

Integración de Sistemas Informáticos

Trabajo Laboratorio



ESCUELA SUPERIOR DE INFORMÁTICA



UNIVERSIDAD DE CASTILLA-LA MANCHA

José Javier Bogado Candia

Gonzalo De Los Reyes Sánchez

Curso 2023/2024

CONTRATO MERCANTIL DE DESARROLLO DE APLICACIÓN Y SERVICIOS ASOCIADOS

Partes Contratantes

Contratante (Cliente): Manuel Sánchez Cristóbal

[Nombre de la empresa o persona física] Bar Manolo

[Dirección] Calle La Mata, 2

[NIF/CIF] 02471382S

Contratista (Desarrollador): Gonzalo De Los Reyes Sánchez y José Javier Bogado Candia

[Nombre de la empresa o persona física] ezMenu SL

[Dirección] UCLM, Ciudad Real.

[NIF/CIF] 932749791E

Objeto del Contrato

El presente contrato tiene como objetivo regular la prestación de servicios de desarrollo de una aplicación y servicios asociados según lo detallado a continuación:

1. Desarrollo de Aplicación:

1.1 Descripción del Proyecto:

El Contratista se compromete a desarrollar una aplicación que permita a los usuarios seleccionar y organizar recetas de comida, generando menús personalizados con códigos QR únicos.

1.2 Funcionalidades Principales:

Búsqueda y selección de recetas.

Creación y personalización de menús.

Generación de menús en formato digital.

Integración con APIs de recetas y generación de códigos QR.

2. Servicios Asociados:

2.1 Integración con APIs:

El Contratista proporcionará servicios de integración con la API de recetas y la API de generación de códigos QR.

2.2 Mantenimiento y Soporte:

El Contratista se compromete a proporcionar servicios de mantenimiento y soporte técnico por un período de 1 año.

3. Pago:

El Contratante se compromete a abonar al Contratista la cantidad de 1.200.000€ de la siguiente manera:

1. Pago Inicial (Fase de Inicio):

Cantidad: 30% del presupuesto total.

Fecha de Pago: A la firma del contrato.

2. Pago Intermedio (Desarrollo de Funcionalidades Clave):

Cantidad: 20% del presupuesto total.

Fecha de Pago: A la entrega de las funcionalidades clave acordadas

3. Pago Parcial (Entrega Parcial del Proyecto):

Cantidad: 20% del presupuesto total.

Fecha de Pago: A la entrega parcial del proyecto, según hitos acordados.

4. Pago Final (Entrega Completa del Proyecto y Aceptación):

Cantidad: 30% del presupuesto total.

Fecha de Pago: A la entrega final y aceptación por parte del Contratante.

4. Facturación:

El Contratista emitirá facturas correspondientes a los servicios prestados, especificando claramente los conceptos y los plazos de pago.

5. Derechos de Propiedad Intelectual:

Los derechos de propiedad intelectual sobre la aplicación desarrollada y los servicios asociados serán propiedad del Contratante cuando se complete el pago total.

6. Confidencialidad:

Ambas partes se comprometen a mantener la confidencialidad de la información sensible intercambiada durante la ejecución del contrato.

7. Vigencia:

El contrato tendrá una vigencia de [especificar duración] a partir de la fecha de firma.

