Integración con la API de OpenAI:

1. **Obtener una clave de API**: Primero, debes obtener una "key" de API de OpenAI.
2. **Instalar la biblioteca de Python**: Para interactuar con la API de OpenAI desde la aplicacion en mi caso uso el framework Django.

```
pip install openai
```

https://platform.openai.com/docs/libraries/

3. **Importar y configurar el cliente de OpenAI**: En tu archivo views.py de Django, importa la biblioteca de OpenAI y configura el cliente con tu clave de API:

```
from openai import OpenAI



      key='tuclaveAPI'
```

4. **Realizar solicitudes a la API**: Puedes utilizar el cliente de OpenAI para realizar solicitudes a la API y obtener respuestas. Por ejemplo, para generar un texto utilizando el modelo GPT-4o, puedes hacer lo siguiente:
   Nota: este es un ejemplo de una solicitud

```
client = OpenAI(api_key=key)

        messages = [
                {"role": "system", "content": "Eres un experto en análisis de
emprendimientos emergentes."},
                {"role": "user", "content": f"\n Que es stratup "},
      ]

        completion = client.chat.completions.create(
                model="gpt-4o",
                messages=messages
        )

        message_content = completion.choices[0].message.content
        print(message_content)
```

Implementación de la API OpenAI en Django

En la implementación se usa la estructura anterior para hacer una petición el api directamente, en el bloque de código de *messages* contendrá el promt (instrucciones)

Las estructuras de estos promts se deben definirse con la estructura de diccionarios de python.

Se empieza asignado *role* y después *content* que seria las instrucciones

Roles:

- **System**: este rol se asigna cuando quieres especificar a la api que el contenido del promt es un comportamiento y regla que debe seguir la IA.
- **User**: este rol se asigna cuando son peticiones, datos e información que deseas proporcionar a la IA.

```
messages = [
                {"role": "system", "content": "Eres un experto en análisis de
emprendimientos emergentes."},
                {"role": "user", "content": f"\n Que es startup?"},
      ]
```

Graficar usando api chatgpt

Para graficar usando la API se debe pasar un promt con el rol de *system* donde se especifica a la IA que la respuesta debe ser en formato HTML, teniendo en cuenta el análisis anterior se grafica usando javascript haciendo uso de la librería chart

Ejemplo:

```
messages = [
                {"role": "system", "content": "Eres un experto en análisis de
emprendimientos emergentes. analiza el siguiente emprendimiento, proporciona
un análisis detallado."},
                {"role": "system", "content": "genera graficas de barras o
pastel sobre el analisis."},
                {"role": "system", "content": "la respuesta debe ser en
formato HTML. las graficas deben ser scripts usando la libreria chart para ser
visualizadas."},
                {"role": "user", "content": f"\nnombre del
emprendimiento:{proyecto.nombre}, categoria:{proyecto.categoria},
descripcion:{proyecto.descripcion}"},
]
```

Para mostrar la respuesta de la petición en el navegador, defino la variable que almacena la respuesta, en la plantilla html correspondiente a la vista.

Ejemplo:

La variable se llama *message_content* pero para evitar que Django renderice la variable como cadena de texto se pone el filtro `|safe` que indica que el contenido es seguro y puede ser renderizado como HTML.

En la primera linea se importa la libreria “chart” correspondiente a javascript que grafica los datos.

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<div class="container mt-4">
    <div class="row">
        <div class="col-md-12">
            <a class="btn btn-primary" href="/gestionar_proyecto/{{
proyecto.id }}"><i class="fas fa-arrow-left"></i></a>
            {{ message_content|safe }}
          </div>
      </div>
</div>
```

Implementación completa en el proyecto startup simulator usc

View.py

```
def analisis(request, proyecto_id):
        proyecto = Proyecto.objects.get(id=proyecto_id)
        financiero = Financiero.objects.get(proyecto_id=proyecto_id)
        marketing_list = Marketing.objects.filter(proyecto_id=proyecto_id)
        producto_list = Producto.objects.filter(proyecto_id=proyecto_id)
        cargos_empleados =
CargoEmpleado.objects.filter(proyecto_id=proyecto_id)
        identidad = Identidad.objects.get(proyecto_id=proyecto_id)



        client = OpenAI(api_key=key)

        messages = [
```

```
                {"role": "system", "content": "Eres un experto en análisis de
emprendimientos emergentes. analiza el siguiente emprendimiento, proporciona
un análisis detallado."},
                {"role": "system", "content": "genera graficas de barras y
pastel sobre el analisis."},
                {"role": "system", "content": " nota: la respuesta debe ser en
formato html. las graficas deben ser scripts usando la libreria chart para ser
visualizadas."},
                {"role": "user", "content": f"\nnombre del
emprendimiento:{proyecto.nombre},categoria:{proyecto.categoria},
descripcion:{proyecto.descripcion}"},
               {"role": "user", "content": f"\nidentidad:
mision:{identidad.mision}, vision:{identidad.vision},
valores:{identidad.valores}, objetivos:{identidad.objetivos}"},
                {"role": "user", "content": f"\nfinanciero:
ventas:{financiero.ventas}, costos de
produccion:{financiero.costos_produccion}, gastos
administrativos:{financiero.gastos_administrativos}, capital
propio:{financiero.capital_propio}, prestamo:{financiero.prestamo},
inversores:{financiero.inversores}"}
        ]

        for marketing in marketing_list:
                messages.append({"role": "user", "content": f"\nmarketing:
mercado objetivo:{marketing.mercado_objetivo}, segmentacion de
cliente:{marketing.segmentacion_cliente}, canal de
marketing:{marketing.canal_marketing}, estrategia de precio y
promocion:{marketing.estrategia_precio_promocion}, gastos de
marketing:{marketing.gastos_marketing}"})

        for producto in producto_list:
                messages.append({"role": "user", "content": f"\nproducto:
nombre del producto:{producto.nombre_producto}, descripcion del
producto:{producto.descripcion_producto}, categoria del
producto:{producto.categoria_producto}, ciclo de vida del
producto:{producto.ciclo_vida}, costo de
desarrollo:{producto.costo_desarrollo}, costo de
produccion:{producto.costo_produccion}, precio de
venta:{producto.precio_venta}"})

        for cargo in cargos_empleados:
                messages.append({"role": "user", "content": f"\nrecursos
humanos: nombre del cargo:{cargo.nombre_cargo}, descripcion del
cargo:{cargo.descripcion_cargo}, requisitos del
cargo:{cargo.requisitos_cargo}, salario:{cargo.salario}, numero de
empleados:{cargo.numero_empleados}"})
```

```
        completion = client.chat.completions.create(
                model="gpt-4o",
                messages=messages
        )

        message_content = completion.choices[0].message.content
        print(message_content)

        return render(request, 'proyecto/modulos/analisis/analisis.html',
{'proyecto': proyecto, 'financiero': financiero, 'marketing_list':
marketing_list, 'producto_list': producto_list, 'cargos_empleados':
cargos_empleados, 'identidad': identidad, 'message_content': message_content})
```

Template.html

```
{% include 'components/navbar.html' %}
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"><
/script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<div class="container mt-4">
    <div class="row">
        <div class="col-md-12">
            <a class="btn btn-primary" href="/gestionar_proyecto/{{
proyecto.id }}"><i class="fas fa-arrow-left"></i></a>
            {{ message_content|safe }}
          </div>
      </div>
</div>
```