**FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION**

**DESARROLLO DE APLICACIONES WEB**

**I TÉRMINO 2019**

**TRABAJO AUTÓNOMO 3**

|  |  |
| --- | --- |
| **Nombre:** | Milton Fabrizio García Cox |
| **Paralelo:** | 2 |

1. **Descripción del conjunto de datos**

***Describa el conjunto de datos, cite la fuente y agregue la metadata[[1]](#footnote-1) del conjunto de datos. Agregue las capturas de pantalla que sean necesarias.***

Los datos que se van a utilizar para la realización del trabajo han sido extraídos de Kaggle, específicamente del sitio <https://www.kaggle.com/datafiniti/pizza-restaurants-and-the-pizza-they-sell>.

Dentro del sitio se expone un dataset con información de Restaurantes-Pizzerías ubicados en diferentes ciudades de Estados Unidos. El título original de la publicación es "Pizza Restaurants and the Pizza They Sell"[[2]](#footnote-2) (Datafiniti, 2019) con una lista de 3510 pizzas de diferentes restaurantes. Las variables o campos tomados en cuenta para cada Menú ofertado por las Pizzerias son los siguientes:

***id\_pizzeria***: identificador para cada pizzería que puede repetirse ya que una misma pizzería oferta diferentes especialidades o Menús.

***Address***: dirección física válida dentro del territorio norteamericano donde puede ser ubicado el restaurante.

***Categories***: define la clase de restaurante o sitio: Bar, Club, Restaurante General, Cafetería, Pastelería, entre otros.

***City***: ciudad ubicada dentro del territorio de Estados Unidos.

Country: únicamente se ha tomado en cuenta a Estados Unidos. Todos sus valores son “USA”

***Keys***: string plano de palabras clave que representan al restaurante con la combinación de ciudad, dirección y coordenadas.

***Latitude***: componente de latitud de las coordenadas geográficas.

***Longitude***: componente de longitud de las coordenadas geográficas.

***menuPageURL***: url del sitio web donde puede ser encontrado el menú e información general del restaurante.

***menusamountMax***: el valor más alto de la pizza de especialidad ofertada.

***menusamountMin***: el valor más bajo de la pizza de especialidad ofertada.

***Menuscurrency***: principal moneda que se maneja para los cobros. Por defecto es el dólar Americano.

***menusdateSeen***: fecha y hora de la publicación del menú más actualizado.

***Menusdescription***: breve descripción del menú y los ingredientes de la pizza.

***Menusname***: título de la pizza de especialidad.

***Name***: Nombre o razón social de la Pizzería.

***postalCode***: código postal de 5 dígitos o menos.

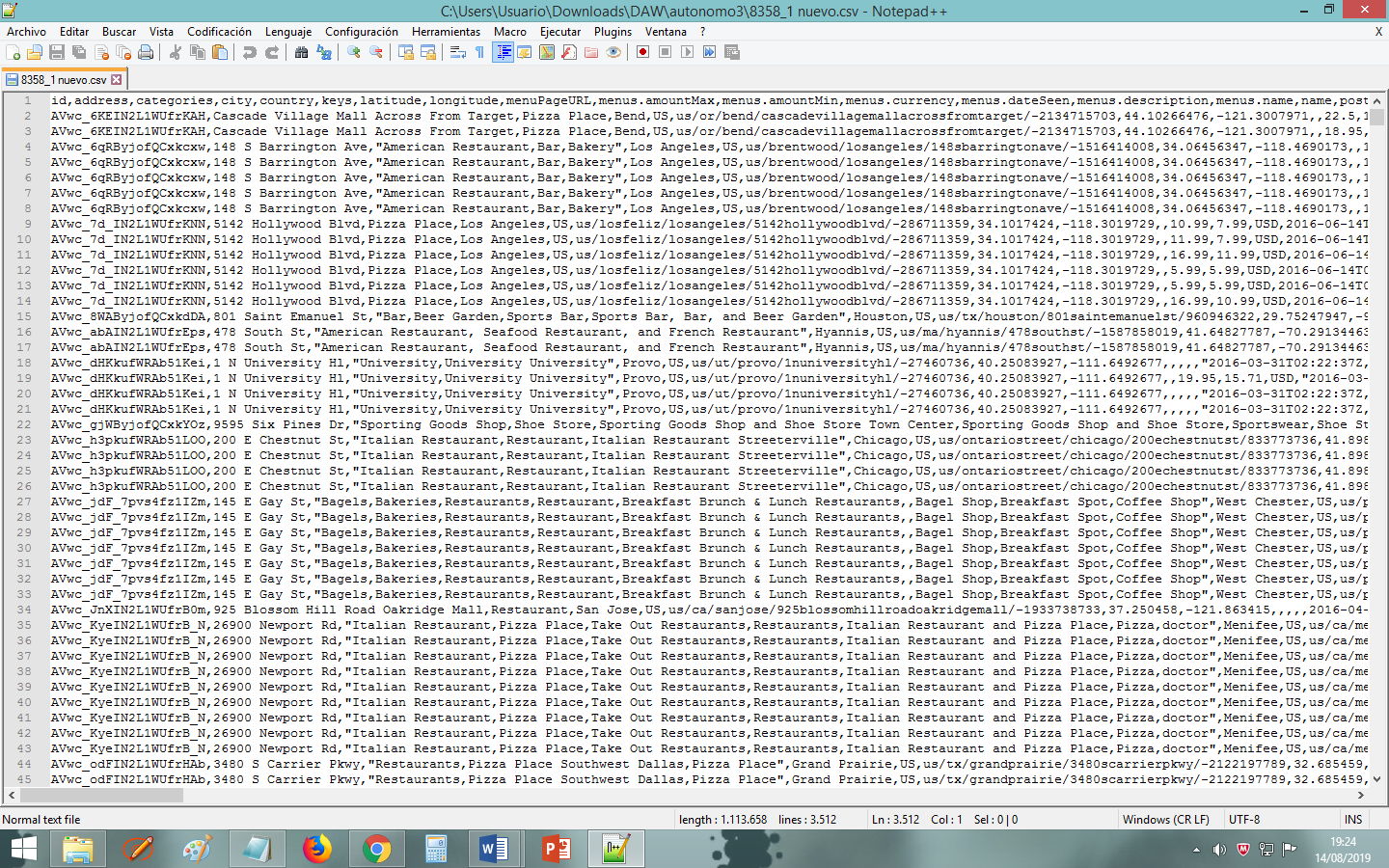
***priceRangeCurrency***: principal moneda que se maneja para los cobros del Menú. Por defecto es el dólar Americano.

***priceRangeMin***: ingresos diarios mínimos.

***priceRangeMax***: ingresos diarios máximos.

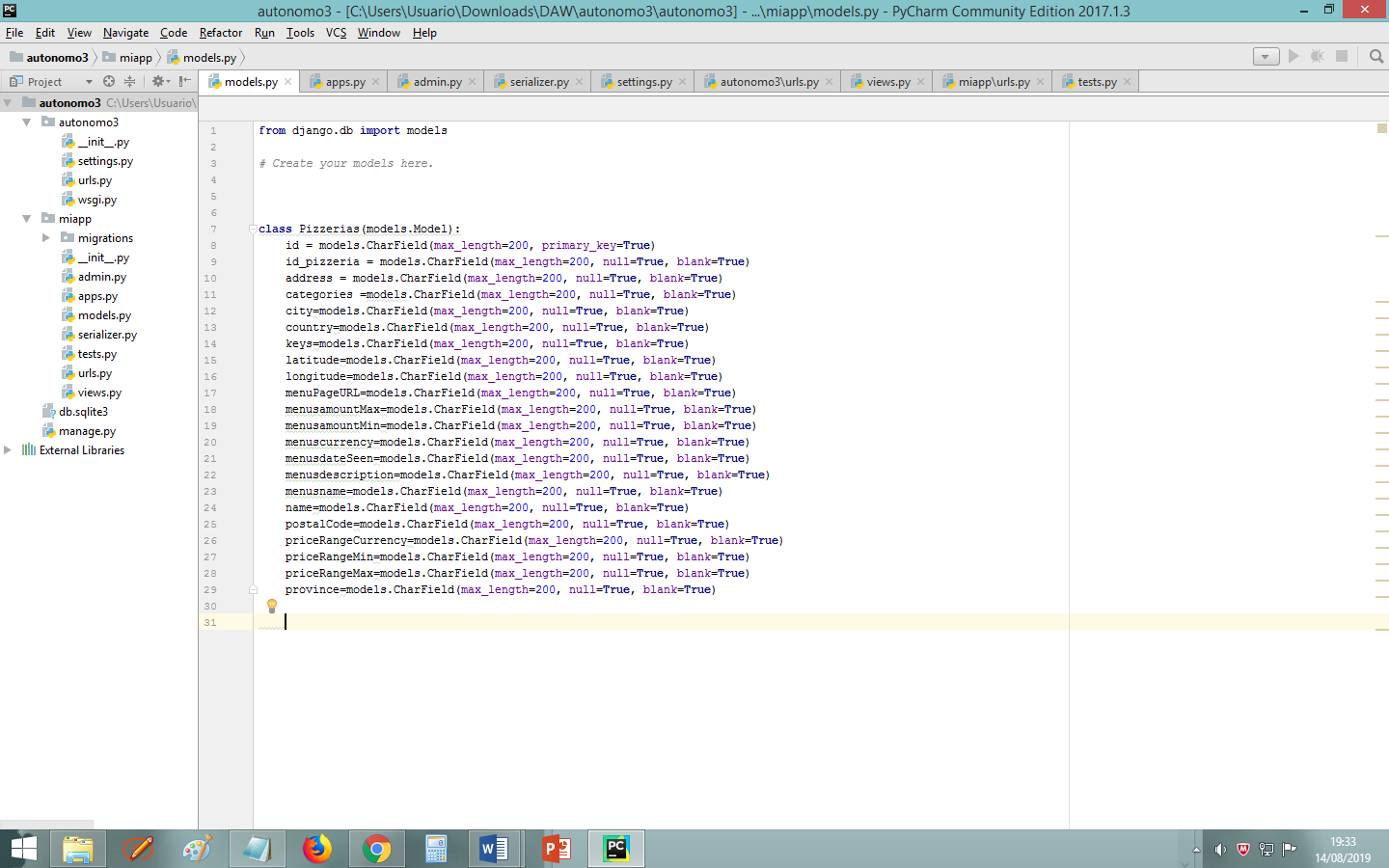
***Province***: nombre del barrio o vecindario en que se ubica el restaurante.

