UNIVERSIDADE DA CORUÑA

DINEOLOGY

DISCOVER CULINARY EXCELLENCE

https://github.com/anoya97/RIWS-Crawling

Integrantes:

Armando Martínez Noya

Brais González Piñeiro

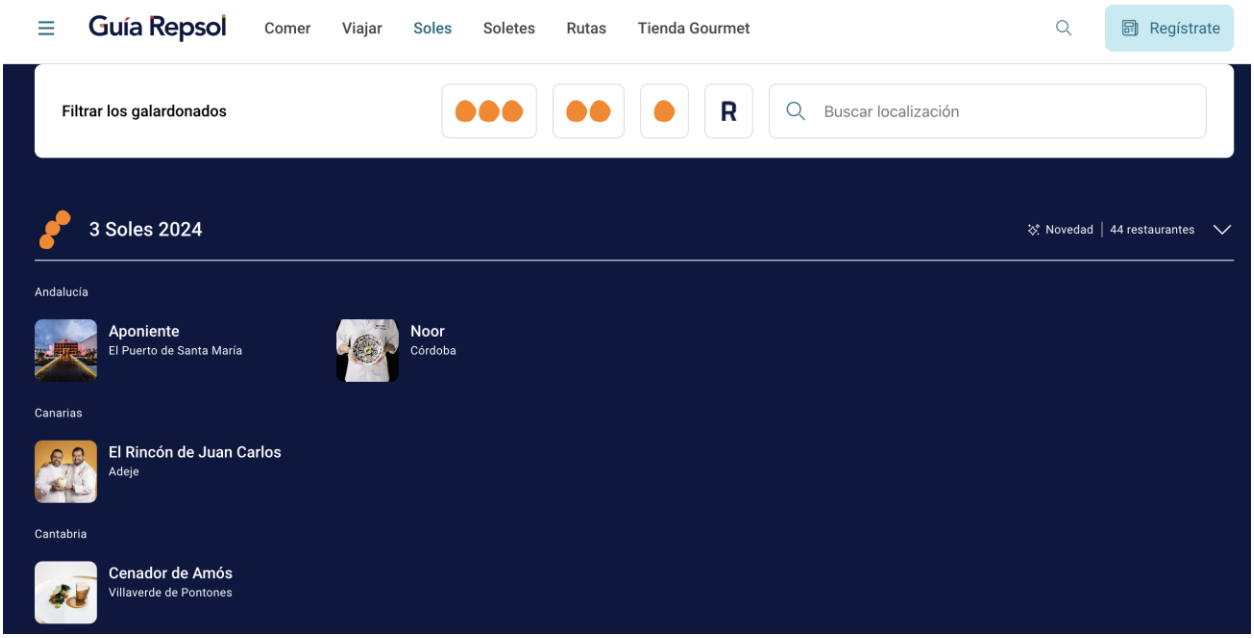
Raúl Fernández del Blanco

Búsqueda del dominio y crawling web

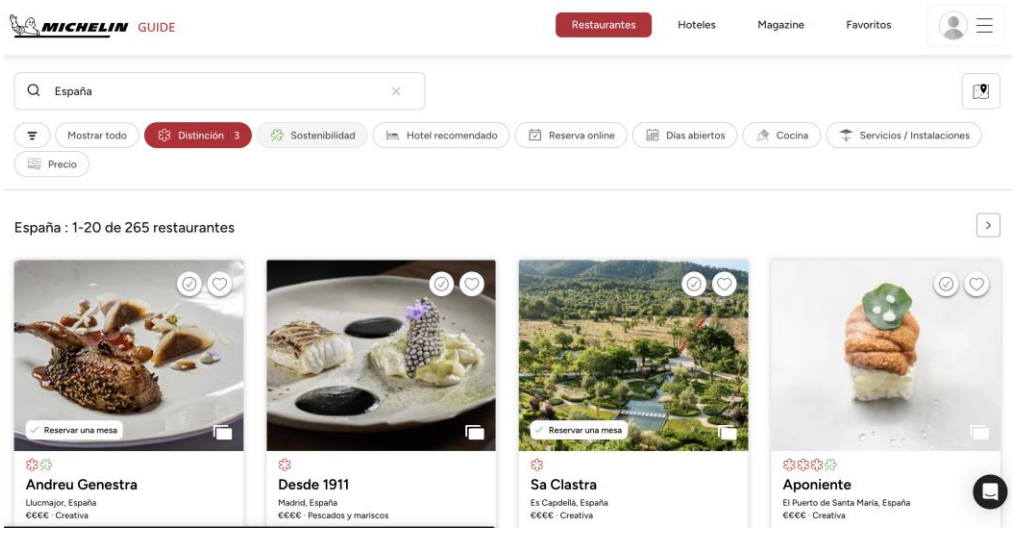
En el inicio del proyecto, se llevó a cabo una búsqueda exhaustiva del dominio, con el objetivo de recopilar información sobre restaurantes que han sido galardonados con las prestigiosas estrellas Michelin y los soles Repsol. Estos reconocimientos son importantes indicadores de calidad en la gastronomía, y su búsqueda requiere una metodología cuidadosa para asegurar la precisión y relevancia de los datos obtenidos.