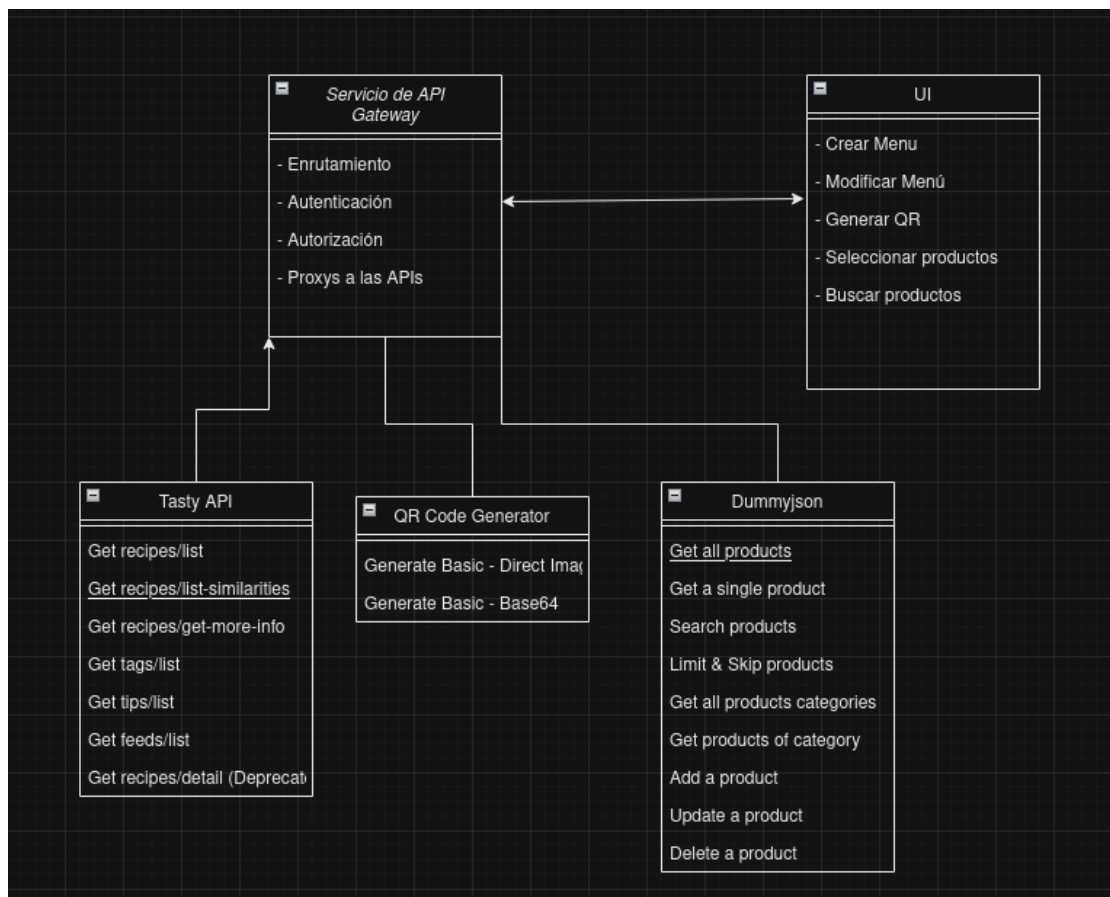
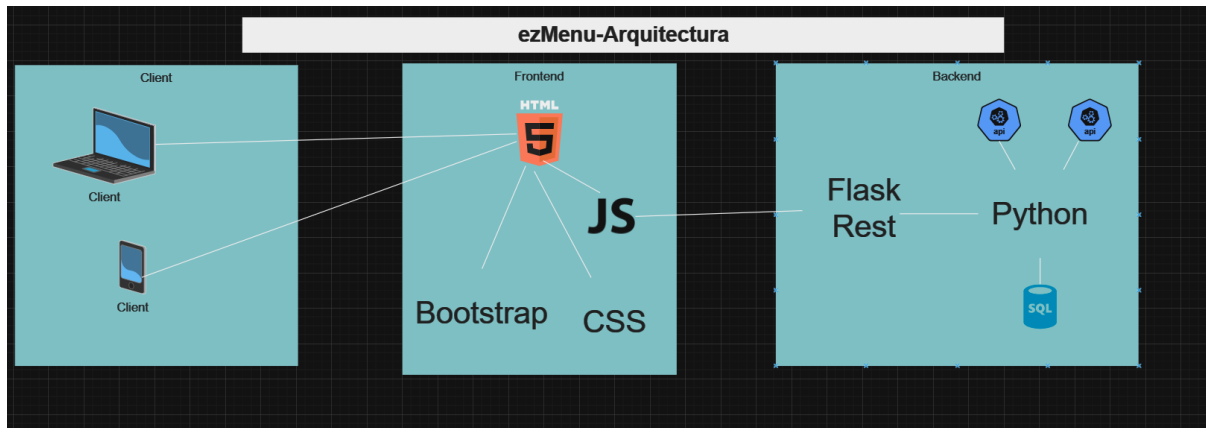
8. Causas de Terminación:

En caso de incumplimiento grave de las obligaciones establecidas en este contrato, podrá terminar el contrato con efecto inmediato.