La información ha sido descargada como un archivo csv para posteriormente cargarla en una base de base de datos no relacional local de MongoDB.

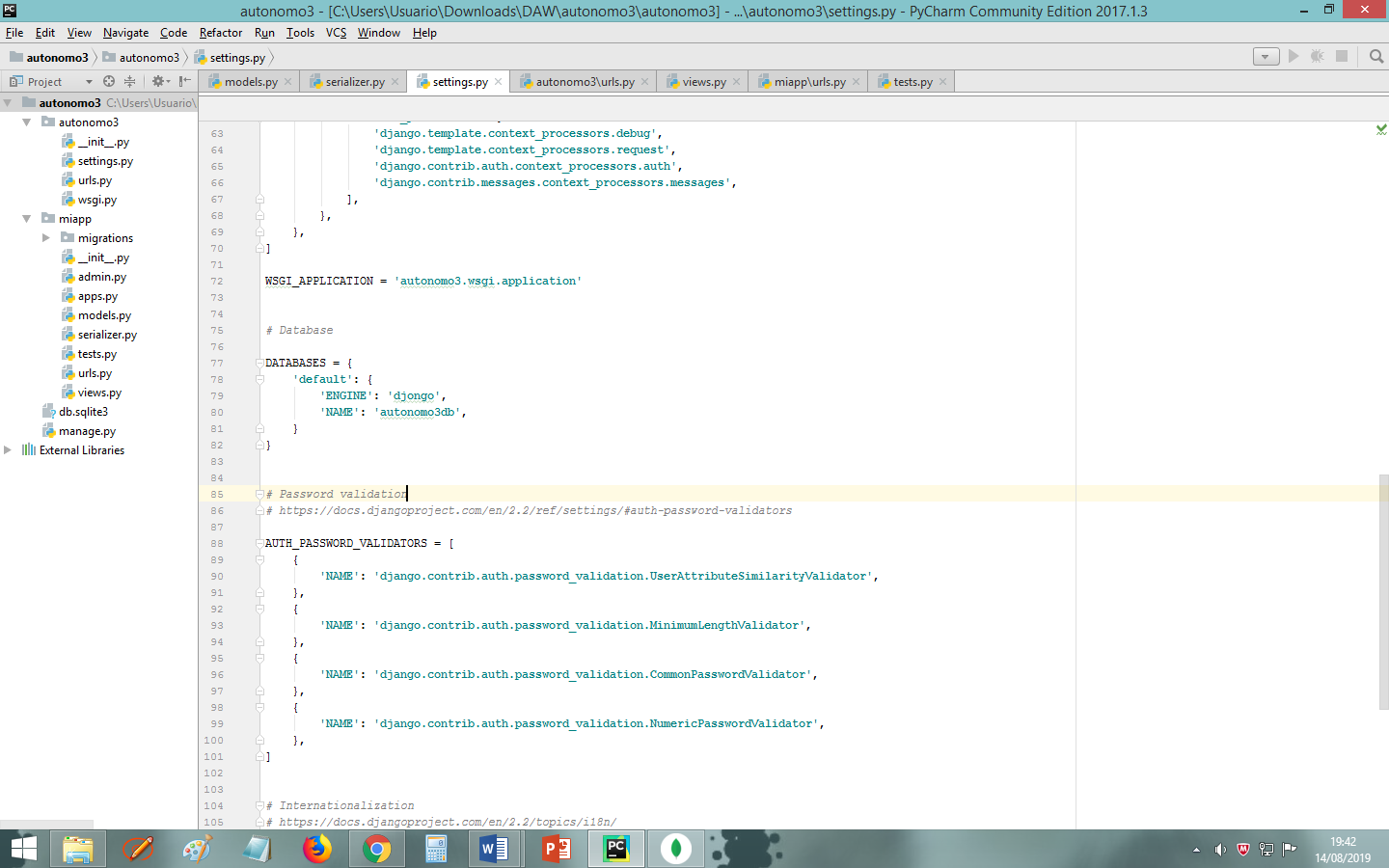


1. **Base de datos no relacional**

***Describa la base de datos de no relacional, el tipo y los parámetros de conexión. Agregue las capturas de pantalla que sean necesarias.***

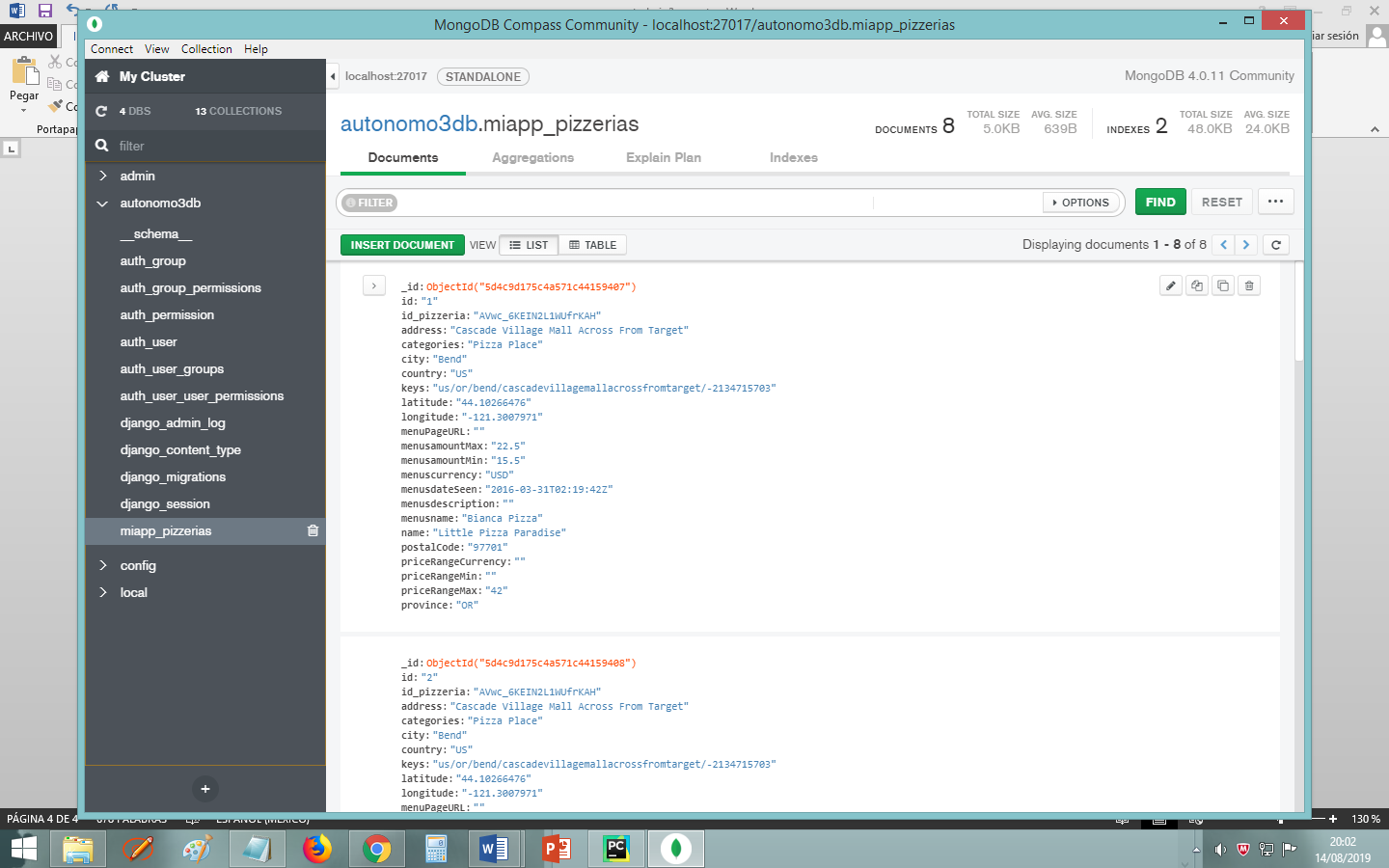
Se ha utilizado el sistema de base de datos NOSQL MongoDB. En models.py se procede a crear el modelo de Pizzerías con sus respectivos campos ya vistos dentro de la página web y el csv. Aquí cada documento u objeto insertado será un menú de Pizzería dentro de la colección “Pizzerías”.

