole comparar el análisis anterior en diferentes momentos de tiempo (será una funcionalidad extra para solamente los usuarios registrados).

- Administracion de usuarios
  A raíz de la funcionalidad “Conexión con la BBDD” también es necesario una gestión de usuarios para el correcto funcionamiento de los usuarios y su acceso al software, esta funcionalidad será la encargada de dar de alta, de baja, modificar los datos, entre otras funciones. Por otro lado, también se podrán realizar otras configuraciones sobre la web desde esta funcionalidad.

- Análisis de Sentimiento
  Otra funcionalidad destinada a evaluar la opinión de los usuarios acerca de un tema o producto, esto se realizará mediante la conexión a una API privada que devolverá los resultados acerca del texto de los usuarios.

En esta memoria se va a explicar el desarrollo backend de la aplicación.

Objetivos

Objetivos general

Realización de una aplicación software capaz de ofrecer información relacionada con viajes a tiempo real.

Objetivos específicos

O1: Implementación de un sistema de control de usuarios
O2: Extracción de información a tiempo real
O3: Almacenamiento correcto y seguro de la información
O4: Extracción de la información necesaria para ser utilizada en el frontend

Beneficios del proyecto

Este proyecto ofrece la capacidad de enseñar en un solo sistema información actualizada. Por otro lado, ofrece un análisis de sentimiento de las distintas cosas que se muestran para indicar lo que otros usuarios piensan. Las consecuencias de esto son:

1. Esto evita incertidumbre al tener toda la información junta.
2. Facilita el proceso de toma de decisiones

Desarrollo del proyecto

Planificación del proyecto

Nuestro proyecto se ha organizado en base a un diagrama de Gantt que adjuntamos.

Las tareas se han definido a lo largo del tiempo de duración de la asignatura PC3. Las tareas se han marcado en el tiempo en el que se han planeado realizarse. En verde están las tareas realizadas, en rojo las que no se han podido realizar a tiempo y en gris las que aún no se han realizado.
En él habíamos definido las distintas tareas que íbamos a tener que realizar para terminar nuestro proyecto en función de sus objetivos. Las siguientes tareas son:

- **Diseño del proyecto**
- **Planificación del proyecto**
- **Realizar los scrappings:** Para ello tenemos 2 subtareas:
    - Mediante las librerías de Python, hacer srapping para bajarnos la información del tiempo, pueblos, alojamiento…
    - Crear variables de tipos de tiempo y otros datos para poder enlazar unas fuentes de datos con otros, así el usuario no solo busca sobre un pueblo sino que puede buscar en base al tiempo
- **Creación de un proyecto laravel**
- **Diseño de la BBDD**
- **Elaboración de la BBDD local:** Creación de la BBDD en mySQL
- **Login de usuario**
- **Conexión de la BBDD con el proyecto**
- **Elaboración de la BBDD en la nube**
- **Uso de scrappers en el proyecto**
- **Conexiones BBDD - proyecto**: toda la información se gestiona correctamente
- **Pruebas**
- **Elaboración de una memoria explicativa**

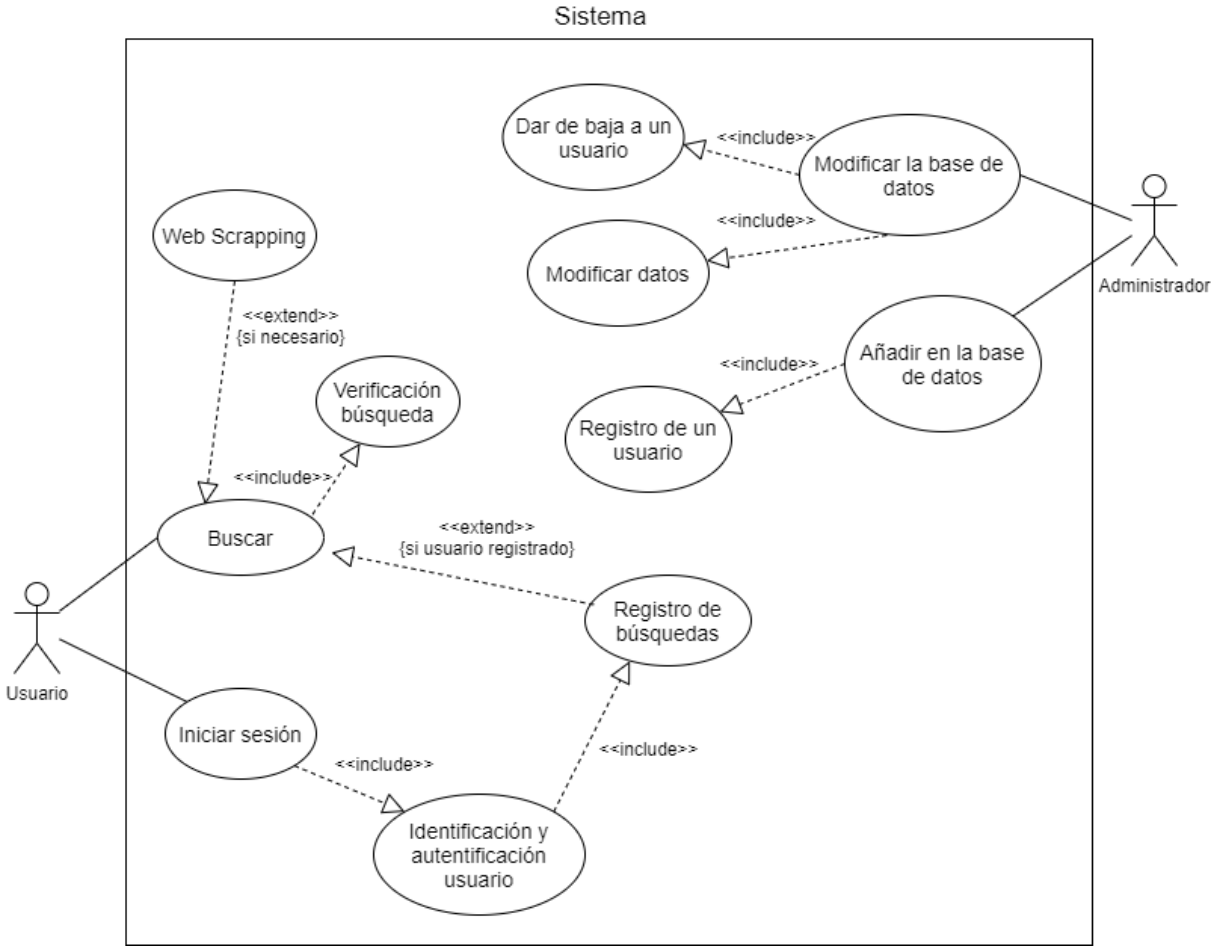
	15/02/2021 - 28/02/2021	1/03/2021 - 14/03/2021	15/03/2021 - 23/03/2021	24/04/2021 - 14/04/2021
Diseño del proyecto				
Planificación del proyecto				
Scrapping del tiempo				
Scrapping de TripAdvisor				
Scrapping de Airbnb				
Creaccion de un proyecto Laravel				
Diseño de la BBDD				
Elaboración de la BBDD local				
Login de usuario				
Conexión BBDD - proyecto				
Elaboración de la BBDD en la nube				
Uso de scrappers en el proyecto				
Conexiones BBDD - proyecto				
Pruebas				
Elaboración de una memoria explicativa				

	15/04/2021 - 27/04/2021	1/05/2021 - 11/05/2021	12/05/2021 - 31/05/2021	1/06/2021 - 15/06/2021
Diseño del proyecto				
Planificación del proyecto				
Scrapping del tiempo				
Scrapping de TripAdvisor				
Scrapping de Airbnb				
Creaccion de un proyecto Laravel				
Diseño de la BBDD				
Elaboración de la BBDD local				
Login de usuario				
Conexión BBDD - proyecto				
Elaboración de la BBDD en la nube				
Uso de scrappers en el proyecto				
Conexiones BBDD - proyecto				
Pruebas				
Elaboración de una memoria explicativa				

Descripción de la solución

Casos de Uso

A continuación se puede ver un diagrama de casos de uso que muestra las interacciones de los usuarios con el sistema.



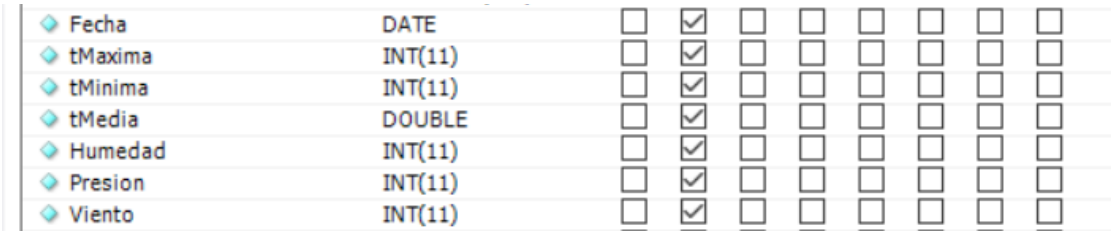
Fuentes de datos

En este proyecto estaba ideado para la utilización de tres fuentes de datos: tiempo, Tripadvisor y Airbnb. A continuación se explicaran cada una de las fuentes.

1. **eltiempo.es**
   De esta fuente de datos lo que hacemos es ir sacando un histórico de las temperaturas que hacen en cada pueblo de España. Se sacan las tres temperaturas que se muestran en las franjas horarias de las 8:00, 14:00 y 20:00 y se hace una media. La media se va actualizando cada vez que llamamos los scrappers y recogemos la información meteorológica de los 7 días de la semana.

Esta información se extrae mediante srappings utilizando la librería de Beautiful Soup. Dentro de el tiempo sacaremos los siguientes datos:

Nombre del dato	Tipo de dato
Fecha	Date
Temperatura máxima	Integer
Temperatura mínima	Integer
Temperatura media	Double
Humedad	Integer
Viento	Integer
Viento	Integer



2. Tripadvisor

En la fuente de Tripadvisor se extrae información de cada pueblo en referencia a:

1. Las cosas que podemos hacer y los puntos de interés que podemos encontrar. Se recoge el nombre de la actividad y su referencia.
2. Los mejores restaurantes de cada pueblo. De cada restaurante se extrae información sobre su nombre, tipo de comida que ofrecen, el rango de precio entre los que podemos encontrar sus platos y comentarios destacados que han puesto los usuarios sobre ellos.
3. Información sobre los hoteles de la zona, sus características y lo que otros usuarios han opinado sobre ellos.