9. Legislación Aplicable y Jurisdicción:

Este contrato se registrará por las leyes de España y cualquier disputa que surja se resolverá mediante arbitraje de conformidad con las reglas de Departamento Jurídico Español.

● Arquitectura Global



Tecnologías a usar

Python, HTML, CSS, Bootstrap JavaScript y las API. Entorno de Trabajo de Linux, con VS Code.

● Interfaces y estructuras de datos.

Interfaces:

Interfaz de Usuario:

- La interfaz mediante la cual los usuarios interactúan con la aplicación web para buscar y elegir los productos que aparecerán en el menú.
- Debe ser fácil de usar, con una barra de búsqueda y una exposición clara de los productos y los pasos a seguir para crear los menús

APIs Externas:

- Permiten la comunicación con sistemas externos para obtener datos sobre productos y varias funcionalidades.
- Deben proporcionar métodos para consultar y actualizar información.

Estructuras de datos

Productos:

- Representan los elementos que formarán parte de los menús.
- Incluyen los atributos de nombre, imagen y precio.
- Podrían ser representados como objetos en Python, y almacenados en tablas en la base de datos.

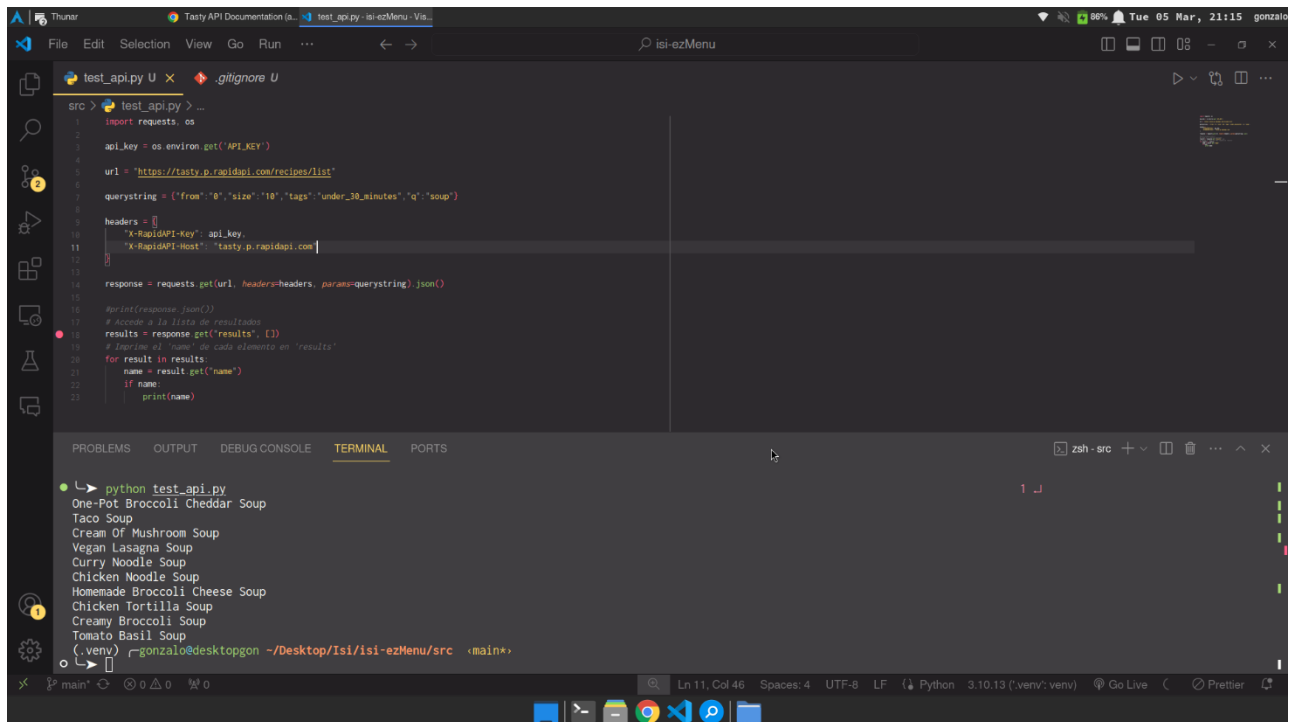
Menús:

- Representan las combinaciones de productos.
- Pueden contener múltiples elementos, cada uno asociado con un producto.
- Podrían ser representados como listas en Python, y almacenados en la base de datos como relaciones entre productos y menús.