Esta colección “Pizzerias” a su vez se encuentra dentro de la base de datos con nombre “autonomo3db” cuya conexión con Django debe configurarse en DATABASES de settings.py:



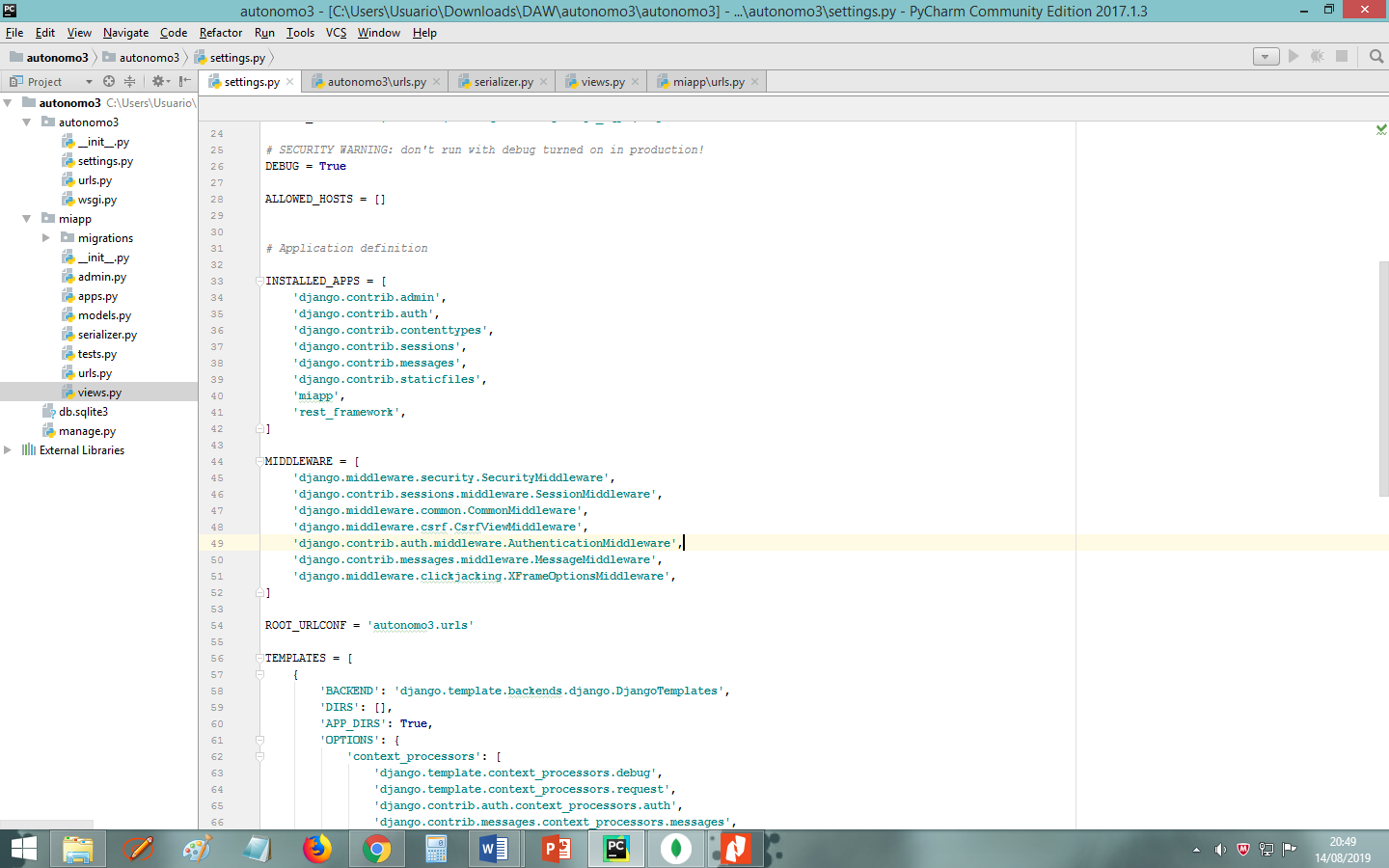
Se ha hecho uso del conector de Django con MongoDB “djongo”[[3]](#footnote-3). Este conector permite unificar bases de datos NOSQL de Mongo con cualquier proyecto de Django. Esto también hace posible que se puedan modificar los documentos de las colecciones de Mongo a través de la interfaz del API REST o la interfaz de Administrador, es decir que no existe algún cambio significativo en el manejo del ORM de Django. El puerto que utiliza MongoDB por defecto es el 27017. Algo muy importante de mencionar es que esta base de datos es de tipo clave-valor.

Una vez que se ha establecido la configuración para la conexión, se procede a realizar las migraciones makemigrations, migrate y se carga el archivo csv mediante Compass lo que permite observar el estado actual de la base de datos:

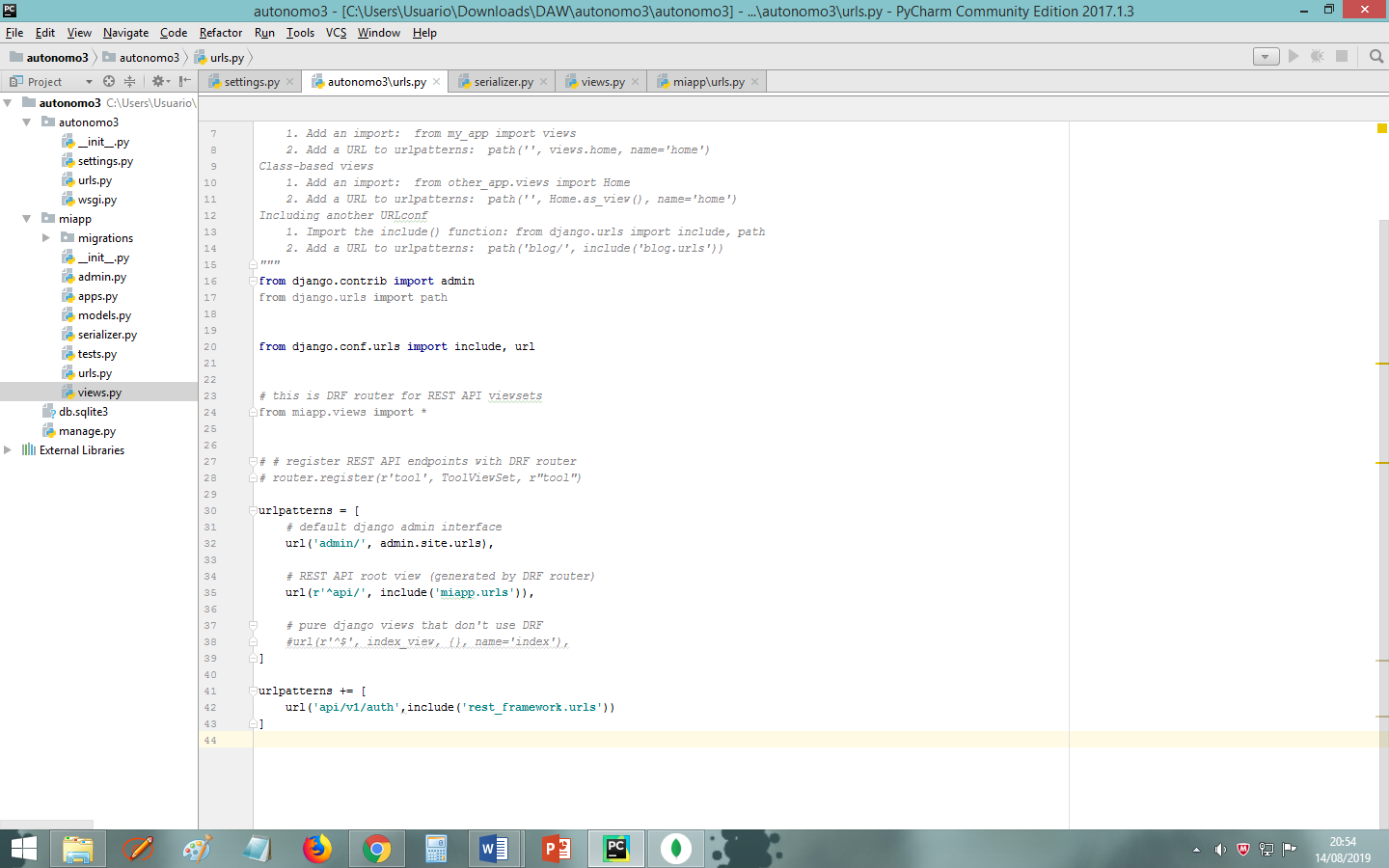


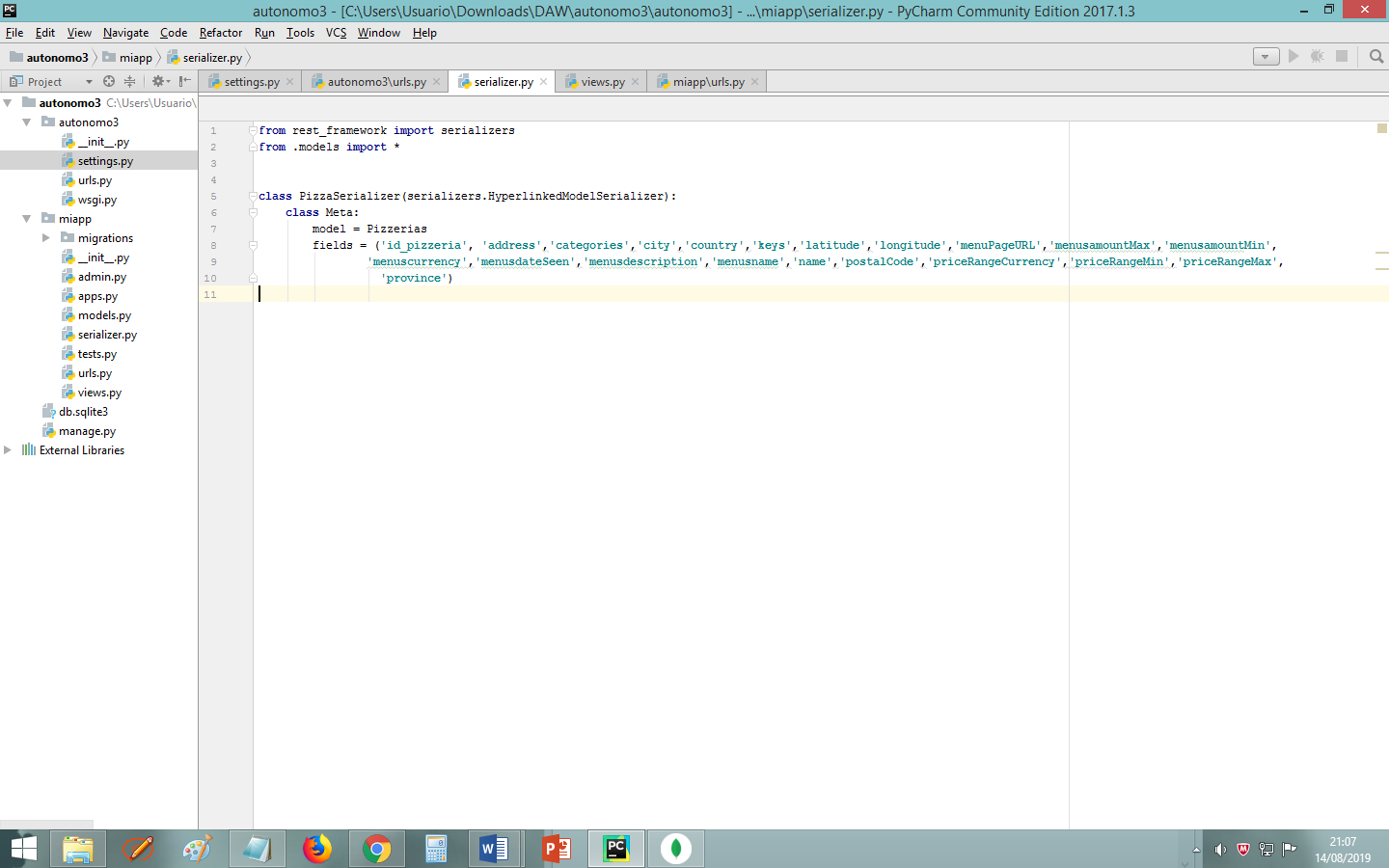
1. **RESTful API**

***Describa los paquetes o los módulos que utilizó para generar el RESTful API. Agregue las capturas de pantalla que sean necesarias.***