Toda esta información se extrae mediante un webdriver importado desde la librería de Selenium. No se va a utilizar la librería BeautifulSoup ya que, de la forma que está elaborada la página web, no permite extraer la información. Por otro lado, existe una API disponible pero de muy difícil acceso al público.

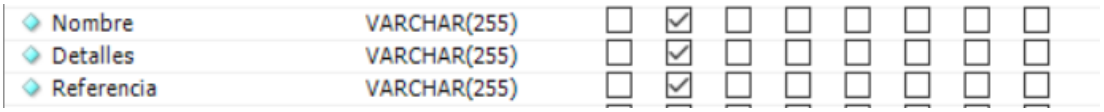
Las tablas que vamos a obtener con la información serán:

OCIOS:

Nombre del dato	Tipo de dato
Lugar	String
Referencia	String

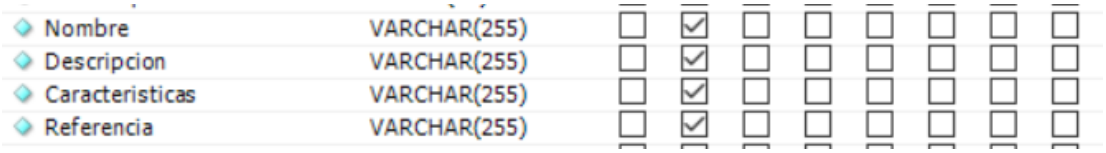
RESTAURANTES:

Nombre del dato	Tipo de dato
Nombre	String
Detalles	String
Referencia	String



HOTELES:

Nombre del dato	Tipo de dato
Nombre	String
Detalles	String
Características	String
Referencia	String



3. **Airbnb**

Esta fuente nos hubiera ayudado a recolectar datos acerca de viviendas en alquiler en la zona buscada, así se podría barajar varias posibilidades de alquiler en diferentes zonas de la localidad. Los datos que recopilaríamos van desde el título de la vivienda, localización dentro del municipio, valoración por parte de los usuarios y principales características de la vivienda.

Se realizó un scrapper mediante la librería de BeautifulSoup en python debido a que la api no está a fácil disposición del público. Posteriormente la página web ha cambiado y no ha sido posible solucionar el código por el tiempo de desarrollo que implicaba.

La tabla planteada con la información que recogeríamos es:

Nombre del dato	Tipo de dato
Título	String
Localización	String
Valoración media	Double
Reglas de uso	String
Nº habitaciones	Integer
Características	String

Análisis general de la BD

La base de datos que vamos a hacer uso es MySQL.

La estructura de la BBDD está formada por 8 tablas que se describen a continuación:

1. **climas:**
   Según nuestra consideración el clima es un elemento/información muy importante en relación con el tema del que trata el proyecto. Al relacionarse con diferentes municipios de España detectamos una información de valor acerca del clima en cada municipio sobre el que esté interesado el usuario.

   La fuente utilizada ha sido https://www.tutiempo.net/ debido a que nos parecía una extracción relativamente fácil y de manera conjunta era una página web apta para desarrollar el Scraper con BeautifulSoup que era uno de los requisitos de la entrega, por otro lado, es una página web con la que habíamos trabajado previamente por lo que resulto facil.

   Los campos por los que está compuesto son:
   - id: identificador del dato
   - idMunicipio: identificador del municipio del que se ha extraído el tiempo
   - Fecha: Día de la extracción
   - tMaxima: temperatura máxima en el día
   - tMinima: temperatura mínima en el día
   - tMedia: temperatura media en el día
   - Humedad: porcentaje de humedad.
   - Presión: presión medida en pascales
   - Viento: calculado en kilómetros por hora

2. **3 tablas de TripAdvisor:**
   Otro punto fuerte de nuestro proyecto es la recuperacion de informacion relacionada con la herramienta TripAdvisor, pagina web muy utilizada a nivel mundial que puede proporcionar información sobre cada municipio de manera categorizada y destacada por los usuarios mediante el uso de valoraciones y reseñas.
   En relación con esta fuente (https://www.tripadvisor.es/) de datos cabe destacar que hemos desarrollados 2 Scrapers diferentes de esta misma fuente debido a que se pueden contabilizar como dos Scraper diferentes al tener una estructura de desarrollo diferentes:
   1. Puntos de Interés, Restaurantes y Hoteles
   2. Comentarios acerca de lo analizado en el Scraper previo (Puntos de Interés, Restaurantes y Hoteles)

   a. **restaurantes**
      Los campos por los que está compuesto son:
      - id: identificador del dato
      - idMunicipio: identificador del municipio del que se ha extraído el dato
      - Nombre: nombre del restaurante
      - Detalles: Detalles del restaurante
      - Referencia: url al restaurante

   b. **ocios**
      Los campos por los que está compuesto son:
      - id: identificador del dato

- idMunicipio: identificador del municipio del que se ha extraído el dato
- Nombre: nombre del ocio
- Referencia: url al ocio

c. **hotels**

Los campos por los que está compuesto son:

- id: identificador del dato
- idMunicipio: identificador del municipio del que se ha extraído el dato
- Nombre: nombre del hotel
- Detalles: Detalles del hotel
- Caracteristicas: Características del hotel
- Referencia: url al hotel

3. **busquedas:** Se guarda información cuando se realiza una búsqueda y el análisis de sentimiento obtenido

   Los campos por los que está compuesto son:

   - id: identificador del dato
   - idMunicipio: identificador del municipio del que se ha extraído el dato
   - AnalisisSentimiento: Número que indica el análisis de sentimiento
   - Scraper: Número del scraper realizado

4. **2 tablas para el manejo de los usuarios:**

   La tabla de los usuarios es la tabla donde se guarda la información de los usuarios y los administradores. Se usa la misma tabla con los mismos campos pero lógicamente cada usuario (registrado o sin registrar) y los administradores tendrán información distinta. Por lo tanto, la información que no pertenezca a ese usuario la pondremos a null. Los campos de esta tabla de usuarios son los siguientes:

   - id: identificador del dato
   - idMunicipio: identificador del municipio del que se ha extraído el dato
   - AnalisisSentimiento: Número que indica el análisis de sentimiento
   - Scraper: Número del scraper realizado

   En la segunda tabla (sessións) se guarda automáticamente la información del inicio de sesión de un usuario. Está relacionada con la tabla users mediante el campo user_id. Esta información no se usa posteriormente.

El scraper de AirBnB se realizó con el siguiente propósito: en algunos pueblos no hay hoteles. La estructura del scrapper creado (no funcional) es la siguiente:
La información la recolectamos de https://www.airbnb.es y lo hemos tenido que realizar con la librería de Selenium ya que esta página no funciona si se le desactiva el Javascript. Los datos que recogemos son los siguientes:

4. **Nombre del municipio**

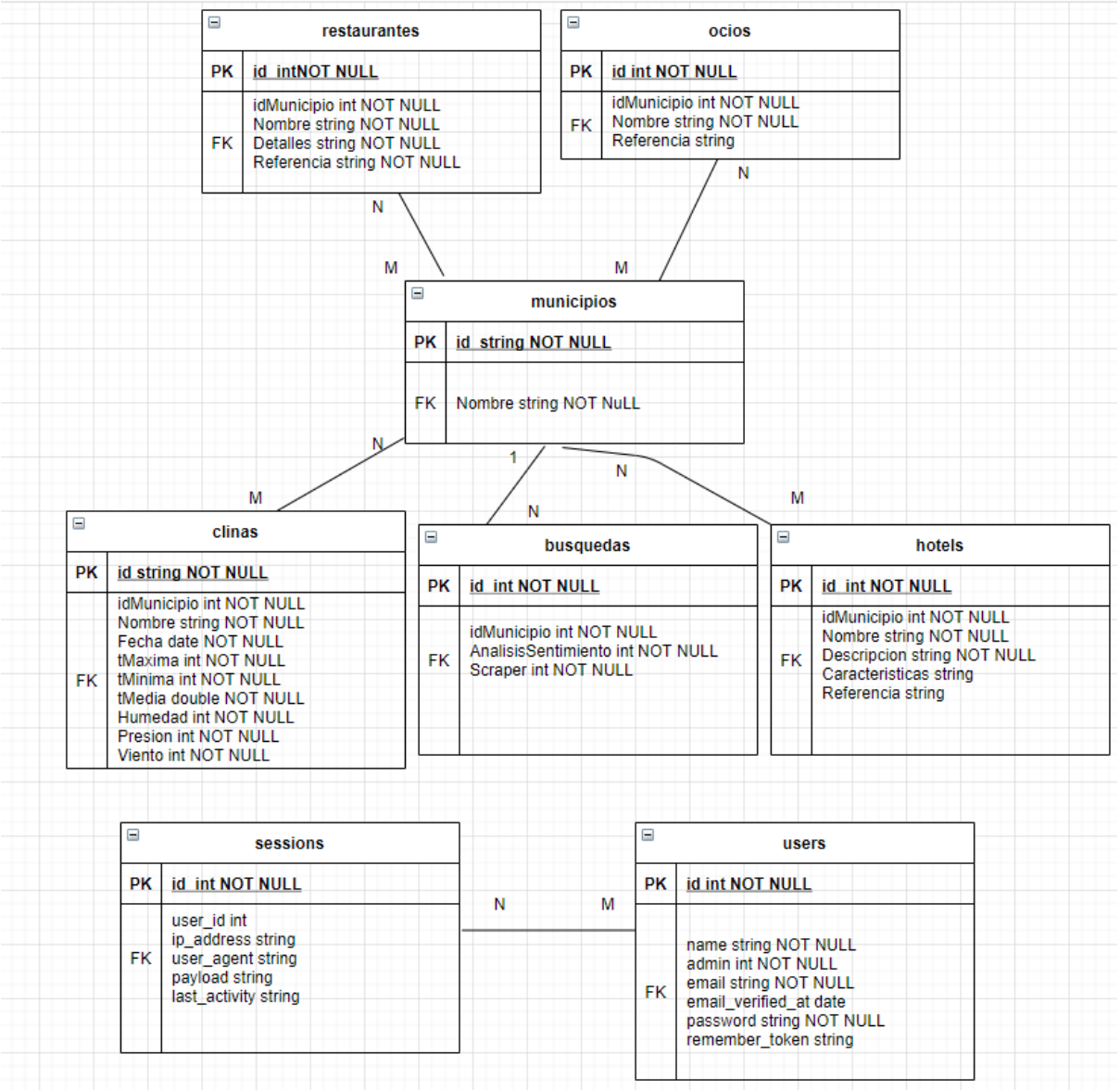
   El nombre lo necesitaremos tener como identificador para que en un futuro cuando queramos recoger información de los diferentes scrapers tengamos un valor que los una.