Para llevar a cabo esta recolección de datos, se seleccionaron dos fuentes principales:

- **Guía Repsol**: Esta guía es reconocida por su extensa recopilación de información sobre los mejores restaurantes de España que han sido galardonados con soles Repsol. En su sitio web, (https://guiarepsol.com/es/soles-repsol/ediciones-de-soles-guia-repsol/) se pueden encontrar listados de restaurantes organizados por diferentes ediciones, así como descripciones detalladas, tipos de cocina, y valoraciones que son cruciales para la aplicación.



- **Guía Michelin**: La Guía Michelin, (https://guide.michelin.com/es/es) es una de las más reconocidas a nivel mundial en el ámbito de la crítica gastronómica. Proporciona información sobre restaurantes que han recibido estrellas Michelin, así como recomendaciones y valoraciones. La riqueza de datos en esta guía, que incluye detalles sobre los chefs, menús, y características distintivas de cada establecimiento, la convierte en una fuente invaluable para el proyecto.



El proceso de *crawling* se llevó a cabo utilizando Scrapy, un marco de trabajo de Python altamente eficaz y especializado en la recolección de datos web. Scrapy permite la creación de spiders, que son scripts diseñados para navegar por las páginas web y extraer información de manera automatizada. La elección de Scrapy se debió a su capacidad para manejar múltiples tareas simultáneamente, su eficiencia en la gestión de solicitudes y su flexibilidad para adaptarse a diferentes estructuras HTML.

Durante la fase de desarrollo de los spiders, se prestó especial atención a la estructura de las páginas web de las fuentes seleccionadas. Se analizaron los elementos HTML y se identificaron las etiquetas que contenían la información relevante sobre los restaurantes, como el nombre, la dirección, el tipo de cocina, el rango de precios, y la calificación de estrellas o soles. Esta etapa fue crucial, ya que garantizó que los spiders pudieran extraer datos precisos y completos.

Desarrollo del proyecto

El **backend** fue implementado en Python utilizando el framework Scrapy, un potente marco de trabajo diseñado específicamente para la recolección de datos web. La elección de Python y Scrapy se debió a su robustez, flexibilidad y la amplia comunidad de desarrolladores, lo que facilita la resolución de problemas y la implementación de nuevas funcionalidades.

En esta fase, se diseñaron varios spiders que se encargan de navegar por las estructuras HTML de las páginas seleccionadas y recolectar información relevante. Cada spider fue configurado para cumplir con objetivos específicos, como extraer los siguientes datos de cada restaurante:

- **Nombre del restaurante**: Identificación clave para los usuarios.
- **Ubicación**: Dirección física del restaurante, esencial para los usuarios que desean visitar.

- **Tipo de comida**: Categoría gastronómica (por ejemplo, mediterránea, asiática, etc.), que permite a los usuarios encontrar restaurantes según sus preferencias culinarias.
- **Precio**: Indicador del rango de precios del menú, lo que ayuda a los usuarios a ajustar sus elecciones según su presupuesto.
- **Estrellas Michelin**: Información sobre la clasificación del restaurante, que es un importante distintivo de calidad en la gastronomía.
- **Soles Repsol**: Clasificación adicional que también denota la excelencia del restaurante.
- **Foto del restaurante**: Imagen representativa del restaurante, que permite a los usuarios visualizar el ambiente del lugar.
- **Descripción**: Texto que proporciona detalles sobre el restaurante, su ambiente y su filosofía gastronómica.
- **Número de contacto**: Teléfono o número de contacto para reservas o consultas, esencial para que los usuarios puedan comunicarse fácilmente.
- **Página web**: Enlace a la página oficial del restaurante, para más información o reservas en línea.
- **Horario de apertura**: Información sobre los días y horas de apertura, facilitando a los usuarios planificar su visita.
- **Descripción del menú corto**: Resumen de algunos de los platos destacados, permitiendo a los usuarios conocer las especialidades.
- **Opciones de menú**: Detalle de menús especiales (por ejemplo, degustación, halal, opciones para celíacos) con precios, que ofrece alternativas para distintas necesidades alimentarias.
- **Servicios del restaurante**: Lista de servicios como accesibilidad, aparcamiento, vistas, opciones para grupos, etc., que brindan una experiencia completa a los usuarios.
- **Propietario(s)**: Nombre(s) del chef o propietario, destacando a personas clave detrás del éxito del restaurante.
- **Usuario de Instagram**: Nombre de usuario en Instagram, donde los usuarios pueden seguir el restaurante para ver fotos y actualizaciones.

  Extracto del Spider de Michelin:

```python
def parse_detail_page(self, response):

    # Obtenemos el item completo
    item = response.meta['item']

    item['restaurant_photo_url'] = response.css('div.masthead__gallery-image::attr(data-bg)').get()

    star_icons = response.css('div.data-sheet__classification-item--content img.michelin-award').getall()
    item['star_number'] = len(star_icons)

    item['direction'] = response.css('div.data-sheet__block--text::text').get()

    item["description"] = response.css('div.data-sheet__description::text').get().strip()

    # item['contact_number'] = response.css('div.d-flex span::text').get().strip()
    item['contact_number'] = re.sub(r'^\+34|\s+', '', response.css('div.d-flex span::text').get().strip())
    #item['contact_number'] = re.sub(r'\D', '', item['contact_number']).lstrip('+')
```

Extracto del Spider de Repsol:

```python
def parse_detail_page(self, response):
    item = DineologyItemRepsol()

    item['name'] = response.css('div.hero__ficha__main h1::text').get().strip()

    number_soles = response.css('div.badge__data span.badge__category::text').get()

    item['soles_number'] = int(number_soles.replace("soles","").replace("sol","").replace("Recomendado","0"))

    item['description'] = ''.join(response.css('div.description-block p *::text').getall()).strip()

    item['short_menu_description'] = response.css('section.list-info-component p::text').get()
```

El **frontend** fue desarrollado utilizando **React** y **JavaScript**, lo que permitió crear una interfaz de usuario interactiva y responsiva. React, como biblioteca de JavaScript, fue elegido por su capacidad para construir componentes reutilizables, lo que hace que la aplicación sea más modular y fácil de mantener.

A la hora de utilizar la barra de búsqueda, se obtiene el input introducido por el usuario y se utiliza para realizar la busqueda en los índices de Elasticsearch. En esta búsqueda se realiza una query que compara este input con los siguientes parámetros:

- Nombre.
- Tipo de comida.
- Dirección.

```javascript
const response = await axios.post('http://localhost:9200/restaurants/_search', {
  size: 1000,
  query: {
    bool: {
      should: [
        // Coincidencia exacta sin considerar mayúsculas
        { term: { "name.keyword": searchQuery.toLowerCase() } },
        { match: { "name": searchQuery.toLowerCase() } },
        { wildcard: { "name": `*${searchQuery.toLowerCase()}*` } },
        { wildcard: { "meal_type": `*${searchQuery.toLowerCase()}*` } },
        { wildcard: { "direction": `*${searchQuery.toLowerCase()}*` } }
      ],
      minimum_should_match: 1
    }
  }
});
```

En el diseño del frontend se prestó especial atención a la experiencia del usuario. La interfaz incluye:

- Barra de Búsqueda.
- Filtros de Búsqueda.
- Cartas clicables con los resultados de la búsqueda.
- Página de detalles de los restaurantes:
    - Componente de opciones de menú.
    - Carta de información de restaurante.
    - Componente con los servicios que ofrecen.
    - Horario de apertura.
    - Mapa interactivo con su dirección.