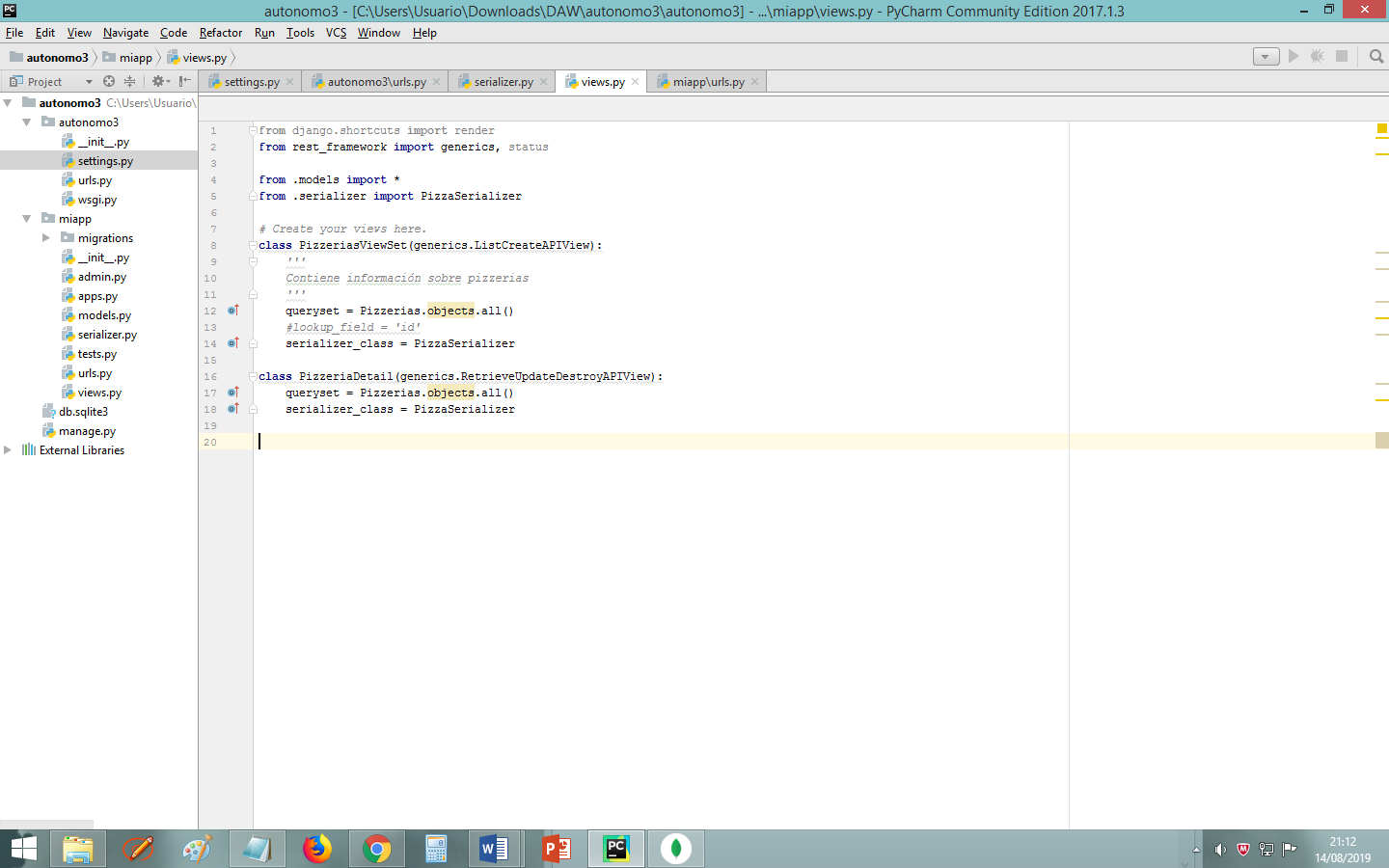
Para generar el RESTful API se hace uso del toolkit que ofrece Django: Django Rest Framewok[[4]](#footnote-4) y se siguen las instrucciones de su sitio oficial <https://www.django-rest-framework.org/> . Posterior a la instalación se agrega el rest framework a las aplicaciones instaladas en settings.py del proyecto para que sus paquetes y módulos puedan ser utilizadas dentro del código.

Para la creación del API en una url determinada es necesario asignar las rutas al urlpatterns que se encuentra en urls.py de la raíz del proyecto:



Posteriormente se crea un serializador en serializer.py dentro del directorio de la aplicación. Este serializador es un componente del Rest Framework que debe ser creado para facilitar la conversión de estructuras en archivos JSON. En este caso la clase PizzaSerializer utiliza el paquete "serializers" de "rest\_framework" para importar un modelo base y a partir de este se construye la estructura de la respuesta al request que son los campos del Modelo Pizzerías:

Adicionalmente es necesario el diseño de controladores que actúen de manera correcta dependiendo del tipo de request. Para esto se crean vistas que respondan a cada caso dentro de views.py:

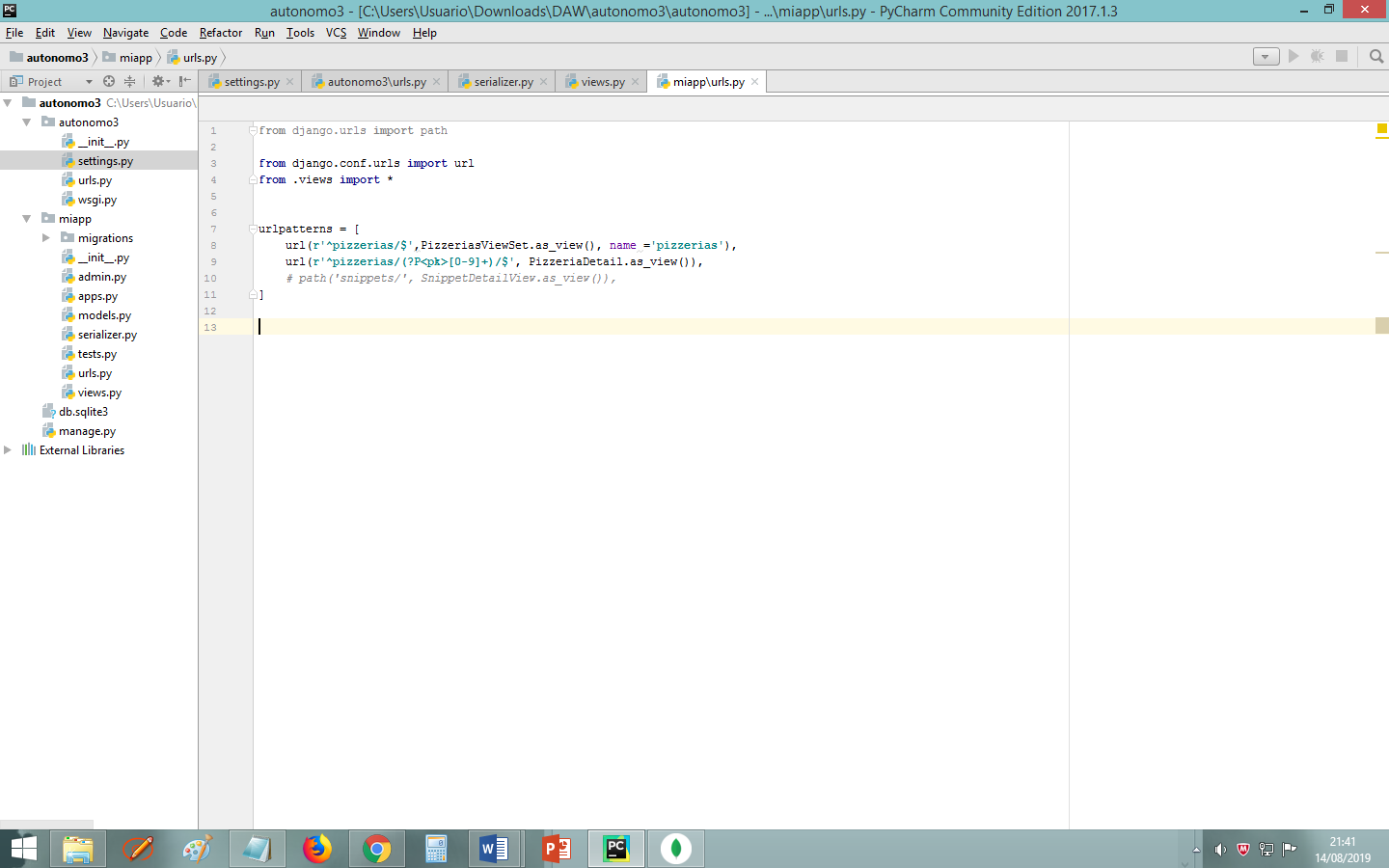


Estas vistas especiales utilizan clases genericas del paquete "generics" del rest framework de Django.

La clase PizzeriasViewSet llama a generics.ListCreateAPIView que permite la ejecución de los métodos GET y POST mediante una interfaz que por defecto proporciona el APIRest ubicada en una url que debe ser asignada posteriormente en urls.py del directorio de la aplicación.

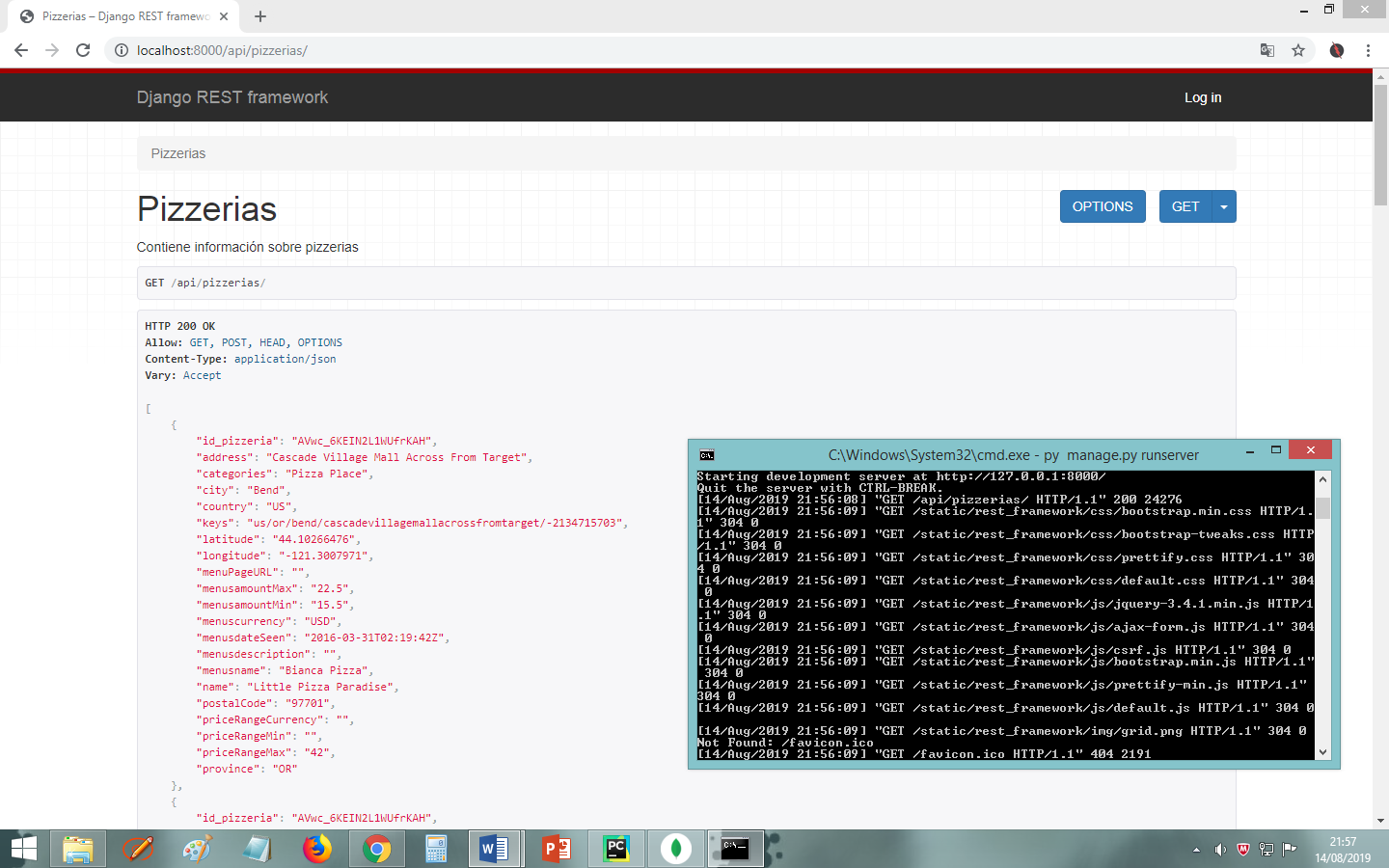
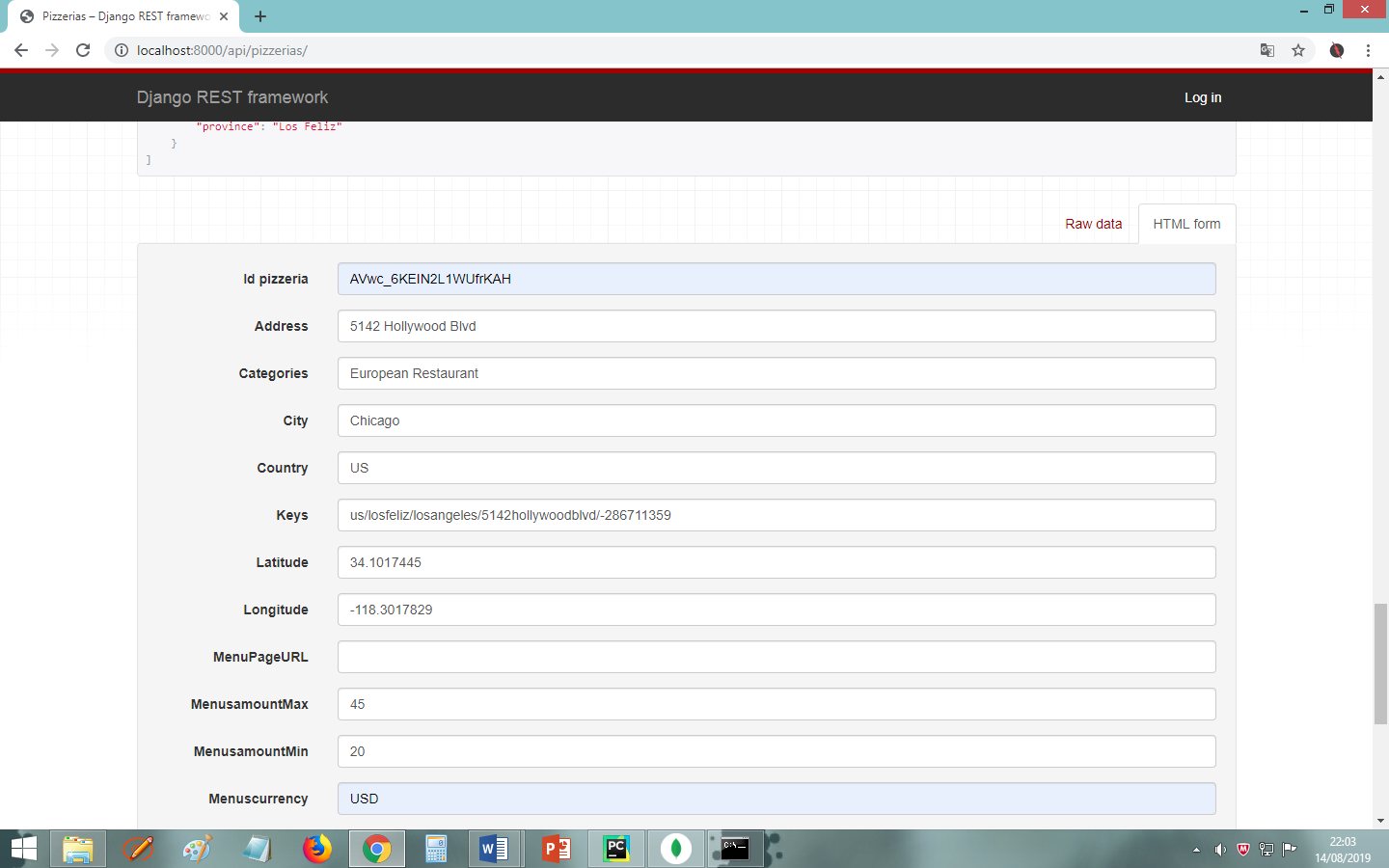
PizzeriaDetail utiliza la clase generics.RetrieveUpdateDestroyAPIView para permitir todas las operaciones CRUD en una pizzería determinada. También es necesario importar los modelos a los cuales se desean aplicar las operaciones con su respectivo serializador, ambos son argumentos que permiten que internamente el APIRest se comunique con la base de datos y realice las operaciones necesarias.

Finalmente se debe incluir las urls del APIRest en las urls de la aplicación, para esto se agrega a urlpatterns las rutas para ingresar a las 2 vistas previamente creadas en views.py. La asignación a PizzeriasViewSet se define simplemente como “pizzerías” que al unirla junto a la url de la api que fue configurada en urls.py de la raíz del proyecto queda finalmente como “<http://localhost:8000/api/pizzerias>”.

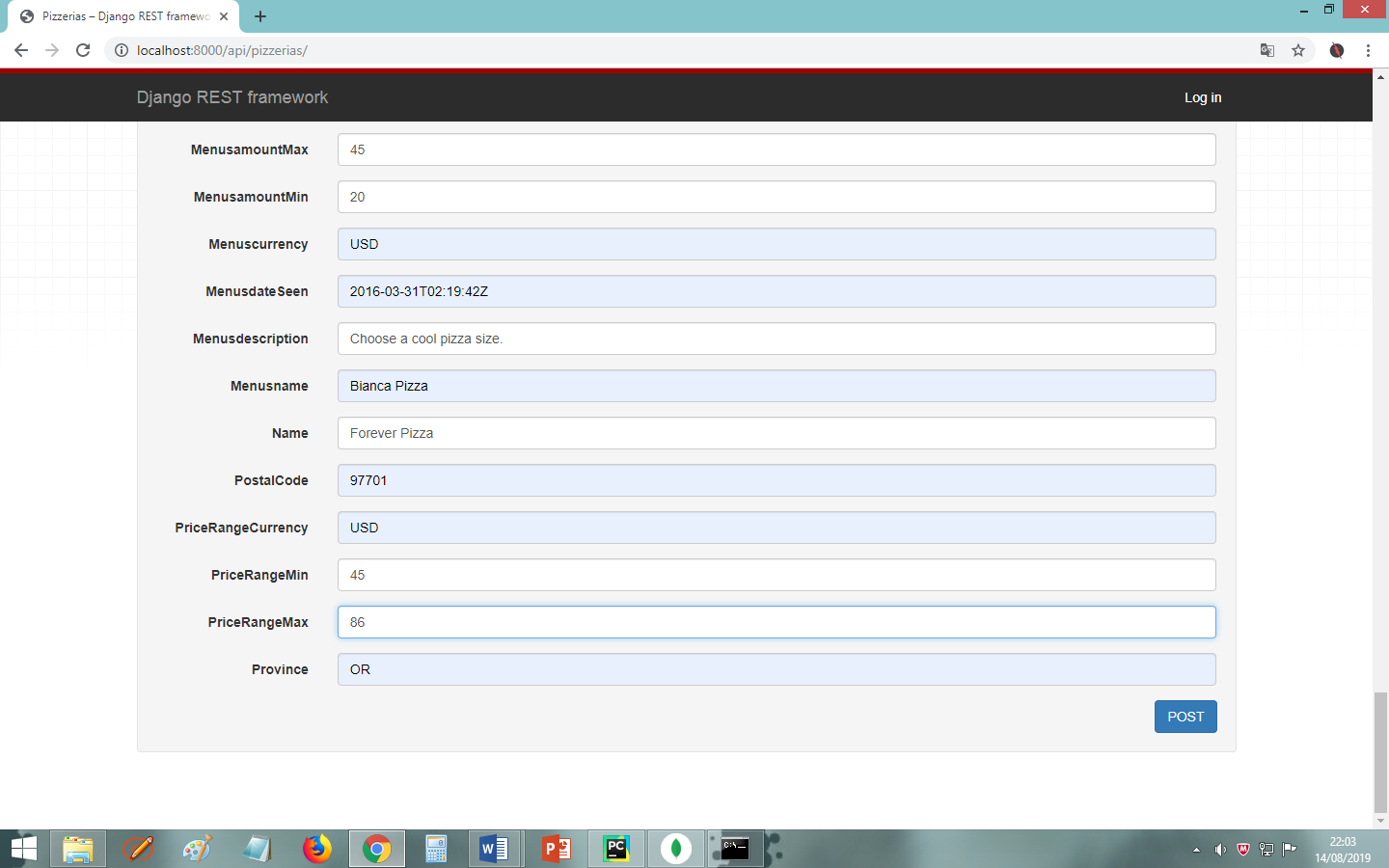
Este sitio nos devuelve una lista de pizzerías obtenidas mediante GET mientras que se permite la creación mediante POST de un nuevo documento dentro de la colección "Pizzerias" en la Base de datos autonomo3db.

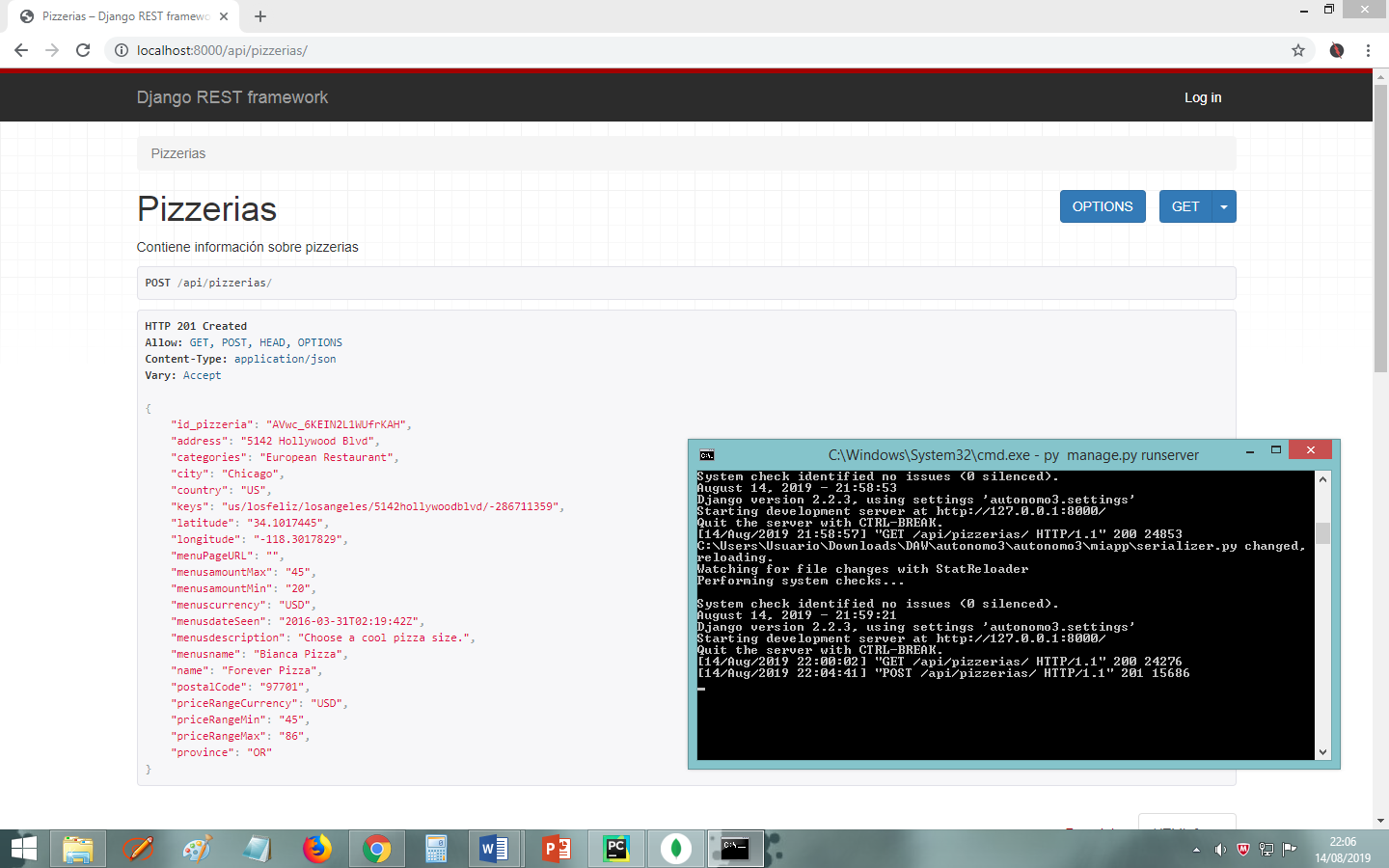
La asignación a PizzeriaDetail se define como “pizzerías/pk” donde pk es la Primary Key de la pizzería que en este caso es un valor numérico a partir de 1, es decir si se desea actualizar o eliminar el objeto con id=1 se debe entrar a: “<http://localhost:8000/api/pizzerias/1>”. Aquí se utilizan las operaciones GET para obtener el elemento, PATCH para modificarlo y DELETE para eliminarlo

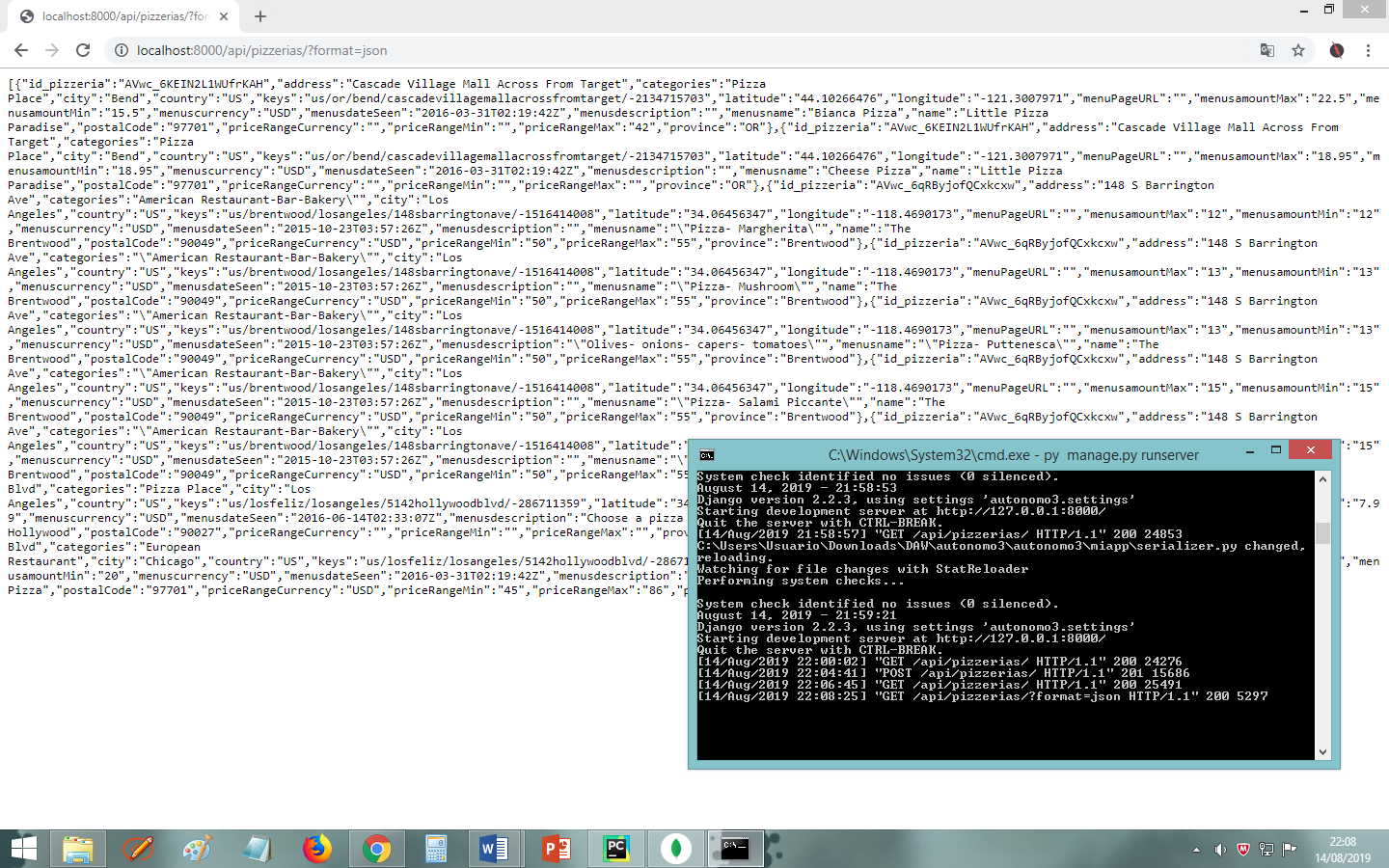
1. **Sitio web de consulta.**

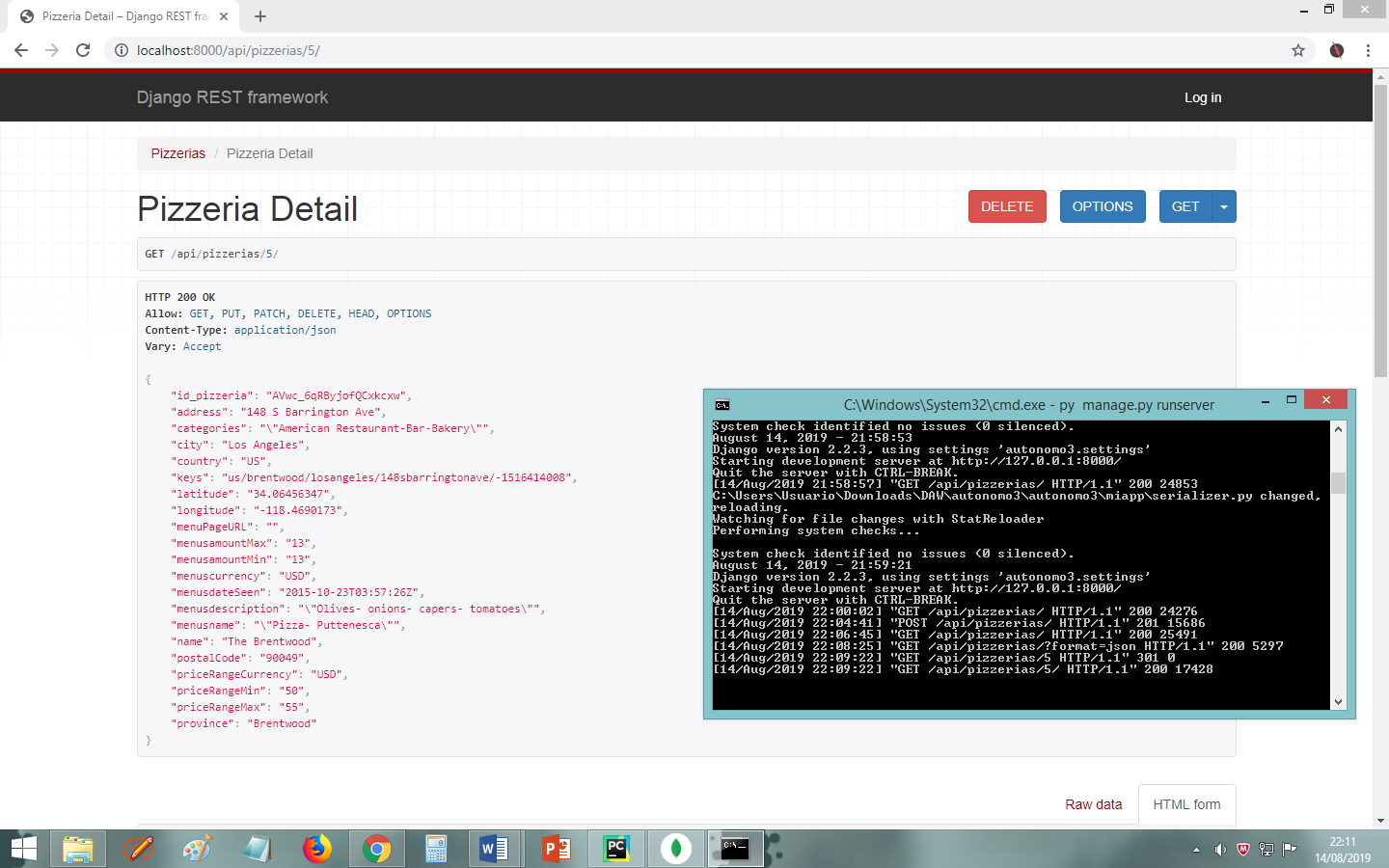
***Agregue las capturas de pantalla del sitio web.***

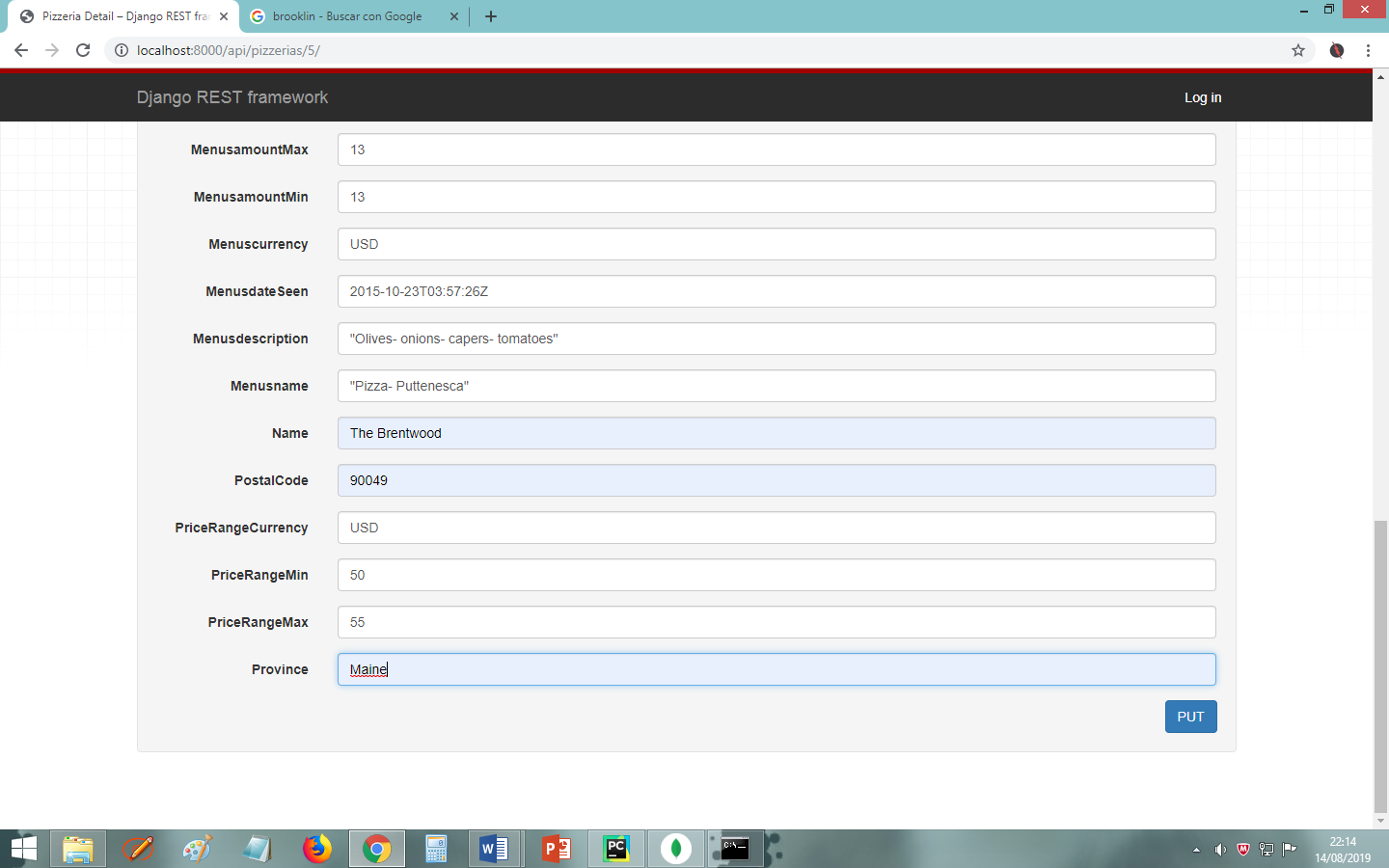
**GET**

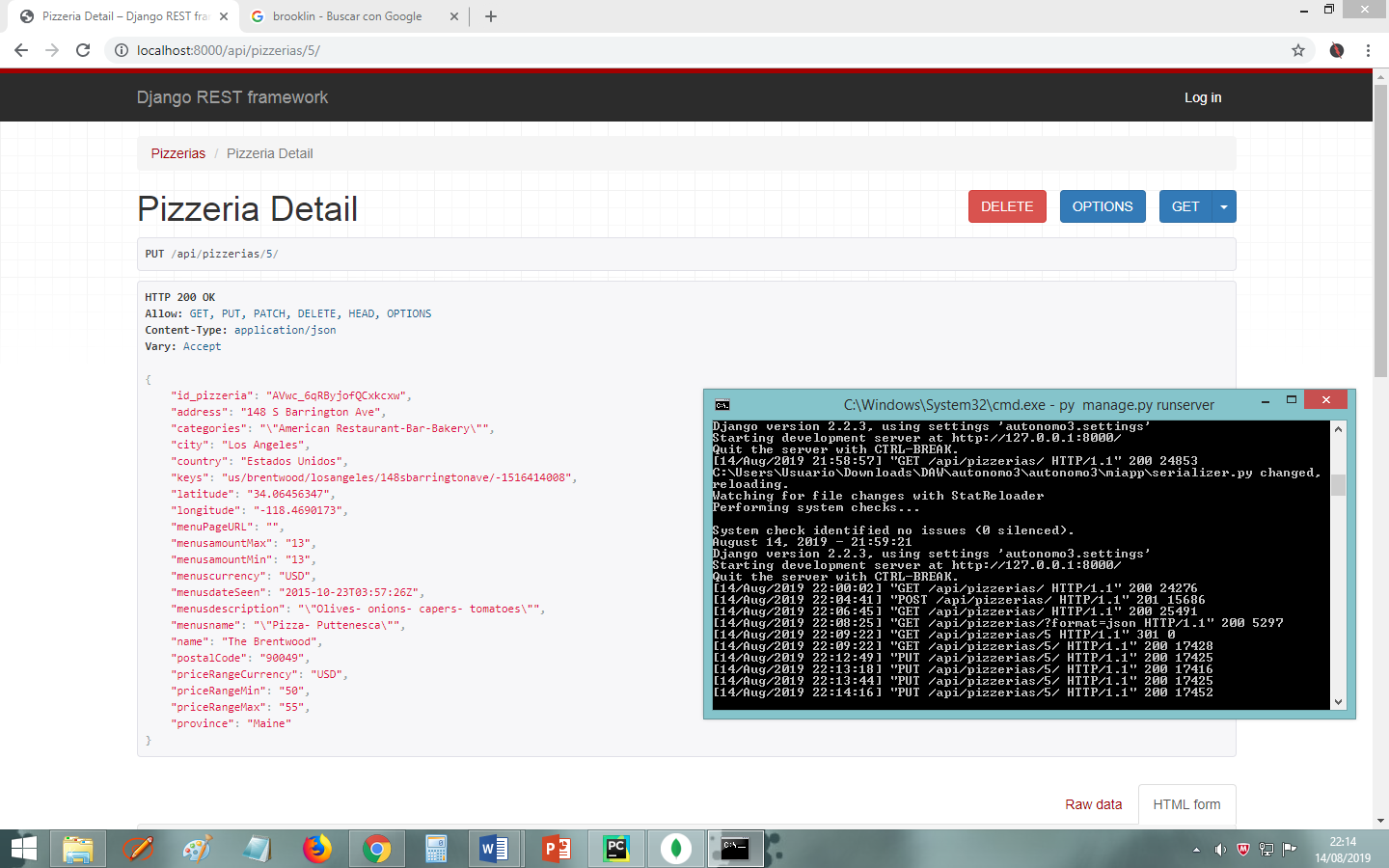
**POST**

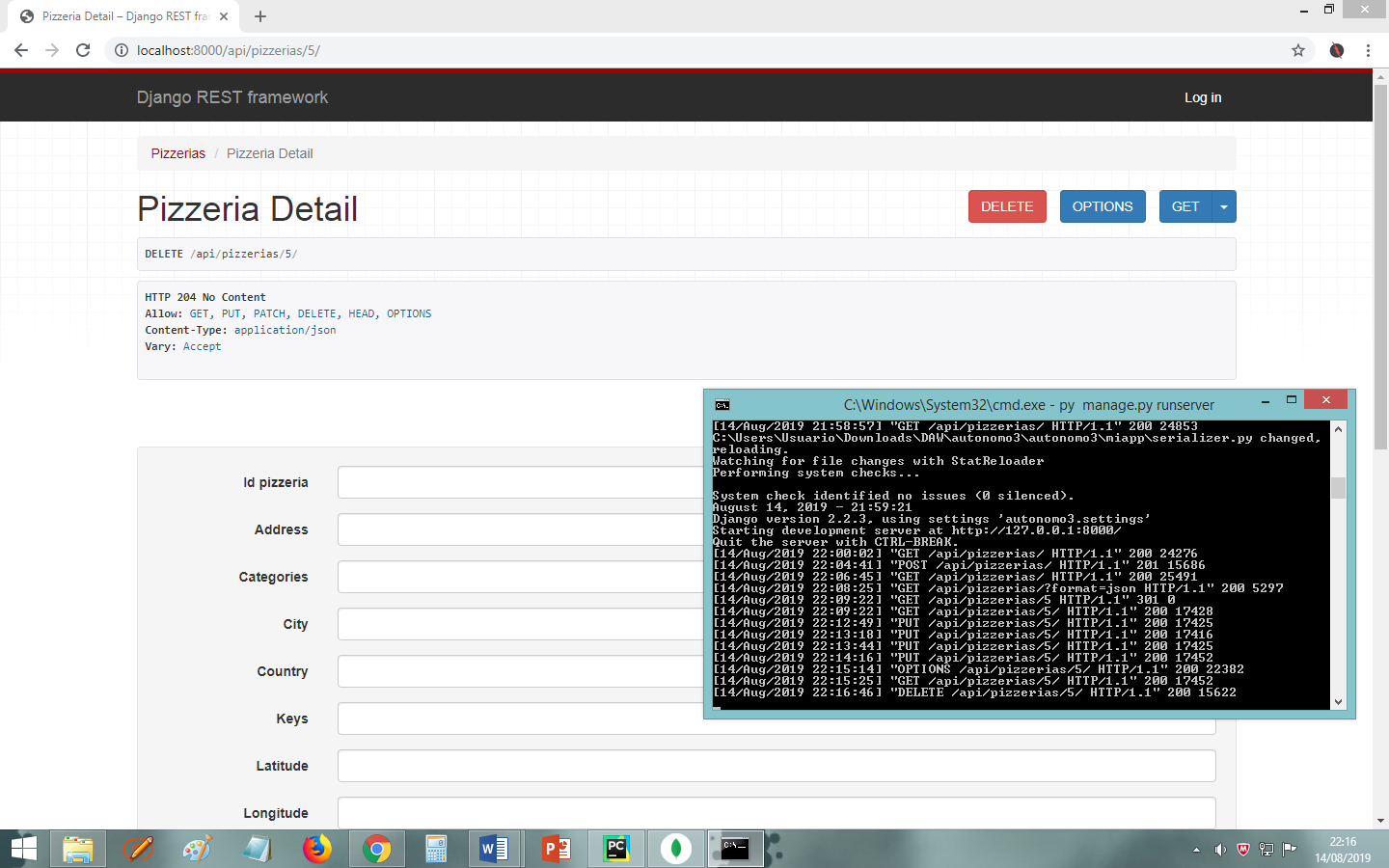
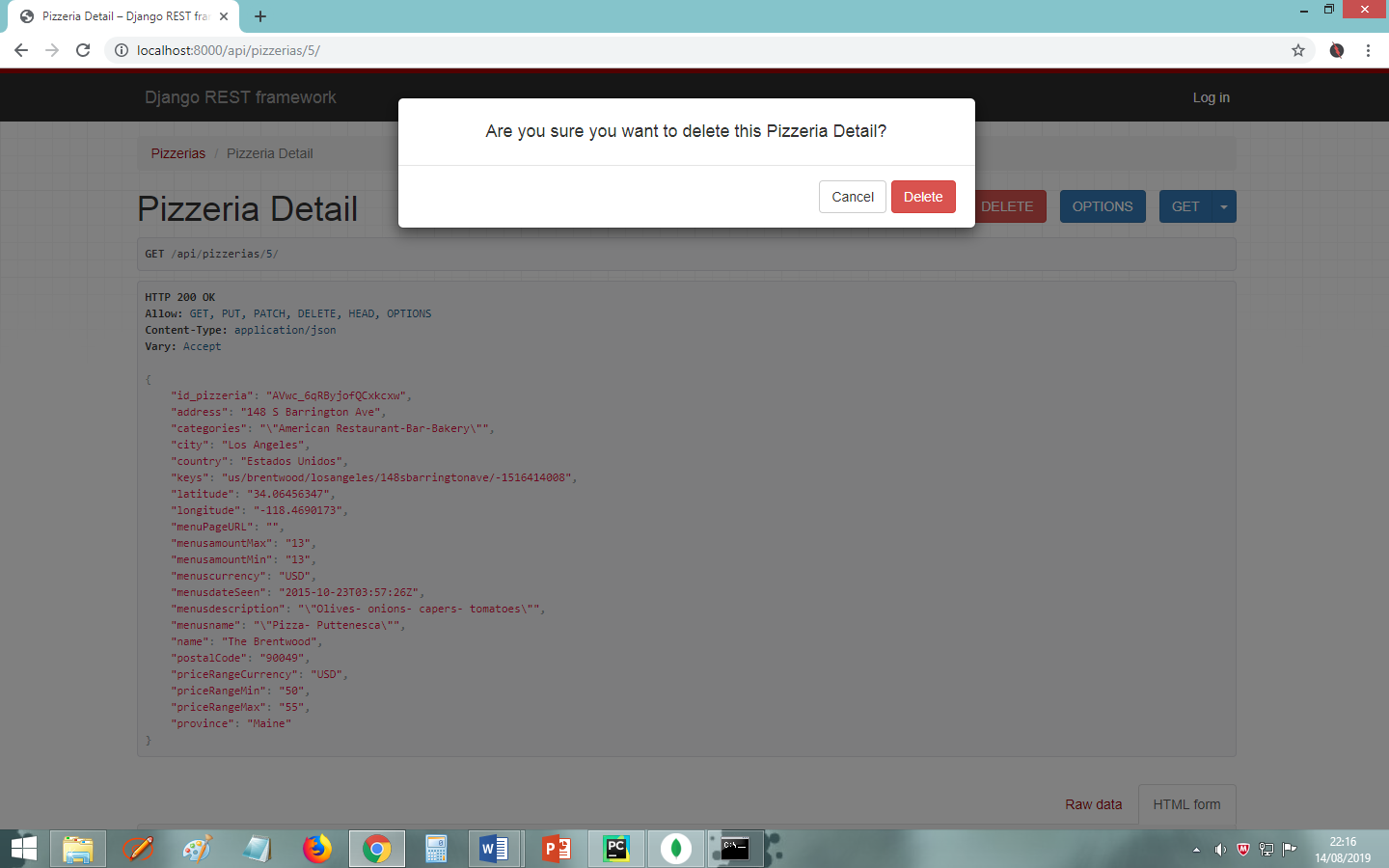
****

**DATOS EN JSON**

**GET ELEMENTO CON ID=5**

**UPDATE**



**DELETE**

# **Referencias**

Datafiniti. (9 de Agosto de 2019). *Kaggle*. Obtenido de Kaggle: https://www.kaggle.com/datafiniti/pizza-restaurants-and-the-pizza-they-sell

1. Metadata. <https://www.powerdata.es/metadatos> [↑](#footnote-ref-1)
2. Dataset. <https://www.kaggle.com/datafiniti/pizza-restaurants-and-the-pizza-they-sell>. [↑](#footnote-ref-2)
3. Djongo. <https://github.com/nesdis/djongo> [↑](#footnote-ref-3)
4. Django Rest Framework. <https://www.django-rest-framework.org/> [↑](#footnote-ref-4)