5. **Título**

   Nombre del domicilio que está puesto en la página web, este nombre lo ha puesto el dueño de la casa.

6. **Descripción**
   Una descripción sobre el domicilio y las ventajas de este. En este apartado hemos tenido un problema ya que hay veces que se recogen las descripciones de las casas y hay veces que no. Tendremos que seguir indagando cómo podemos sustituir esta información ya que pensamos que una descripción es una pieza de información que nos puede ayudar bastante, o en caso contrario, seleccionar otra página de casas.
7. **Detalles**
   Los detalles son información de lo que la casa incluye, los servicios. Como pueden ser; la cocina y sus electrodomésticos, wifi, calefacción, piscina… Son detalles que marcan la diferencia y que el usuario necesitará seguramente para decidirse entre una casa y otra.
8. **Características**
   Los detalles son información de la estructura de la casa; cuántos baños tiene, cuántas habitaciones tiene, cuántas camas tiene.. Es información que la utilizaremos para ver qué casa le recomendamos al cliente a partir de la búsqueda que ellos hagan.

Esquema de la BBDD



Dejamos el enlace al diagrama para poder verlo con más detalle.

Conectividad con la BBDD

En el proyecto de Laravel tenemos un archivo que es el .env en el que metemos los datos de la conexión a la bbdd:

```
DB_CONNECTION=mysql
DB_HOST=2.139.176.212
DB_PORT=3306
DB_DATABASE=prgrupoa
DB_USERNAME=pr_grupoa
DB_PASSWORD=pr_grupoa
```

La base de datos está localizada en un servidor remoto. Para acceder a ella se ha de indicar el host y puerto para todas las bases de datos de los distintos grupos que componen la asignatura. Por otro lado hay que incluir los datos específicos de nuestra base de datos, usuario contraseña.

Login + Registro de usuarios

El Login y el Registro de la aplicación se realizan usando tokens. A cada usuario se le asigna un token cuando inicia la sesión.

Cuando nos registramos se crea ese token y al hacer un login es el resultado que te devuelve después de introducir el email y la contraseña.

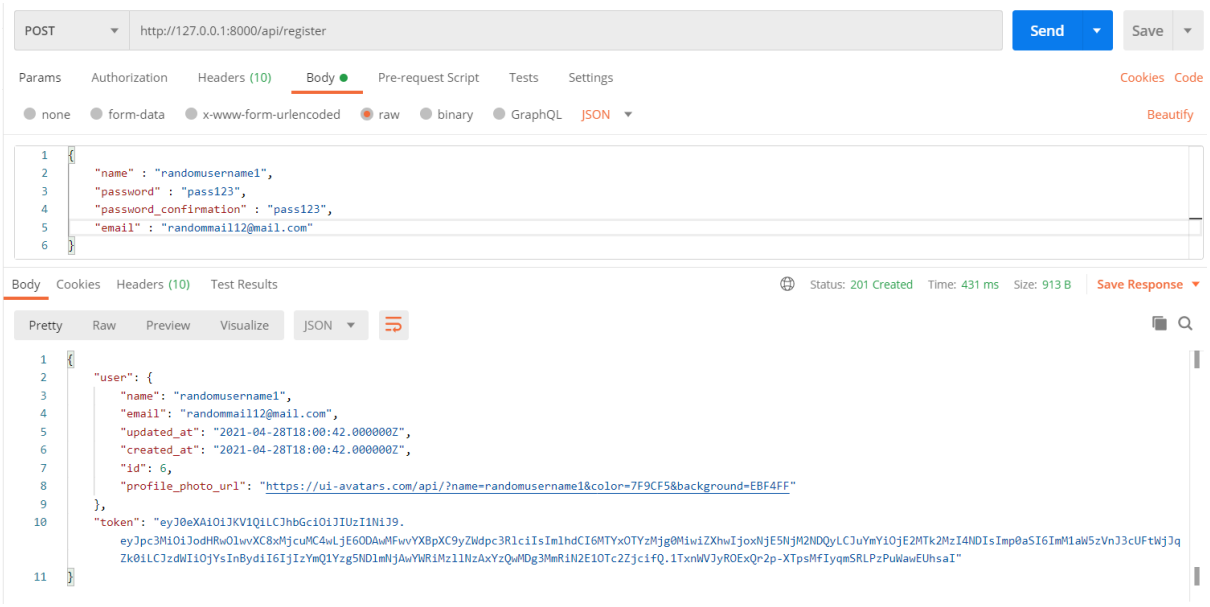
Para su creación se ha utilizado el comando: `php artisan make:auth` Lo que hace este comando es crear todas las vistas necesarias para la autenticación y las coloca en el directorio resources / views / auth. El comando make: auth también crea un directorio de recursos / vistas / diseños que contiene un diseño base para su aplicación.

Esto junto a la conectividad con la BBDD del paso anterior nos da el resultado. Para hacer comprobaciones hemos utilizado Postman cómo se va a mostrar a continuación.

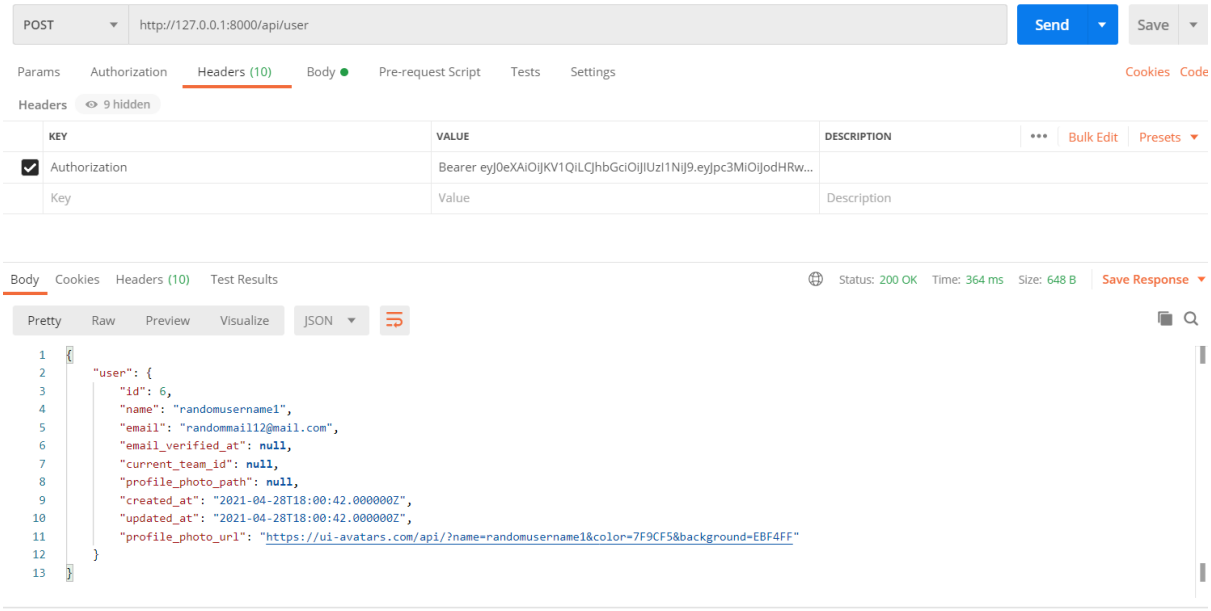
Carga de datos desde API

Para comprobar que todo funciona correctamente se ha realizado una carga de datos desde Postman. Postman es una herramienta que funciona como un API REST.

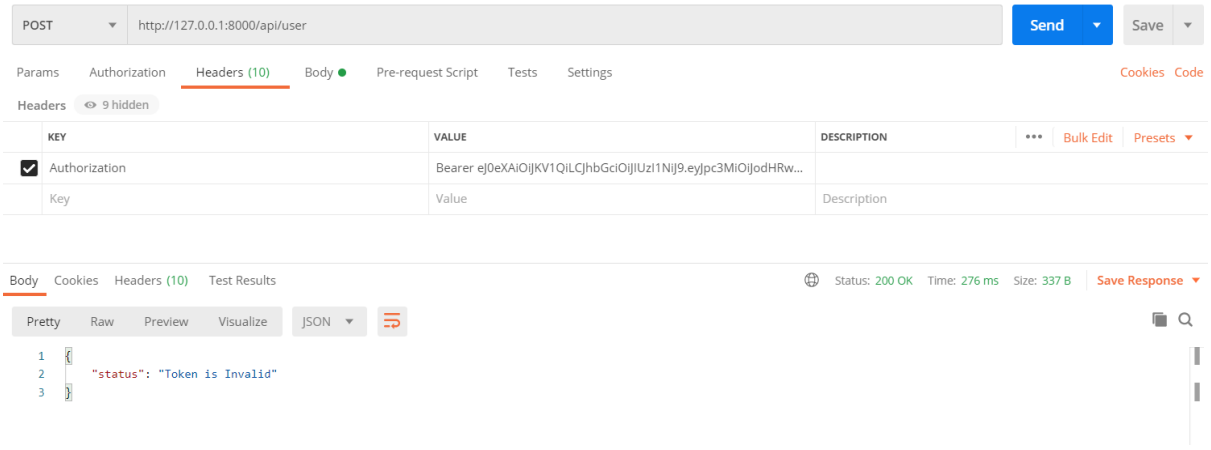
Para comprobar el funcionamiento vamos a registrar un nuevo usuario. Accedemos al endpoint “api/register” de nuestro archivo api.php para registrar un nuevo usuario en la BBDD, enviamos el JSON con los datos y nos devuelve un JSON con el registro del usuario en la tabla y el token asignado.