● Prototipos/test tecnologías individuales

1. TASTY API

Documentación: <https://rapidapi.com/apidojo/api/tasty>



```
src > test_api.py > ...
1 import requests
2
3 api_key = os.environ.get('API_KEY')
4
5 url = "https://tasty.p.rapidapi.com/recipes/list"
6
7 querystring = {'from': '0', 'size': '10', 'tags': 'under_30_minutes', 'q': 'soup'}
8
9 headers = {}
10
11 'X-RapidAPI-Key': api_key,
12 'X-RapidAPI-Host': 'tasty.p.rapidapi.com'
13
14 response = requests.get(url, headers=headers, params=querystring).json()
15
16 #print(response.json())
17 # Accede a la lista de resultados
18 results = response.get('results', [])
19 # Iteramos el nombre de cada elemento en 'results'
20 for result in results:
21     name = result.get('name')
22     if name:
23         print(name)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
python test_api.py
One-Pot Broccoli Cheddar Soup
Taco Soup
Cream Of Mushroom Soup
Vegan Lasagna Soup
Curry Noodle Soup
Chicken Noodle Soup
Homemade Broccoli Cheese Soup
Chicken Tortilla Soup
Creamy Broccoli Soup
Tomato Basil Soup
(gonzalo@desktopgon ~/Desktop/Isi/isi-ezMenu/src -main*)
```

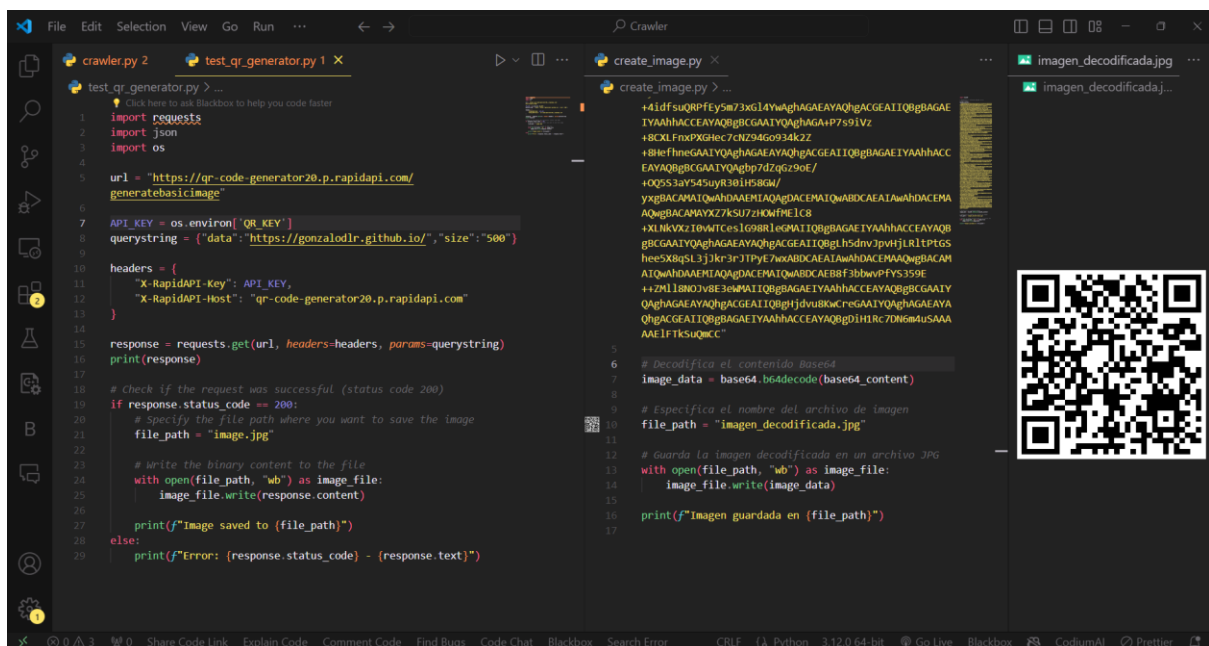
Test API de Tasty.