Adaptación de los datos y volcado en Elastic

En primer lugar, al realizar el crawling de cada una de las webs, llevamos a cabo una cuidadosa adaptación y limpieza de los datos. Esto incluyó la corrección de errores tipográficos y la estandarización de algunos formatos, como el de los números de teléfono, asegurándonos de que todos los datos recogidos tuvieran un formato uniforme. Asimismo, completamos y homogenizamos las URLs de los restaurantes para mantener una estructura estable y coherente a lo largo de todo el conjunto de datos.

Para la creación del índice en Elasticsearch, seguimos el procedimiento habitual. Sin embargo, el verdadero desafío surgió en el proceso de inserción de los datos, ya que las dos webs rastreadas podían contener restaurantes duplicados. Esto hizo necesario implementar una estrategia que permitiera identificar duplicados y fusionar (merge) la información, manteniendo los datos relevantes de cada entrada.

```
index_name = "restaurants"
if not es.indices.exists(index=index_name):
    try:
        es.indices.create(index=index_name)
        print(f"Índice '{index_name}' creado.")
    except exceptions.ElasticsearchException as e:
        print(f"Error al crear el índice '{index_name}': {e}")
```

Para lograr esta deduplicación, primero realizamos una query compleja al intentar insertar los datos de la segunda fuente (Repsol) en Elasticsearch. Partiendo de que los restaurantes de la primera fuente (Michelin) ya estaban indexados, la query comprueba, en el momento de insertar los nuevos registros, si alguno de los restaurantes indexados (con estrellas y sin soles) coincide (aunque sea

parcialmente) en el nombre con algún restaurante de la segunda fuente. De esta manera, identificamos posibles duplicados que requieren confirmación.

```
def check_similar_restaurant_with_stars(restaurant_name):
    query = {
        "query": {
            "bool": {
                "must": [
                    {
                        "match": {
                            "name": {
                                "query": restaurant_name,
                                "fuzziness": "AUTO"  # Fuzziness automático para mayor flexibilidad
                            }
                        }
                    },
                    {
                        "exists": {
                            "field": "star_number"
                        }
                    },
                    {
                        "bool": {
                            "must_not": [
                                {
                                    "exists": {
                                        "field": "soles_number"
                                    }
                                }
                            ]
                        }
                    }
                ]
            }
        }
    }
```

La confirmación de duplicados se lleva a cabo mediante las siguientes verificaciones:

- **Verificación de contacto y web:** en primer lugar, comparamos el número de contacto y la URL del restaurante en ambas fuentes. Si al menos uno coincide, se considera que se trata del mismo restaurante y se procede a fusionar la información en el índice, combinando los datos relevantes de ambas webs.

```
def check_url_and_contact_equal(doc1, doc2):
    url1 = doc1.get("web_url")
    url2 = doc2.get("web_url")
    contact1 = doc1.get("contact_number")
    contact2 = doc2.get("contact_number")
    return url1 == url2 or contact1 == contact2
```

- **Verificación de dirección mediante distancia de Levenshtein:** en los casos en que no haya coincidencia en el número de contacto o la URL, comparamos las direcciones de ambos registros. Para ello, calculamos la distancia de Levenshtein entre las direcciones. Si la distancia es suficientemente baja (es decir, las direcciones son relativamente similares según un umbral configurado), consideramos que es probable que se trate del mismo restaurante y también fusionamos los documentos en el índice.

```
# Comparar usando Levenshtein
name_distance = levenshtein_distance(doc.get("name", ""), existing_doc.get("name", ""))
direction_distance = levenshtein_distance(doc.get("direction", ""), existing_doc.get("direction
if name_distance <= LEVENSHTEIN_THRESHOLD_NAME or direction_distance <= LEVENSHTEIN_THRESHOLD_D
    merged_doc = merge_documents(existing_doc, doc)
    es.index(index=index_name, id=existing_doc_id, body=merged_doc)
    merged = True
    break
```

Gracias a este proceso de verificación, logramos un índice en Elasticsearch que evita duplicados y preserva los datos más relevantes de cada fuente. Esta técnica nos permite disponer de una base de datos consolidada y precisa, mejorando la calidad y la integridad de la información disponible para el usuario final.

Funcionalidades implementadas

El sistema desarrollado cuenta con varias funcionalidades clave que mejoran la experiencia del usuario al buscar restaurantes:

- Barra de Búsqueda: Permite realizar búsquedas de restaurantes mediante palabras clave.
- Filtros de Búsqueda:

    - Selección de soles o estrellas (o ambos).
    - Selección de rango de precios.
    - Filtrado por tipo de comida.
    - Selección de número de estrellas o soles.

Tecnologías utilizadas

El proyecto empleó varias tecnologías que contribuyeron a su desarrollo:

- **Scrapy**: Framework en Python para el scraping web, que facilita la creación de spiders y la recolección de datos.
- **Python**: Lenguaje de programación utilizado para el backend del proyecto.
- **React**: Biblioteca de JavaScript utilizada para el desarrollo del frontend, que permite crear interfaces de usuario interactivas.
- **JavaScript**: Lenguaje utilizado en el frontend para manejar la lógica de la aplicación y la interacción con el usuario.

APIs y librerias externas:

Para el desarrollo de nuestro sistema, integramos varias APIs y librerías externas que nos permitieron añadir funcionalidades clave y mejorar la eficiencia en el manejo de datos, la interfaz de usuario y la localización. A continuación, detallamos las principales librerías y APIs empleadas en el proyecto:

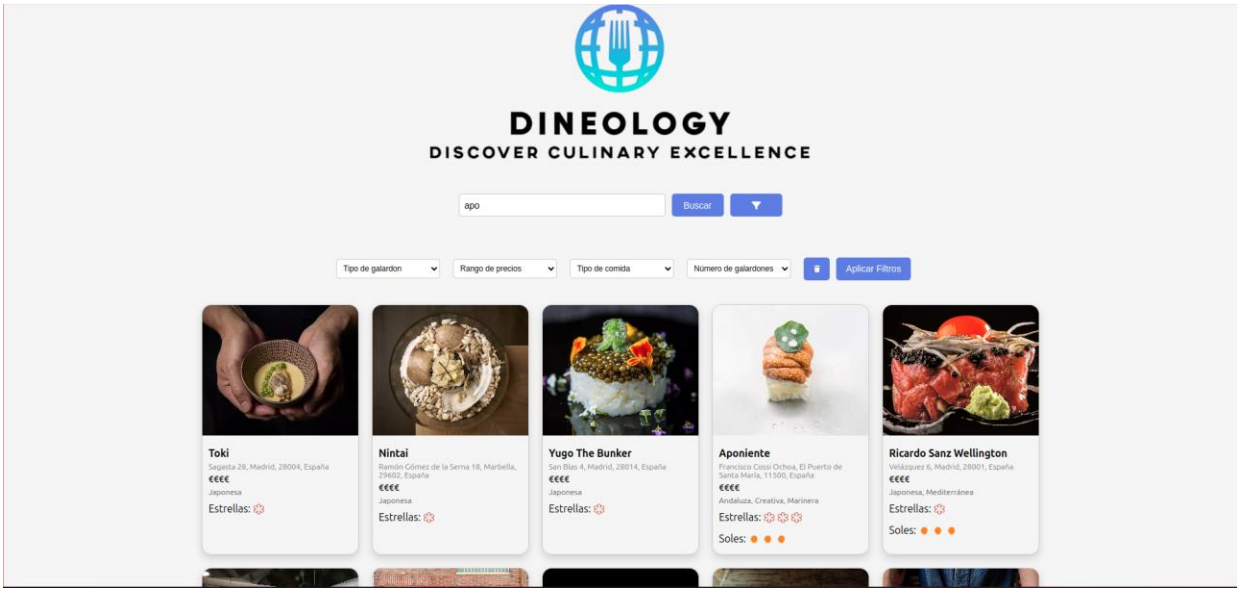
- **Axios:** esta librería fue fundamental para gestionar las peticiones HTTP hacia nuestro backend. Utilizamos esta librería para realizar las peticiones al endpoint de elastic y obtener las respuestas a las queries.

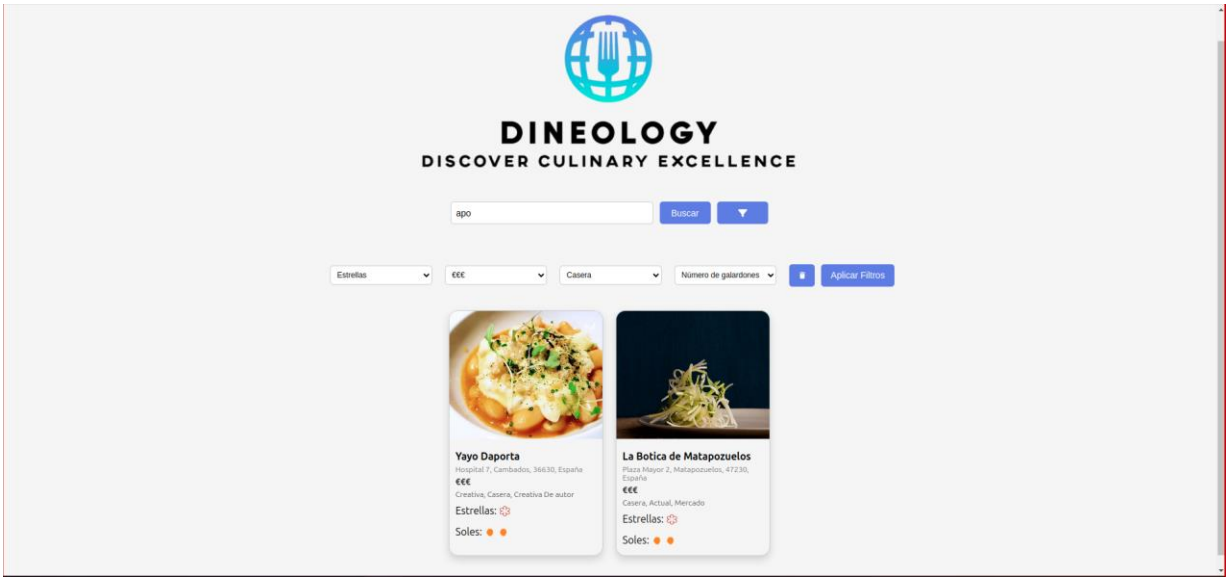
- **FontAwesomeIcon:** para enriquecer la interfaz gráfica y facilitar la comprensión visual de funciones y características, utilizamos esta librería de iconos. Fue clave a la hora de seleccionar iconos para los diferentes botones presentes en el buscador.

- **Nominatim:** nominatim es una API de geocodificación basada en OpenStreetMap que utilizamos para convertir direcciones en coordenadas geográficas (latitud y longitud) y viceversa. Esto fue particularmente útil en nuestro sistema para localizar y representar restaurantes en un mapa de manera precisa, mejorando la experiencia de búsqueda por ubicación.

- **OpenStreetMap:** Como fuente de mapas de código abierto, se integró en el proyecto para proporcionar mapas interactivos y detallados. Utilizando esta plataforma, pudimos mostrar la ubicación exacta de los restaurantes y ofrecer al usuario una experiencia de navegación completa, sin depender de servicios de mapas propietarios.

Dineology:





← Volver

Aponiente

Ángel León es irrepetible: investigador de los secretos del mar. Ha creado una cocina a tono con los pescados antes desconocidos y el plancton..., creando embutidos también de pescado. En su molino de mareas del s. XVII comer es una plenitud de los sentidos con la posibilidad de armonizar los dos únicos menús degustación por Juan Ruiz-Henestrosa (Premio Nacional de Gastronomía mejor sumiller 2015).Todo un espectáculo de saber y sabor. Irrepetible.

Precio: €€€€
Tipo de comida: Andaluza, Creativa, Marinera
Estrellas: ❄❄❄
Soles: ● ● ●
Propietarios: Ángel León
Usuario de Instagram: **@aponiente_angel_leon**
Contacto: 606225859
Dirección: Francisco Cossi Ochoa, El Puerto de Santa María, 11500, España
Visitar Web

Opciones de menú

Tortillita de camarones, caballa, morena, plancton

MENÚ DEGUSTACIÓN	OPCIONES HALAL	OPCIONES PARA CELÍACOS
290.00€	Precio a consultar	Precio a consultar
Mar de Fondo		

Precio a consultar

Servicios del restaurante

Establecimiento accesible	Aparcamiento propio	Aparcacoches	Comedor privado	Buenas vistas
Carta en inglés	Opciones para grupos	Cerca de carretera	Vinos terroir	Organiza eventos
		Huerto propio		

Horario de trabajo

Día	Horario
Lunes	cerrado
Martes	09:00-23:30
Miércoles	09:00-23:30
Jueves	09:00-23:30
Viernes	09:00-23:30
Sábado	09:00-23:30
Domingo	cerrado