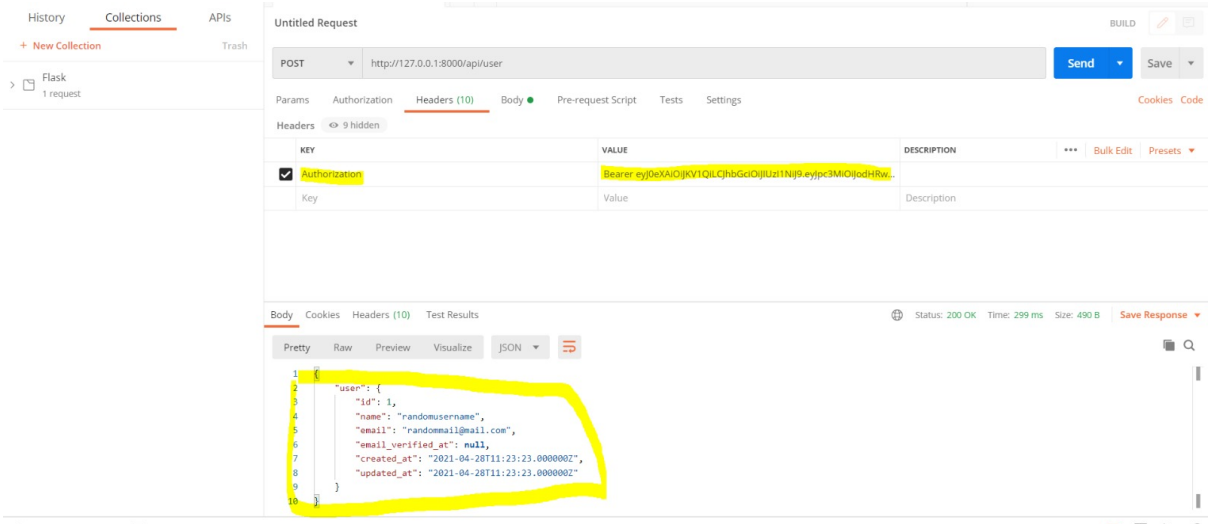
El segundo paso para la comprobación del correcto funcionamiento es acceder al endpoint “/api/login” para acceder con el usuario creado anteriormente, esto nos dará como respuesta el token asignado a ese usuario.



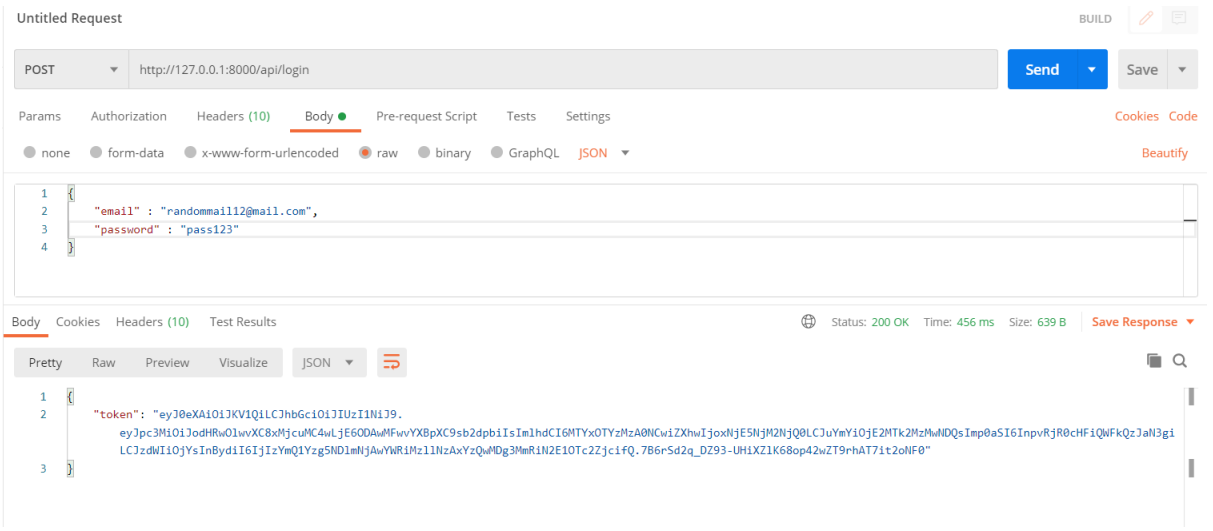
Como prueba adicional para asegurar el funcionamiento si modificamos algún carácter del token asignado nos devuelve un error con el mensaje “Token is invalid.”



Como vemos introducimos la ruta de Laravel completa más la ruta del user. La llamada que realizamos es una llamada post en la que le pasamos una variable que va a ser el email y una contraseña (nombre y contraseña).



Esto nos devuelve el id del usuario creado (primary key) y los datos introducidos más datos como el timestamp de cuando se ha creado esa cuenta o cuando se ha verificado. Como vemos el valor está cifrado por lo tanto la autenticación de tokens es válida.



Por último, para acceder al perfil del usuario utilizamos el endpoint “/api/user” y especificamos en la cabecera la “KEY”:Authorization y el “

Ejecución de los scrappers

Se han implementado los métodos para obtener, llamar y ejecutar los scrappers que luego son guardados en la BBDD. Los scrapper se encuentran en el proyecto dentro de una carpeta llamada Scrappers.

El principal problema que hemos tenido es acceder a la información de 2 de las fuentes de datos (TripAdvisor y Airbnb). Estas dos páginas webs cambian continuamente el nombre de la clase de sus variables para que no se realicen procesos de web scraping. Así mismo se modifica la estructura para impedir el acceso mediante Selenium con los xpath. Es por esto que actualmente no se recogen correctamente los datos pese a que antes sí se podía.

Se han desarrollado los 3 scrappers funcionales pero a la hora de implementar esta entrega solo funcionaba la fuente de datos del tiempo. Se solucionó la fuente TripAdvisor por lo que la solución está implementada. A la hora de realizar la memoria de la entrega hemos visto que la página web ha vuelto a cambiar por lo que no está operativo. Los 3 scrapers se ejecutarían al hacer las búsquedas para obtener una información actualizada.

Una cosa a destacar común es la conexión con la base de datos. Para establecerla todas las funciones utilizan la siguiente función

```php
public function conexionBD(){
    $hostname = "2.139.176.212";
    $username = "pr_grupoa";
    $password = "pr_grupoa";
    $db = "prgrupoa";
    $dbconnect=mysqli_connect($hostname,$username,$password,$db);

    if ($dbconnect->connect_error) {
        print("Fallo al conectarse a la base de datos: ". $dbconnect->connect_error);
    }else{
        return $dbconnect;
    }
}
```

Scraper tiempo

Se ha declarado las rutas para que la API pueda recoger los datos en el archivo api.php situado en: PC2-3 , routes

```php
Route::post('buscar.tiempo', 'App\Http\Controllers\Controller@scraperTiempo');
```

La API ejecuta la función scraperTiempo() que se muestra y explica a continuación.

- Recoge el nombre de el municipio buscado
- Ejecuta el scrapper
- Establece la conexión con la base de datos

```php
public function scraperTiempo(Request $request)
{
    $arg = $request->input('name');
    $command = "C:\Users\isabe\Anaconda3\python.exe C:\\xampp\\htdocs\\PC3\\PythonAPI\\Scrapers\\tiempo.py " . escapeshellarg($arg);
    $result = exec($command);
    $dbconnect=$this->conexionBD();
```

- Se gestiona el resultado del scrapper y se extrae la fecha de ayer con la librería carbon

```php
$result = utf8_encode($result);
$result = json_decode($result,true);
$fecha_carbon = new Carbon("yesterday");
```

- Extrae de la BDD el id correspondiente al municipio buscado

```php
$idMunicipio=$result[0]["idMunicipio"];
$query = mysqli_query($dbconnect,"SELECT id FROM municipios WHERE Nombre = '$idMunicipio'");
$row = mysqli_fetch_assoc($query);
$id = $row['id'];
```

- Se realiza un bucle for que recorra todos los datos donde:
    - Se añade 1 día a la fecha. En el primer resultado el día se convierte en el actual, por eso antes se determina que extraiga el día de ayer mediante Carbon. Posteriormente se formatea para que no tenga las horas, minutos y segundos.

```
foreach($result as $value){
    $fecha_carbon = $fecha_carbon->addDays(1);
    $fecha_carbon_formateada = $fecha_carbon->format('Y-m-d');
```

- Se extrae la información:

```
$idMunicipio=$value["idMunicipio"];
$Nombre=$value["Nombre"];
$Fecha=$value["Fecha"];
$tMaxima=$value["tMaxima"];
$tMinima=$value["tMinima"];
$tMedia=$value["tMedia"];
$Humedad=$value["Humedad"];
$Presion=$value["Presion"];
$Viento=$value["Viento"];
```

- Se intenta obtener datos que coincidan con la información obtenida para no meter en la BDD información duplicada. Para ello se intenta sacar el nombre del ocio para el ocio obtenido en el scrapper para el pueblo analizado

```
$query_comprobacion =mysqli_query($dbconnect,"SELECT Fecha FROM climas WHERE (idMunicipio = '$id') AND (Fecha = '$fecha_carbon_formateada')");
$row_select_fecha = mysqli_fetch_assoc($query_comprobacion);
$fecha_select = $row_select_fecha['Fecha'];
```

- En el caso de que exista ese dato en la BDD en ese municipio se actualiza

```
if($fecha_select == $fecha_carbon_formateada ){
    $query_update =mysqli_query($dbconnect,"UPDATE climas SET tMaxima = '$tMaxima' , tMinima = '$tMinima', tMedia = '$tMedia',
        Humedad = '$Humedad', Presion = '$Presion', Viento = '$Viento' WHERE (idMunicipio = '$id') AND (Fecha = '$fecha_carbon_formateada')");
    //echo "Datos actualizados - ";
}
```

- En el caso de que la información no concuerda por que no se ha podido extraer información de algo inexistente se guardan los datos en la BDD.

```
else{
    $query2 = mysqli_query($dbconnect,"INSERT INTO climas (idMunicipio,Fecha,tMaxima,tMinima,tMedia,Humedad,Presion,Viento)
        VALUES ('$id', '$fecha_carbon','$tMaxima','$tMinima','$tMedia','$Humedad','$Presion','$Viento')");
    //echo "Nuevos datos - ";
}
```

- Finalmente se devuelve el resultado del scrapper

Scrapper TripAdvisor

Se ha declarado las rutas para que la API pueda recoger los datos en el archivo api.php situado en: PC2-3 , routes

```
Route::post('buscar.tripadvisor', 'App\Http\Controllers\Controller@scraperTripAd');
```