Testeo de la Api probando a buscar la comida: soup.

Obtengo el Json de los resultados correctamente e imprimo los resultados de solo los nombres encontrados.

2. QR Code Generator

Documentación: <https://rapidapi.com/hydrone/api/qr-code-generator20>



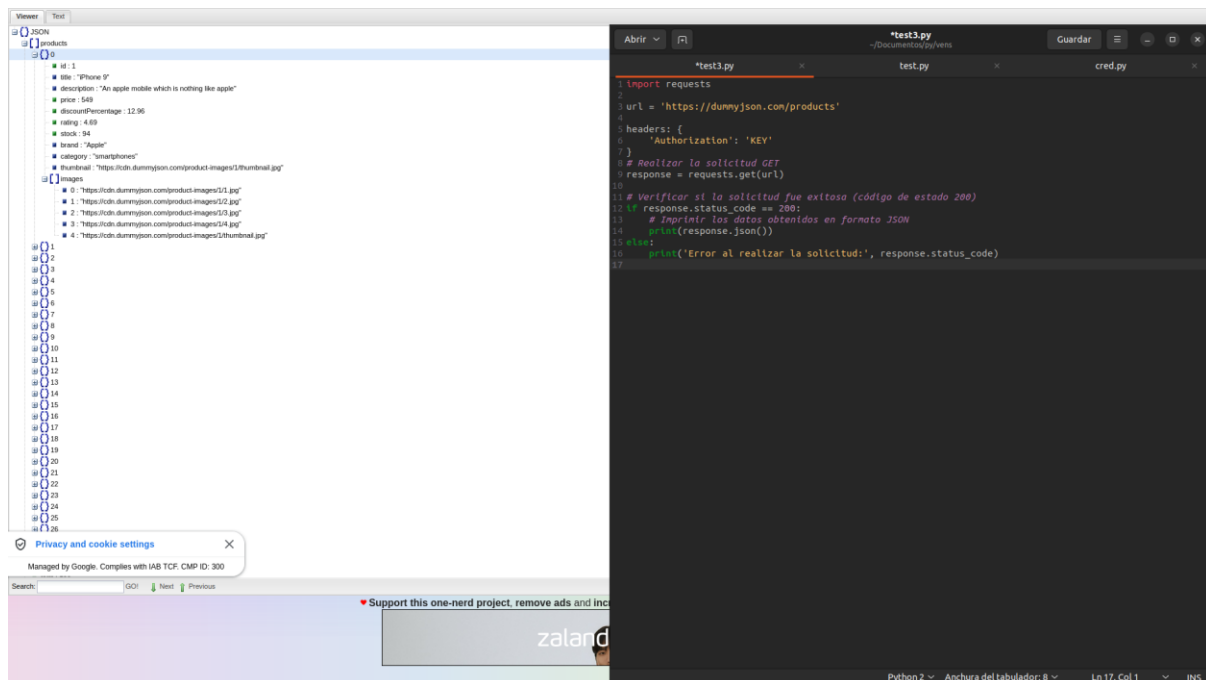
```
test_qr_generator.py > ...
1 import requests
2 import json
3 import os
4
5 url = "https://qr-code-generator20.p.rapidapi.com/generatebasicimage"
6
7 API_KEY = os.environ['QR_KEY']
8 querystring = {'data': 'https://gonzalodr.github.io/', 'size': '500'}
9
10 headers = {
11     'X-RapidAPI-Key': API_KEY,
12     'X-RapidAPI-Host': 'qr-code-generator20.p.rapidapi.com'
13 }
14
15 response = requests.get(url, headers=headers, params=querystring)
16 print(response)
17
18 # Check if the request was successful (status code 200)
19 if response.status_code == 200:
20     # Specify the file path where you want to save the image
21     file_path = "image.jpg"
22
23     # write the binary content to the file
24     with open(file_path, "wb") as image_file:
25         image_file.write(response.content)
26
27     print(f"Image saved to (file_path)")
28 else:
29     print(f"Error: (response.status_code) - (response.text)")
```

```
create_image.py > ...
1
2 +t1dfsuQRPfey5m73x6l4YwAghAGAEAYAQhACGEAI1QBgBAGAEIYAAhHACCEAYAQBgCGAIIYQAgAGAHp7S9IVZ
3 +8CXLfmxPXGhec7cN294Go934K2Z
4 +8WefhneGAIIYQAgAGAEAYAQhACGEAI1QBgBAGAEIYAAhHACC
5 EAYAQBgCGAIIYQAgAgp7d7Q6z9oE/
6 +OQ553av54SuyR30lH58GM/
7 yxqBACAMAYXZ7kSUTzH0wfmELC8
8 +XLNKVXZ10vWTCes1G98R1eGMAIIQBgBAGAEIYAAhHACCEAYAQBgCGAIIYQAgAGAEAYAQhACGEAI1QBgH5dnv3pvHjLR1PtG5
9 hee5X8qsl3jJkr3r3T1PyE7wABDCAEIAwAHDACEMAQwBgBACAM
10 ATQwAHDAEIAQAgDACEMAIQwABDCAEBF3bbwvPFY5359E
11 ++ZML18NOY8E3eWMAIIQBgBAGAEIYAAhHACCEAYAQBgCGAIIYQAgAGAEAYAQhACGEAI1QBgHjdvu8KwcreGAIIYQAgAGAEAYAQhACGEAI1QBgBAGAEIYAAhHACCEAYAQBgD0H1RC7DN6m4USAAA
12 AAE1FTksuQecC"
13
14 # Decodifica el contenido Base64
15 image_data = base64.b64decode(base64_content)
16
17 # Especifica el nombre del archivo de imagen
18 file_path = "imagen_decodificada.jpg"
19
20 # Guarda la imagen decodificada en un archivo JPG
21 with open(file_path, "wb") as image_file:
22     image_file.write(image_data)
23
24 print(f"Imagen guardada en (file_path)")
```