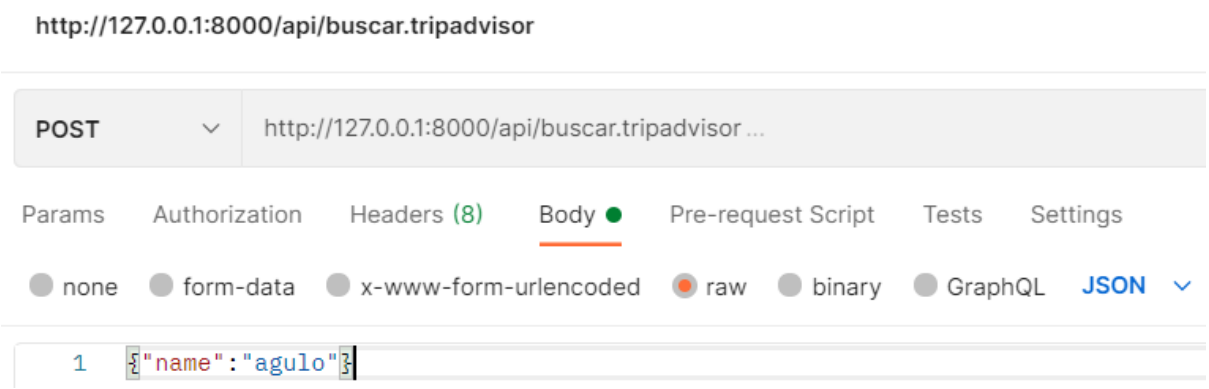
La API ejecuta la función scraperTripAd() que se muestra y explica a continuación.

```php
public function scraperTripAd(Request $request){
    $vArg = $request->input('name');
    $dbconnect=$this->conexionBD();
    $queryid = mysqli_query($dbconnect,"SELECT id FROM municipios WHERE Nombre = '$vArg'");
    $filaid = mysqli_fetch_assoc($queryid);
    $id = $filaid['id'];
    $querysentimiento = mysqli_query($dbconnect,"SELECT AnalisisSentimiento FROM busquedas WHERE created_at between DATE_ADD(now(),INTERVAL -7 DAY)
    and now() and idMunicipio = '$id' and Scraper =1");
    if ($querysentimiento -> num_rows==0){
       $resultado = $this->scraperTripAdyCommsParam($vArg);
       $sentimiento = $resultado["analisis sentimiento"];
       $queryinsertsentimiento = mysqli_query($dbconnect,"INSERT INTO busquedas (idMunicipio,AnalisisSentimiento,created_at,updated_at,Scraper)
                VALUES ('$id', '$sentimiento',now(),now(),'1')");
    }else{
        $filasentimiento = mysqli_fetch_assoc($querysentimiento);
        $queryocio = mysqli_query($dbconnect,"SELECT Nombre FROM ocios o WHERE o.idMunicipio = '$id'");
        while($filasocio =mysqli_fetch_assoc($queryocio)){
            $ocio[]=$filasocio;
        }
        $queryrestaurantes = mysqli_query($dbconnect,"SELECT Nombre, Detalles FROM restaurantes r WHERE r.idMunicipio = '$id'");
        while($filasrestaurantes =mysqli_fetch_assoc($queryrestaurantes)){
            $restaurantes[]=$filasrestaurantes;
        }
        $queryhoteles = mysqli_query($dbconnect,"SELECT Nombre, Descripcion, Caracteristicas FROM hotels h WHERE h.idMunicipio = '$id'");
        while($filashoteles =mysqli_fetch_assoc($queryhoteles)){
            $hoteles[]=$filashoteles;
        }
        $querysentimientonuevo = mysqli_query($dbconnect,"SELECT AnalisisSentimiento FROM busquedas b WHERE b.idMunicipio = '$id' LIMIT 1");
        while($filassentimiento =mysqli_fetch_assoc($querysentimientonuevo)){
            $sentimiento[]=$filassentimiento;
            $analisissentimiento=$filassentimiento['AnalisisSentimiento'];
            $queryinsertarsentimiento = mysqli_query($dbconnect,"INSERT INTO busquedas (idMunicipio,AnalisisSentimiento,created_at,updated_at,Scraper)
            VALUES ('$id', '$analisissentimiento',now(),now(),'0')");
        }

        $resultado=array_merge($ocio,$restaurantes,$hoteles,$sentimiento);
    }
    return $resultado;
}
```

Esta función recoge el id del municipio que le pasa el usuario y ve si en la tabla búsquedas hay algún registro con ese id entre la fecha de hoy y hace una semana. También tiene la condición si el scraper es igual a 1, ya que para hacer las búsquedas populares necesitábamos ir ingresando en la tabla nuevas búsquedas y si siempre estábamos ingresando búsquedas nunca ejecutaría el scraper. Si hay alguna búsqueda en la semana anterior con el scraper a 1 entonces no ejecuta el scraper y lo que hace es un select de los datos de las tres tablas del scraper de tripadvisor, y tambien mete en búsquedas un nuevo registro pero con el scraper a 0 porque no se ha ejecutado. Si no hay ningún registro en la semana anterior con scraper a 1 entonces ejecuta el scraper e inserta los nuevos datos en la base de datos y añade un nuevo registro con el campo de Scraper a 1 porque esta vez sí se ha ejecutado el scraper.

Para su ejecución se necesita un parámetro que se introducirá en el front end. Desde el backend hemos de utilizar una aplicación llamada Postman para poder ejecutarla.



Los pasos que sigue son:

Dentro del scraper de Tripadvisor lo que hacemos es seleccionar todos los comentarios de cada hotel, punto de ocio y restaurantes del pueblo y lo pasamos por el análisis de sentimiento con la librería textblob. Una vez tenemos todos los comentarios con sus análisis de sentimiento los sumamos y los dividimos entre el índice (número de comentarios que hay) para así obtener un análisis de sentimiento global.

```
    for e in vComentarios:
        vC1 = e.find("q") or e.find("p", {"class":"partial_entry"})
        try:
            vAnalisis = TextBlob(vC1.text).translate(to='en')
        except:
            vAnalisis = TextBlob(vC1.text)
        vAnalisis1 = vAnalisis.polarity#).replace("Sentiment(polarity=","").replace(" subjectivity=","").replace(")","")
        aSentimiento.append(float(vAnalisis1))
        #print(vAnalisisF)
        #print("----------------------------------------")
        #print(vC1.text)
        #print('########################')
        #print(vContador)
        vContador+=1

#print(aSentimiento)
#print(vContador)
vTotal = sum(aSentimiento)
vTotalF = float("%.2f" % vTotal)
#print(vTotal)
#print(vTotalF)
vAnalisisSentimientoFinal = vTotalF/vContador
#print(vAnalisisSentimientoFinal)
vAnalisisSentimientoFinalF = float("%.2f" % vAnalisisSentimientoFinal)
vJsonGlobal = {'lugares':vJSON, 'analisis sentimiento':vAnalisisSentimientoFinalF}
print(json.dumps(vJsonGlobal))
```

Se llama a un a función llamada scraperTripAdyCommsParam() donde se ejecuta el scrapper:

```
public function scraperTripAdyCommsParam($vArg){
    set_time_limit (5000);
    $command = "C:\Users\isabe\Anaconda3\python.exe C:\\xampp\\htdocs\\PC3\\PythonAPI\\Scrapers\\TripAdFinal.py " . escapeshellarg($vArg);
    $result = exec($command);
    $result = utf8_encode($result);
    $result = json_decode($result,true);

    $dbconnect=$this->conexionBD();

    $Municipio = $result["lugares"]["ocio"][0]["Municipio"];
    $query = mysqli_query($dbconnect,"SELECT id FROM municipios WHERE Nombre = '$Municipio'");
    $row = mysqli_fetch_assoc($query);
    $idMunicipio = $row['id'];


    //insert de los sitios
    $this->insert_localizaciones_TripAd($result, $vArg, $idMunicipio, $dbconnect);

    return $result;
}
```

El proceso que realiza es:

- Establece que el tiempo de ejecución para que el scraper de Tripadvisor pueda recoger todos los datos (ya que el proceso es muy lento)
- Ejecuta el scrapper
- Establece conexión con la base de datos (BDD).
- Extrae de la BDD el id correspondiente al municipio buscado

- LLama a una función que se va a explicar a continuación donde se insertan los datos en la BDD llamada insert_localizaciones_TripAd()
- Devuelve el resultado

Se llama a un a función llamada insert_localizaciones_TripAd() donde se introducen los datos en la BDD. Se le pasan como parametros el resultado en formato json de los datos encontrados, el nombre del pueblo analizado, el id del municipio analizado y la conexión a la BDD anteriormente establecida. Primeramente se realiza un proceso donde se guarda la información de cada tabla de la BDD en 3 json (json_ocio, json_hoteles y json_restaurantes).

```
public function insert_localizaciones_TripAd($result, $vArg, $idMunicipio, $dbconnect){
    $json_ocio = $result["lugares"]["ocio"];
    $json_hoteles = $result["lugares"]["hoteles"];
    $json_restaurantes = $result["lugares"]["restaurantes"];
```

Posteriormente se realiza el mismo proceso para cada json. Se realiza un bucle for para recorrer todos los elementos obtenidos. Se va a explicar el proceso realizado dentro del bucle con los datos relacionados con el ocio:

- Se recogen los valores obtenidos

```
foreach($json_ocio as $value){
    $nombre_ocio = $value["Nombre"];
    $referencia_ocio = $value["Referencia"];
```

- Se intenta obtener datos que coincidan con la información obtenida para no meter en la BDD información duplicada. Para ello se intenta sacar el nombre del ocio para el ocio obtenido en el scrapper para el pueblo analizado.

```
$query_comprobacion_ocio =mysqli_query($dbconnect,"SELECT Nombre FROM ocios WHERE (idMunicipio = '$idMunicipio') AND
    (Nombre = '$nombre_ocio')");
$row_nombre_ocio = mysqli_fetch_assoc($query_comprobacion_ocio);
$nombre_select = $row_nombre_ocio['Nombre'];
```

- En el caso de que exista ese ocio en la BDD en ese municipio se actualizan los datos

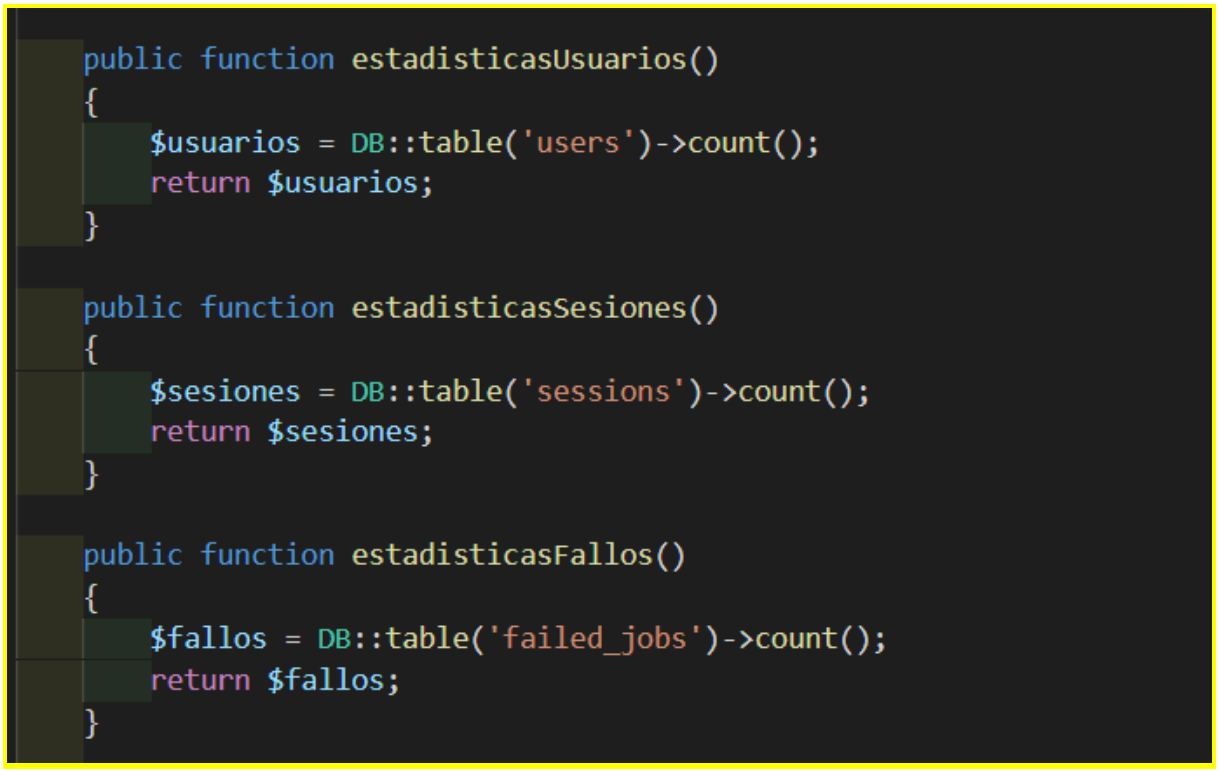
```
if($nombre_select == $nombre_ocio){
    $query_update =mysqli_query($dbconnect,"UPDATE ocios SET Referencia = '$referencia_ocio' WHERE (idMunicipio = '$idMunicipio') AND
        (Nombre = '$nombre_ocio')");
    //echo "Datos ocio actualizados - ";
}
```

- En el caso de que la información no concuerda por que no se ha podido extraer información de algo inexistente se guardan los datos en la BDD.

```
else{
    $query2 = mysqli_query($dbconnect,"INSERT INTO ocios (idMunicipio,Nombre,Referencia)
        VALUES ('$idMunicipio', '$nombre_ocio','$referencia_ocio')");
    //echo "Nuevos datos ocio - ";
}
```

Estadísticas en controller

Estas tres llamadas lo que hacen es un count de cuantos usuarios, sesiones y trabajos erróneos hay en la base de datos.



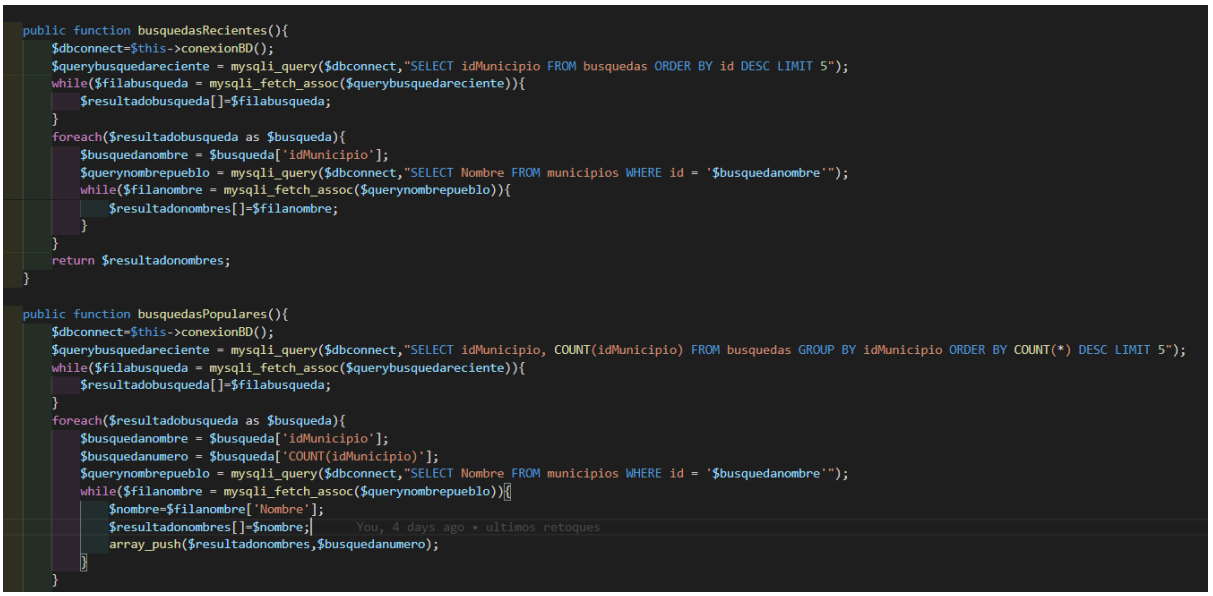
```php
public function estadisticasUsuarios()
{
    $usuarios = DB::table('users')->count();
    return $usuarios;
}

public function estadisticasSesiones()
{
    $sesiones = DB::table('sessions')->count();
    return $sesiones;
}

public function estadisticasFallos()
{
    $fallos = DB::table('failed_jobs')->count();
    return $fallos;
}
```

Búsquedas en controller

Las llamadas de busquedas es para implementar nuevas funcionalidades de los 5 pueblos buscados recientemente y los 5 pueblos mas populares con el numero de busquedas totales. Esto es una funcionalidad que se la añadiremos al usuario que se registre, para que haya diferencia entre el usuario registrado y el no registrado.



```php
public function busquedasRecientes(){
    $dbconnect=$this->conexionBD();
    $querybusquedareciente = mysqli_query($dbconnect,"SELECT idMunicipio FROM busquedas ORDER BY id DESC LIMIT 5");
    while($filabusqueda = mysqli_fetch_assoc($querybusquedareciente)){
        $resultadobusqueda[]=$filabusqueda;
    }
    foreach($resultadobusqueda as $busqueda){
        $busquedanombre = $busqueda['idMunicipio'];
        $querynombrepueblo = mysqli_query($dbconnect,"SELECT Nombre FROM municipios WHERE id = '$busquedanombre'");
        while($filanombre = mysqli_fetch_assoc($querynombrepueblo)){
            $resultadonombres[]=$filanombre;
        }
    }
    return $resultadonombres;
}

public function busquedasPopulares(){
    $dbconnect=$this->conexionBD();
    $querybusquedareciente = mysqli_query($dbconnect,"SELECT idMunicipio, COUNT(idMunicipio) FROM busquedas GROUP BY idMunicipio ORDER BY COUNT(*) DESC LIMIT 5");
    while($filabusqueda = mysqli_fetch_assoc($querybusquedareciente)){
        $resultadobusqueda[]=$filabusqueda;
    }
    foreach($resultadobusqueda as $busqueda){
        $busquedanombre = $busqueda['idMunicipio'];
        $busquedanumero = $busqueda['COUNT(idMunicipio)'];
        $querynombrepueblo = mysqli_query($dbconnect,"SELECT Nombre FROM municipios WHERE id = '$busquedanombre'");
        while($filanombre = mysqli_fetch_assoc($querynombrepueblo)){
            $nombre=$filanombre['Nombre'];
            $resultadonombres[]=$nombre;      You, 4 days ago • ultimos retoques
            array_push($resultadonombres,$busquedanumero);
        }
    }
```

Graficar fechas en controller

Esta funcionalidad es para el administrador, con sentencias sql traemos el número de usuarios que hay y en qué fecha se han registrado para luego poder plasmar esos datos en una gráfica y ver como va creciendo la actividad del proyecto.

```php
public function graficaFechas()
{
    $aFechas = [];
    $data = DB::table('users')->get();
    $data = json_encode($data);
    $data = json_decode($data, true);
    foreach ($data as $value){
        $fecha = $value['created_at'];
        $fecha = explode(" ", $fecha);
        $fecha = $fecha[0];
        //print($fecha . " / ");
        array_push($aFechas, $fecha);
    }
    $aFinal = array_count_values($aFechas);
    return(array_slice($aFinal, -5));
}
```

Metodología

Este proyecto se ha realizado mediante la metodología agile Scrum. Se han ido realizando las tareas en sprints de una corta duración. Al final de cada sprint se ha proporcionado al cliente (el profesor de la asignatura) un informe demostrativo y explicativo de todas las funcionalidades implementadas. El cliente ofrecía su feedback con lo que el equipo de desarrollo ha podido modificar y adaptar el proyecto en todo momento.
Las tareas que se han ido realizando han estado siempre actualizadas en un documento en google drive compartido por el equipo donde se encontraba el diagrama de Gantt.

Herramientas empleadas y recursos requeridos

1. **Xampp**: Se necesita instalar Xampp ya que tiene una versión de PHP mayor que 7.2 (requerimiento de laravel) junto con Apache, MariaDB y Perl.

2. **Visual Studio Code:** Es un editor de texto desde el cual vamos a ejecutar el servidor de nuestro proyecto por línea de comandos. Así mismo necesitamos tener este editor de texto para adaptar las rutas como se detalla a continuación.

3. **Anaconda:** Software libre de código abierto para el uso de los lenguajes Python y R. Necesitamos esta aplicación para la ejecución de los scripts hechos en Python que realizan web scrapping.

4. **Google Chrome**: Necesitamos tener Google Chrome. Este va a ser el navegador desde el cual se realiza el proceso de web scrapping.

5. **Chromedriver**: Se ha de descargar un driver de Google Chrome para el uso de Selenium a la hora de hacer web scrapping.

6. **Composer**: Composer es un sistema gestor de paquetes para programar en PHP.

7. **Laravel**: Laravel es un framework de código abierto para desarrollar aplicaciones web. Su principal ventaja es que, al utilizarse, se evita el “código espagueti”.

8. **GitHub**: En Github se aloja este proyecto.

9. **WinRAR**: Software que nos sirve para descomprimir el fichero donde se encuentra el proyecto (comprimido en formato ZIP).

Presupuesto

En cuanto al tiempo de desarrollo empleado en la realizacion e implementacion del proyecto de PC1 lógicamente no hicimos uso de la totalidad del tiempo de impartición de la asignatura debido a que necesitamos un tiempo para familiarizarnos con los conceptos y herramientas nuevas que íbamos a utilizar posteriormente en el desarrollo.

Una vez realizado el aprendizaje tuvimos alrededor de 1 mes para el desarrollo completo del cual empleamos cada uno unas 3 horas de clase más 6 horas de manera autónoma a la semana, esto hace un total de 36 horas de trabajo para completar el proyecto de PC1.

En PC2 y PC3 se han necesitado muchisimas mas horas ya que, aparte de ser dos asignaturas en vez de una, las asignaturas requieren de un desarrollo mayor al de PC1. Entre PC2 y PC3 trabajamos las 8 horas de clases semanales más 10 horas de trabajo autónomo. El desarrollo de la aplicación dura desde el 15 de febrero al 15 de junio (3 meses teniendo en cuenta que el 1º se dedica a PC1). 18 horas semanales hacen 72 horas al mes, lo que suma un total de 216 horas

En total del semestre,haremos aproximadamente unas 252 horas.

Mano de obra			
Tipo	Cantidad de horas	Costo por hora	Total
Diseño (Data Engineer en Madrid)	18	19,02€	345,6€
Desarrollo (Backend Junior en Madrid)	198	12,94€	2562,12€
Análisis de información (Data Engineer en Madrid)	18	19,02€	345,6€
Testing (Junior Tester en España)	18	14,7€	264,6,4€
Total			3598.92

Hardware			
Tipo	Precio	Cantidad	Total
Computadora	1200€	3	3600€
Total			3600€

Total: 7.198,92

Conclusiones

Al ir desarrollando el proyecto hemos ido viendo el alcance del mismo. Se han cambiado las tareas para abarcar mejor el proyecto y se han determinado los períodos de tiempo con períodos de tiempo reales (no por semanas). Se habían generalizado mucho las etapas

Hemos tenido numerosos problemas con las parte de realizar los scrappers. Las páginas web de tripadvisor y airbnb están en continuo cambio por lo que ha sido una tarea imposible. Se ha tenido que cambiar el scrapper de tripadvisor a lo largo del desarrollo al menos 5 veces y el de airbnb no ha llegado a ser posible adaptarlo por tiempos de desarrollo. Es por ello que hemos tenido que descartar esta funcionalidad del software.

Si este proyecto pudiera ser funcional y viable en un futuro podría llegar a ser una aplicación revolucionaria en el mercado, pero actualmente no lo es porque las páginas web pueden volver a cambiar.

Futuras líneas de trabajo

Las futuras líneas de trabajo son:

- Encontrar sitios web de los que extraer información que no sean cambiantes
- Adaptar el diseño del proyecto a las nuevas fuentes de datos
- Adaptar el código del proyecto a las nuevas fuentes de datos
- Adaptar la base de datos del proyecto a las nuevas fuentes de datos
- Programar una tarea mediante crontab para que se pueda recoger la información automáticamente.
    - En el caso de la información del tiempo sería hacer una tarea periódica que hiciera el scrapping todos los domingos de cada semana para que no se vuelvan a incluir el mismo día más en la media total. Una vez que tenemos las medias de las temperaturas de los distintos pueblos lo que se haría es hacer unos rangos para determinar si el tiempo es Mediterraneo, Seco, LLuvioso o Frío. También se tendrá en cuenta las precipitaciones y la nieve para poder clasificar un pueblo en una de estas categorías

Anexos

Manual de instalación

Instalación de programas y otras descargas

10. **Xampp**: Se necesita instalar Xampp ya que tiene una versión de PHP mayor que 7.2 (requerimiento de laravel) junto con Apache, MariaDB y Perl.

Para instalar Xampp vamos a utilizar el siguiente link (https://www.apachefriends.org/es/download.html) y seleccionar la opción mejor para el ordenador desde el cual se va a ejecutar.

11. **Visual Studio Code:** Es un editor de texto desde el cual vamos a ejecutar el servidor de nuestro proyecto por línea de comandos. Así mismo necesitamos tener este editor de texto para adaptar las rutas como se detalla a continuación.

Para instalar Visual Studio Code vamos a utilizar el siguiente link (https://code.visualstudio.com/download) y seleccionar la opción mejor para el ordenador desde el cual se va a ejecutar.

12. **Anaconda:** Software libre de código abierto para el uso de los lenguajes Python y R. Necesitamos esta aplicación para la ejecución de los scripts hechos en Python que realizan web scrapping.

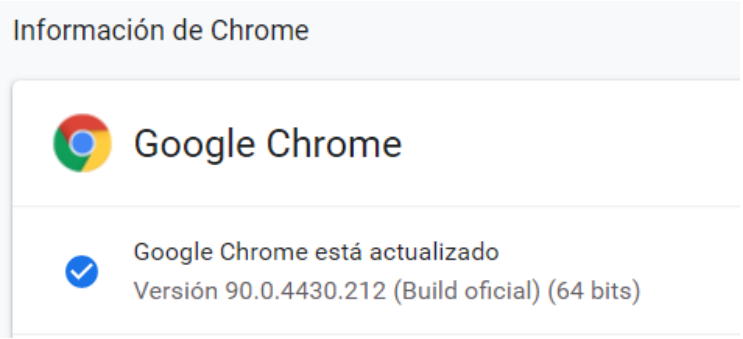
Para instalar Anaconda vamos a utilizar el siguiente link (https://www.anaconda.com/products/individual#Downloads) y seleccionar la opción mejor para el ordenador desde el cual se va a ejecutar.

13. **Google Chrome**: Necesitamos tener Google Chrome. Este va a ser el navegador desde el cual se realiza el proceso de web scrapping.

Para instalar Google Chrome en caso de no tenerlo vamos a utilizar el siguiente link (https://www.google.com/intl/es/chrome/?brand=CHBD&gclid=CjwKCAjwhYOFBhBkEiwASF3KGe2380IirSJvV-Q8D7cBlveloVHPx_Z8_19Ab317raquyklTSwkDexoC7rAQAvD_BwE&gclsrc=aw.ds)

14. **Chromedriver**: Se ha de descargar un driver de Google Chrome para el uso de Selenium a la hora de hacer web scrapping.

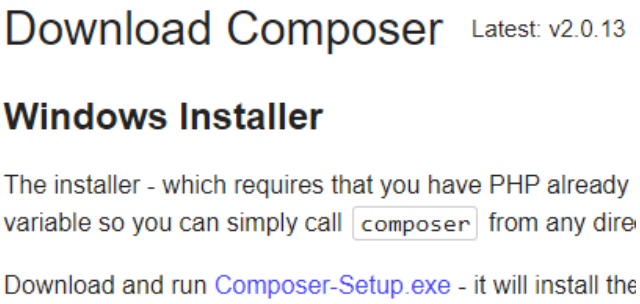
Se ha de descargar la versión de driver que coincide con tu versión de Google Chrome. Para saber la versión has de ir a Personaliza y controla Google Chrome (los 3 puntos situados en la esquina superior derecha). Ahí hemos de seleccionar: Ayuda - Información de Chrome



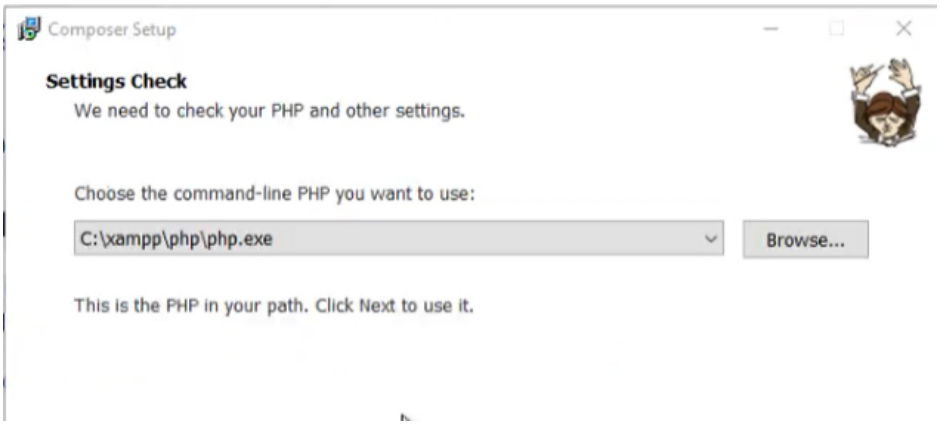
Para descargar el driver vamos a utilizar el siguiente link (https://chromedriver.chromium.org/downloads) y seleccionar la opción que indica la versión que hemos visto que tenemos.

15. **Composer**: Composer es un sistema gestor de paquetes para programar en PHP.

Para descargar Composer vamos a utilizar el siguiente link (https://getcomposer.org/download/) y seleccionar “Composer-Setup.exe”.

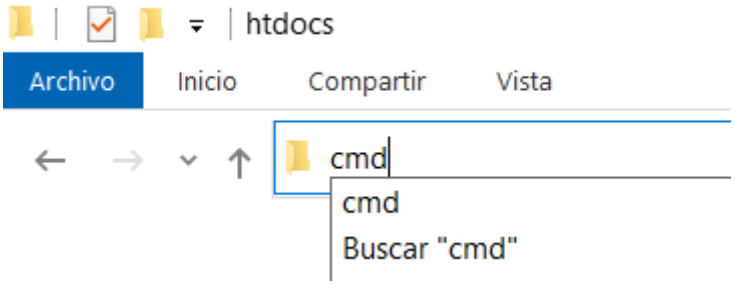


Cuando estamos realizando el proceso de instalación tenemos que indicarle la ruta de nuestro ordenador donde tenemos instalado nuestro PHP (instalado en Xampp):



16. **Laravel**: Laravel es un framework de código abierto para desarrollar aplicaciones web. Su principal ventaja es que, al utilizarse, se evita el “código espagueti”.

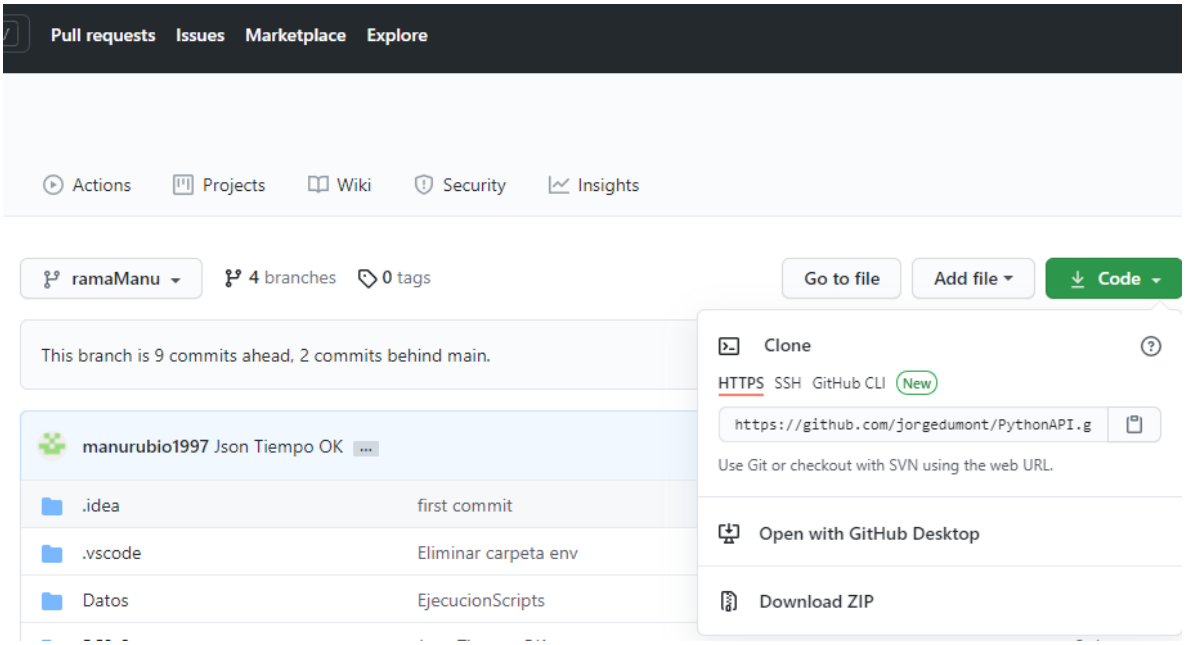
Para instalarlo hemos de ir acceder a la línea de comandos de nuestro ordenador desde la ruta de la carpeta htdocs dentro de Xampp. Para ello solo tenemos que acceder a la carpeta desde la aplicación de archivos y poner “cmd” en donde se ve la ruta.



Desde ahí introducimos la siguiente instrucción: `composer global require "laravel/installer=~1.1"`

17. **Descargar el proyecto del repositorio en GitHub**: Hemos de descargar el proyecto del repositorio y alojarlo en la carpeta htdocs de Xampp.

Para ello vamos a ir al siguiente enlace (https://github.com/jorgedumont/PythonAPI/tree/main) y seleccionar “Code” - “Download Zip”

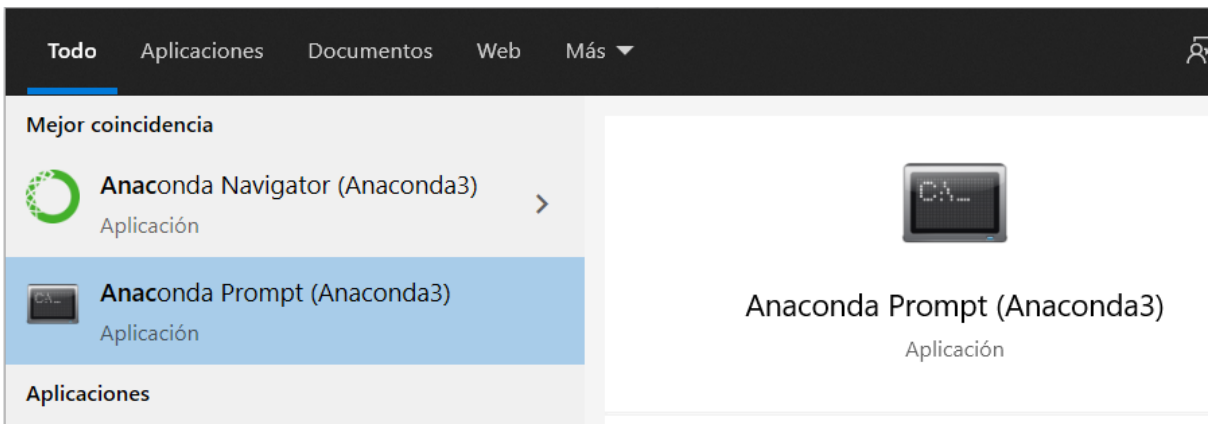


18. **WinRAR**: Software que nos sirve para descomprimir el fichero donde se encuentra el proyecto (comprimido en formato ZIP).

Para instalar WinRAR vamos a utilizar el siguiente link (https://www.winrar.es/descargas) y seleccionar la opción mejor para el ordenador desde el cual se va a ejecutar.

Instalación de librerías en Anaconda

Para instalar las librerías necesarias en Anaconda tenemos que acceder primeramente a la aplicación Anaconda Prompt. Así accedemos a la línea de comandos de Anaconda.



Desde la línea de comandos hemos de introducir 2 comandos para instalar las siguientes librerías para ejecutar los scripts de Python.

1. **BeautifulSoup**: biblioteca de Python para analizar documentos HTML

   Hemos de introducir el comando: `conda install beautifulsoup4`

2. **Pandas**: librería para el análisis de datos

   Hemos de introducir el comando: `conda install pandas`

Las demás librerías que se van a utilizar ya están instaladas.

Configuración de Laravel en Visual Studio

Para realizar la configuración de Laravel hemos de utilizar Anaconda. Desde la aplicación de Anaconda iniciamos la aplicación Visual Studio Code. Se abre el proyecto descomprimido (que se encuentra dentro de htdocs en Xampp).

Desde la línea de comandos hemos de instalar:

1. El instalador de Laravel con el siguiente comando: `composer require laravel/installer` Se actualiza con el siguiente comando: `composer global update laravel/installer`
2. Symfony: `composer require symfony/process`
3. El paquete jwt-auth : `composer require tymon/jwt-auth`
4. El paquete laravel ui: `composer require laravel/ui`
5. npm: `npm install`
6. Debemos ejecutar las siguientes instrucciones:
   a. `php artisan key:generate`
   b. `php artisan jwt:secret`
   c. `php artisan cache:clear`
   d. `php artisan config:clear`
7. Instalar el paquete jetstream: `composer require laravel/jetstream`
8. Instalar la librería carbon : `composer require nesbot/carbon`

Edición de las variables en Visual Studio

Para que el proyecto funcione correctamente hemos de cambiar algunas variables:

1. **Variables de entorno globales**: Todas las rutas que utilizamos tanto en los controladores (UserController.php y Controller.php) como los de la BBDD las declaramos como variables dentro de nuestro entorno (.env):

```
PYTHON_PATH=C:\\Users\\jdumo\\Anaconda3\\python.exe
TIEMPO_SCRIPT_PATH=C:\\Users\\jdumo\\OneDrive\\Escritorio\\Proyecto2\\Scrapers\\tiempo.py
TRIPADVISOR_SCRIPT_PATH=C:\\Users\\jdumo\\OneDrive\\Escritorio\\Proyecto2\\Scrapers\\TripAdFinal.py
```

   Para poder ejecutarlos tendremos que cambiar las rutas de las variables adecuándose a donde tenemos el proyecto. Las de la BBDD utilizamos toda la información que mostramos a continuación, y estas no habrá que cambiarlas:

```
DB_CONNECTION=mysql
DB_HOST=2.139.176.212
DB_PORT=3306
DB_DATABASE=prgrupoa
DB_USERNAME=pr_grupoa
DB_PASSWORD=pr_grupoa
```

   Las siguientes rutas tendremos que veremos tendremos que cambiarlas a mano ya que son archivos que se llaman desde los archivos de Python y desde ahi no podemos generar variables de entorno.

2. **Scrapper tiempo**: Hemos de ir al archivo llamado tiempo.py que se encuentra dentro de: Scrapers. En la línea 5 hay que poner la ruta desde tu ordenador a el archivo list-mun-2012.xls

```
49  def comprobarPueblo(nombrepueblo):
50      nombrespueblos = pd.read_excel("C:\\Users\\manu1\\GitHub\\PythonAPI\\Datos\\list-mun-2012.xls")
51      nombrespueblos["Municipio"] = nombrespueblos["Municipio"].str.lower()
52      nombrespueblos = nombrespueblos["Municipio"].tolist()
53      return nombrepueblo.lower() in nombrespueblos
```

3. **Scrapper TripAdvisor**:Hemos de ir al archivo llamado TripAd.py que se encuentra dentro de: Scrapers. En la línea 19 hemos de poner la ruta desde tu ordenador a el web driver que nos hemos descargado de Google Chrome. En la línea 244 hay que poner la ruta desde tu ordenador a el archivo list-mun-2012.xls

```
18      #Path con el ejecutor del driver
19      vDriverPath = "C:\\Users\\manu1\\Downloads\\chromedriver.exe"
20      vDriver = webdriver.Chrome(vDriverPath, chrome_options=vOptions)
21
```

```
243 def comprobarPueblo(nombrepueblo):
244     nombrespueblos=pd.read_excel("C:\\Users\\manu1\\GitHub\\PythonAPI\\Datos\\list-mun-2012.xls")
245     nombrespueblos["Municipio"]=nombrespueblos["Municipio"].str.lower()
246     nombrespueblos=nombrespueblos["Municipio"].tolist()
247     return nombrepueblo.lower() in nombrespueblos
```

4. **Scrapper Airbnb**:Hemos de ir al archivo llamado airbnb.py que se encuentra dentro de: Scrapers. En la línea 16 hemos de poner la ruta desde tu ordenador a el web driver que nos hemos descargado de Google Chrome. En la línea 251 hay que poner la ruta desde tu ordenador a el archivo list-mun-2012.xls

```
15
16          vDriverPath = "C:\\Users\\manu1\\Downloads\\chromedriver.exe"
17          vDriver = webdriver.Chrome(vDriverPath, options=vOptions)
18
```

```
250 def comprobarPueblo(nombrepueblo):
251     nombrespueblos=pd.read_excel("C:\\Users\\manu1\\GitHub\\PythonAPI\\Datos\\list-mun-2012.xls")
252     nombrespueblos["Municipio"]=nombrespueblos["Municipio"].str.lower()
253     nombrespueblos=nombrespueblos["Municipio"].tolist()
254     return nombrepueblo.lower() in nombrespueblos
255
```

Tag versión en repositorio

El enlace a nuestro repositorio en github es:
https://github.com/jorgedumont/PythonAPI/releases/tag/3.0
El archivo .sql se encuentra dentro del proyecto.