Esta API ha sido más complicada de entender, en la página oficial te dice que la respuesta de la API la recibes en un json: `response = response.json()`. Pero al hacerlo me saltaba error, ya que la API realmente lo que manda es una imagen codificada en base64 dentro de un json. Para solucionar este error hay que decodificar la imagen y volver a crearla tipo jpg. El resultado del test ha sido positivo generando un QR correcto de mi página web.

DummyJSON

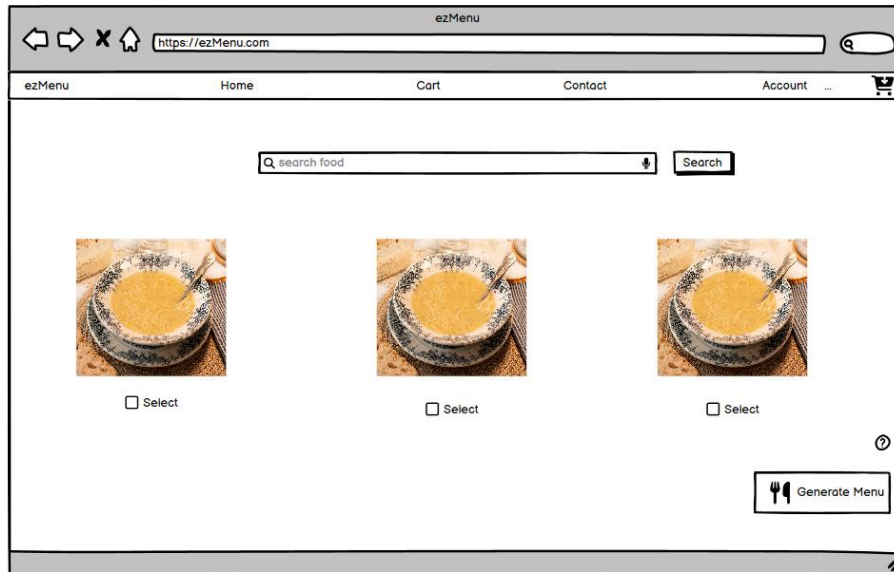
Documentación <https://dummyjson.com/docs>



Prueba de la API, se han añadido unos elementos con una serie de características (id, title, description, price, discountPercentage, rating, stock, brand, category e imagen) y se han pedido de vuelta.

- **Mockup de GUI y user experience**

Prototipo de diseño de la página inicial:



Prototipo con el caso de uso de abrir el carrito de seleccionados:

