

Diseño y desarrollo de una página web para optimizar la comunicación y el proceso de contratación

Jennifer Andrea Quiceno Aristizábal

Ingeniería de Software

Cod. Estudiante: 100322430

Profesor

David Seligmann

Trabajo de Grado

Politécnico Grancolombiano

Bogota D.C, 2024

Contenido

1.	TÍTULO DE LA PROPUESTA	3
1.1.	INTRODUCCIÓN.....	3
2.	PLANTEAMIENTO DEL PROBLEMA	4
3.	OBJETIVOS	4
4.	JUSTIFICACIÓN	5
5.	MARCO TEÓRICO.....	6
5.1.	ESTADO DEL ARTE.....	7
6.	METODOLOGÍA.....	8
6.1.	FASES DE DESARROLLO.....	8
6.2.	PLANIFICACIÓN	9
6.3.	DISEÑO	9
6.4.	IMPLEMENTACIÓN	9
6.5.	PRUEBAS.....	9
7.	RESULTADOS	9
7.1.	PLANTIFICACIÓN.....	10
7.2.	DISEÑO.....	11
7.2.1.	Modelo de Entidad Relación	11
7.2.2.	Mockups.....	13
7.3.	IMPLEMEMENTACIÓN	17
7.3.1.	BACKEND.....	17

7.3.2.	Base de datos.....	23
7.3.3.	Frontend	23
8.	CRONOGRAMA.....	33
9.	CONCLUSIONES	34

1. TÍTULO DE LA PROPUESTA

Diseño y desarrollo de una página web para optimizar la comunicación y el proceso de contratación

1.1. INTRODUCCIÓN

Las grandes y pequeñas compañías siempre han elegido a sus empleados, por eso, en el proceso de contratación agregan filtros para evaluar el perfil del candidato y asegurarse de cumplir con las capacidades y conocimientos necesarios para el cargo. Según un artículo de Buk(2022), “una comunicación efectiva para los procesos de reclutamiento y selección” es esencial para mantener a los candidatos informados durante el proceso.

Antes, la mayoría de los trabajos requerían más esfuerzo físico o presencia en el lugar de trabajo, hoy hay áreas de empleo donde se permite el trabajo remoto y manejo de tecnologías para sistematizar y automatizar procesos, lo que se refleja en el estudio de Eric (2021) sobre “9 estrategias para conquistar los mejores talentos a través del reclutamiento digital” que señala, en algunos casos, que los empleados ya no se encuentran en las oficinas sino en un trabajo remoto.

En vista de lo anterior, algunas compañías han cambiado el proceso de elección de candidatos y han optado por apoyarse en aplicaciones o sistemas para gestionar filtros de contratación como se menciona en el blog De La Cueva - Ceo Billin (2024) sobre "Digitalización de procesos administrativos: Beneficios". Además, algunas empresas realizan un proceso de contratación totalmente remoto, Sin

embargo, en ocasiones, los reclutadores no informan sobre el estado del proceso de contratación, lo que genera incertidumbre acerca de la continuidad del proceso.

Aunque hoy existen múltiples plataformas y sitios web donde se realizan y se puede monitorear el proceso de contratación, no es tan claro como lo quisiera un candidato, por lo que se necesita una página web que motive al reclutador a exponer el estado de contratación y ayude al candidato a informarse sobre su proceso de contratación. La calidad de la contratación se puede medir y mejorar, tal como se indica en un artículo de Emi Blog (2023), y se deben considerar los problemas que pueden surgir en el reclutamiento y selección de personal (Del Val, 2024).

2. PLANTEAMIENTO DEL PROBLEMA

Todas las empresas usan filtros y procesos de contratación distintos, actualmente muchos se realizan de forma remota o con ayuda de aplicaciones, pero no se preocupan por dar información al candidato. En muchas ocasiones el candidato envía su hoja de vida o se postula a algún cargo y nunca recibe respuesta, o, por el contrario, siempre se recibe una respuesta genérica en donde se menciona algo relacionado con que “El perfil no es el adecuado para el cargo” o “Guardaremos su información en nuestra base de datos para próximas ofertas de empleo”. Teniendo en cuenta lo anterior es importante preguntarse ¿Puede una página web mejorar la comunicación, retroalimentación y proceso de contratación de una empresa informando de forma recurrente al candidato?

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Diseñar y desarrollar una página web para optimizar la comunicación y proceso de contratación mediante la implementación de lenguajes de programación y bases de datos.

3.2. OBJETIVOS ESPECÍFICOS

- Realizar el levantamiento de requerimientos y arquitectura para orientar el desarrollo del software mencionado.

- Implementar un sistema de registro y seguimiento de candidatos que facilite la gestión de postulaciones, entrevistas y evaluaciones.
- Diseñar una interfaz de usuario intuitiva tanto para reclutadores como para candidatos, mejorando la experiencia del usuario y la facilidad de uso.
- Desarrollar un sistema de notificaciones automáticas para mantener a los candidatos informados sobre el estado de su postulación y sus respectivas observaciones.
- Documentar la funcionalidad del sistema construido.

4. JUSTIFICACIÓN

El interés por las páginas web y aplicaciones ha aumentado en los últimos años. Más allá de esto, se han transformado en una herramienta usada por la mayoría de las personas y/o empresas a nivel mundial. Esto se debe a la accesibilidad de estas plataformas, considerando, también, las funcionalidades que ofrece para sistematizar y automatizar procesos y de este modo, reducir carga laboral.

Es importante el considerar tener una aplicación web como apoyo para procesos administrativos, tal como menciona De La Cueva - Ceo Billin (2024) “La digitalización de los procesos administrativos no solo ayuda a optimizar la operación interna de una empresa, también facilita la toma de decisiones más informada y ágil.”

Ahora hay que considerar que existen plataformas, redes sociales y páginas web que ayudan con el proceso de contratación de un candidato, pero lo que se busca con el desarrollo de esta página web es enfocarse con mayor comunicación entre el candidato y el reclutador o reclutadores. “Es muy útil implementar el uso de herramientas en las que todos los involucrados accedan a la misma información, puedan compartir los avances y mantengan una comunicación fluida. Una buena comunicación no solo genera una buena experiencia para el candidato, sino que también para tu equipo.” (Buk, 2022)

5. MARCO TEÓRICO

Como se ha mencionado anteriormente, es importante tener una buena comunicación con el candidato durante su proceso de contratación o postulación. Para entender mejor este proceso, algunos conceptos claves son:

- Optimización de la comunicación: Mejorar como se comparte la información entre los candidatos y reclutadores, para que sea rápida y clara
- Reclutamiento digital: Proceso de encontrar, atraer y contratar candidatos mediante medios digitales.

Un ejemplo de esto son plataformas como LinkedIn, que permiten mantener una comunicación directa con el candidato, pero ¿Es verdaderamente necesario usar una página web como apoyo para el proceso de contratación? Eric (2021) dice: “La estrategia de reclutamiento digital más importante es ajustarse a esta realidad. Te permite conocer mejor a tus clientes potenciales y entender dónde se encuentran para así elaborar el mensaje perfecto que van a leer antes de encontrarte. “.

Teniendo en cuenta todo lo anterior, se puede apreciar la importancia de una página web para la contratación de empleados y Buk nos menciona la importancia hacer uso de herramientas para lograr una buena comunicación, pero ¿Por qué? Buk (2022) también nos menciona que “Con una buena comunicación es posible compartir los diversos procesos de selección que se llevan a cabo en la compañía. De esta forma todos al interior de la organización están involucrados e informados, lo que ayuda a tomar decisiones más acertadas al momento de contratar.

Las tecnologías creadas para los procesos de reclutamiento y selección permiten escoger candidatos de una forma fácil, rápida y económica. Además de que brindan soluciones para los distintos flujos de información entre los involucrados en todo el proceso.”.

Claro, todo el enfoque que se lleva a cabo con el desarrollo de una página web trae consigo un beneficio para el candidato debido a que estará informado de forma recurrente, pero como se aprecia

en el texto anterior, esto también trae beneficios a la empresa que se está encargando del proceso de contratación para la elección de un buen candidato, y no sólo esto sino también es un beneficio económico ya que “encontrar el mejor talento para tus vacantes resulta efectivo en cuanto a costos dado que reduce las posibilidades de rotación [...]. Como resultado, mejorar la calidad en la contratación tiene beneficios importantes para tus resultados y éxito a largo plazo.” (“Calidad De La Contratación: Cómo Medirla Y Consejos Para Mejorarla,” 2023).

5.1. ESTADO DEL ARTE

Muchas personas, día a día lidian con un mal proceso de contratación debido a la falta de comunicación en donde existe la ausencia de información, comentarios y seguimiento durante este proceso. Como nos menciona Del Val (2024) “Alargar un proceso de selección más de lo debido no es beneficioso. En todo caso, hay que dar feedback a los candidatos, informándoles sobre el mismo. Y es que, a falta de información, un candidato puede perder el interés o encontrar otra oferta que le interese.”.

Amedirh and Amedirh (2019) también nos menciona algo muy similar a lo anterior, pero en este caso tratamos con algunas cifras “El candidato ve una vacante por internet, captura su CV, lo contactan y todo parece ir bien. Le prometen llamar al día siguiente, pero eso no pasa. Solo escucha silencio, después vienen las dudas (¿qué hice mal?) y la molestia por la falta de seguimiento. Incluso si esa persona recibe una llamada luego de una o dos semanas, las posibilidades de que esté abierto o entusiasmado serán más escasas. [...] Ciertamente, eso se refleja en altos niveles de rotación. De hecho, en empresas de seguridad privada ese número llega a ser de 80% anual. Ante ese problema, es necesario hacer una reflexión de la importancia de mantener una buena comunicación durante el proceso de reclutamiento de principio a fin.”.

6. METODOLOGÍA

Hacer uso de una metodología en el desarrollo de un proyecto es muy importante para definir prioridades, tener control y conocimiento del problema y tomar mejores decisiones teniendo en cuenta los obstáculos o imprevistos que puedan presentarse durante el desarrollo.

Teniendo en cuenta el caso que se presenta en este documento se opta por usar la metodología Kanban para el desarrollo de este proyecto.

“La metodología Kanban es un sistema de gestión de tareas que forma parte de las metodologías ágiles. Su propósito principal es supervisar y optimizar el flujo de trabajo desde el inicio hasta la finalización de las tareas, asegurando un proceso continuo y eficiente.” (Apd, 2024)

Mediante un tablero Kanban se podrá tener un mayor control del proceso y objetivos del proyecto, además de que se pueden ir corrigiendo errores (en caso de presentarse) sobre la marcha.

6.1. FASES DE DESARROLLO

Para el desarrollo de este proyecto se hará uso de las cuatro primeras etapas del ciclo del desarrollo de software.

6.2. PLANIFICACIÓN

En la etapa de planificación, como su nombre lo menciona, se realizará todo el análisis inicial del proyecto en donde se verá reflejado el levantamiento de requisitos y la toma de decisión sobre que lenguaje o lenguajes usar para la construcción de la página web.

6.3. DISEÑO

La etapa de diseño servirá como base para la construcción de la página web, en esta etapa se realizará un modelo entidad relación y un mockup con páginas principales para brindar información y una referencia sobre cómo será desarrollada la interfaz del usuario. El modelo de entidad relación tiene como objetivo estructurar de forma correcta la aplicación con el fin de que cumpla su objetivo.

6.4. IMPLEMENTACIÓN

La fase de implementación es probablemente la más larga debido a que en esta se realizará la creación de la base de datos que fue presentada en el modelo de entidad relación, además de la creación de los componentes de backend y frontend, además de la integración de cada una de las partes mencionadas.

6.5. PRUEBAS

Las pruebas se realizarán durante y después del desarrollo. Durante el desarrollo con el fin de probar que los servicios funcionen y realizar una integración de componentes de forma correcta. Después del desarrollo se harán pruebas funcionales con el fin de verificar que el software cumpla con el objetivo propuesto.

7. RESULTADOS

A continuación, se expondrán los resultados obtenidos durante el desarrollo del proyecto de grado. Los resultados se han dividido según cada etapa del desarrollo de software.

7.1. PLANTIFICACIÓN

En la primera fase de planificación se realizó un levantamiento de requerimientos funcionales y no funcionales los cuales se representan en la siguiente tabla de dos columnas, donde la primera columna hace referencia a un identificador del requerimiento el cual consta de las iniciales RF (Requerimiento Funciona) y una enumeración, la segunda columna especifica el requerimiento y/o funcionalidad de que se requiere en la aplicación.

Tabla 1

Requerimientos funcionales

Requerimientos funcionales	
RF01	Registrar a los candidatos y reclutadores en la plataforma mediante un formulario.
RF02	Implementar un sistema de inicio de sesión que permita a candidatos y reclutadores ingresar al sistema.
RF03	Permitir a los reclutadores publicar ofertas de empleo para que sean visibles a los candidatos.
RF04	Permitir a los candidatos postularse a ofertas de empleo disponibles en la plataforma
RF05	Permitir a los candidatos actualizar su perfil y hoja de vida en cualquier momento
RF06	Permitir al reclutador visualizar el perfil de candidato.
RF07	Mostrar a los candidatos el estado de su postulación (Ej: En revisión, Entrevista, Rechazado, Aceptado) en su perfil personal.
RF08	Implementar un sistema de notificaciones automáticas por correo electrónico para informar a los candidatos sobre el estado de su postulación
RF09	Permitir a los reclutadores realizar comentarios o retroalimentación sobre su postulación al final del ciclo y mostrar la retroalimentación en el perfil del candidato.
RF10	Incluir un panel de control para reclutadores donde puedan ver todas las postulaciones y su estado.
RF11	Implementar una base de datos que almacene la información de los reclutadores, candidatos, ofertas de empleo y el estado de las postulaciones.
RF12	Implementar un sistema de roles y permisos que permita a los administradores gestionar los accesos de los usuarios (reclutadores, candidatos, administradores).

Nota. Tabla con el listado de los requerimientos funcionales de la aplicación. Tomado de: Autoría propia

El levantamiento de requerimientos no funcionales, como su nombre lo indica, no serán funcionalidades del sistema, sino más bien las necesidades que se tienen para que el sistema funcione, para este caso también se presentará una tabla, muy similar a la de Requerimientos funcionales en

donde la primera columna corresponde a las iniciales RNF (Requerimiento No Funcional) con su respectiva enumeración, y la segunda columna es la especificación del requerimiento no funcional.

Tabla 2

Requerimientos no funcionales

Requerimientos no funcionales	
RNF01	Diseñar una interfaz que sea intuitiva y accesible para todos los usuarios.
RNF02	Implementar autenticación de usuarios como método de seguridad de los datos.
RNF03	Implementar un plan de pruebas que garantice la funcionalidad.
RNF04	Hacer uso de persistencia de datos mediante una base de datos relacional
RNF05	Como primera etapa, el proyecto no tendrá un diseño 100% responsive para ser usado desde dispositivos móviles o pantallas de tamaño pequeño

Nota. Tabla con el listado de los requerimientos funcionales de la aplicación Tomado de: Autoría propia

En el desarrollo del proyecto se definió utilizar Python como lenguaje de programación para el backend haciendo uso de una API, Angular para el frontend, y un gestor de bases de datos relacional SQL.

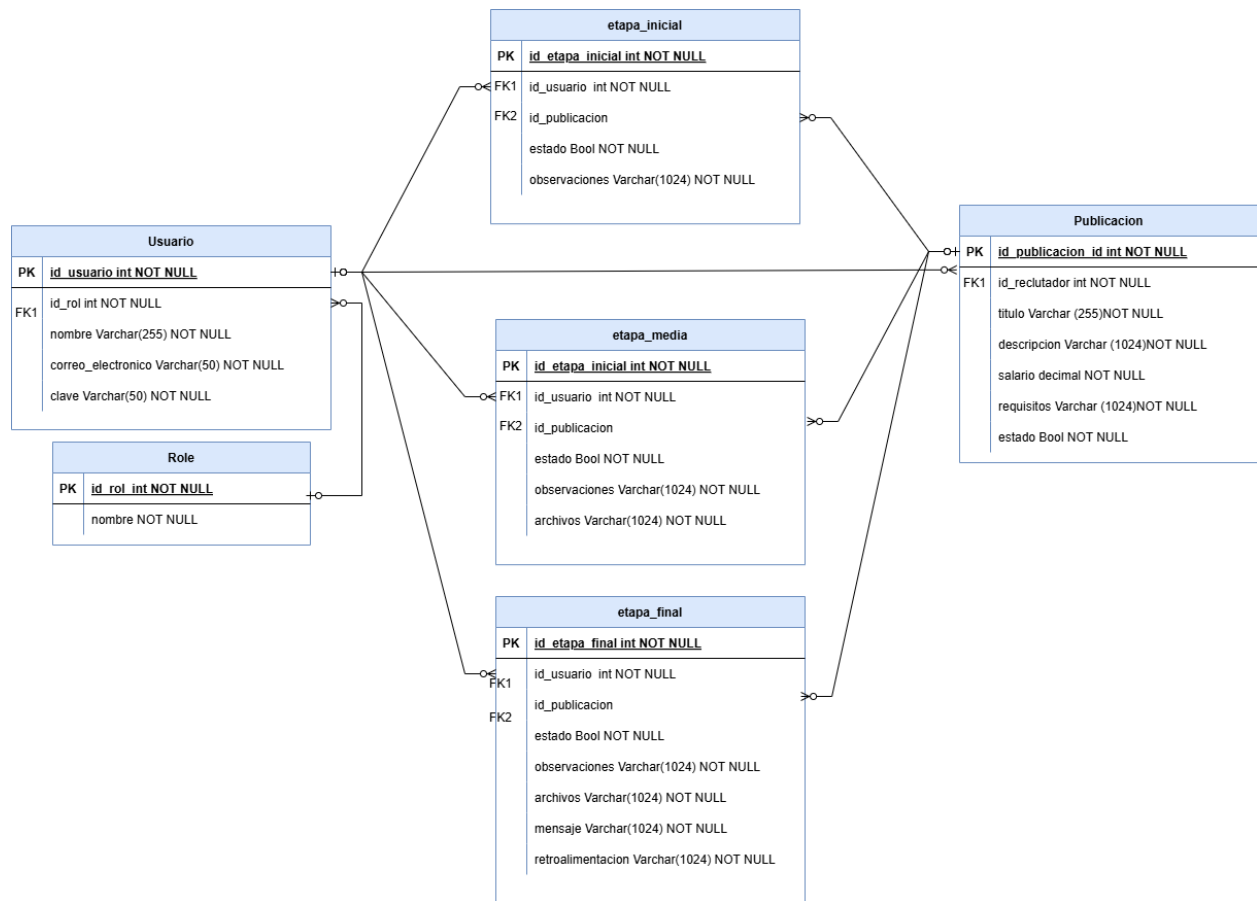
7.2. DISEÑO

7.2.1. Modelo de Entidad Relación

En la fase de diseño, se construyó un modelo de entidad relación con el fin de definir cómo será estructurada la base de datos para la aplicación, de igual forma, esto también tendrá importancia al definir los componentes o módulos de la aplicación.

Imagen 1

Modelo entidad relación



Nota. Modelo entidad relación planteado para el proyecto. Tomado de: Autoría propia

Sumado a esto, también se construyó el diseño de las principales pantallas de la aplicación

7.2.2. Mockups

Se realiza la construcción de un mockup con las principales pantallas que tendrá la aplicación, al ser un mockup o prototipo se busca asimilar el diseño al producto final sin que este sea realmente el diseño que tendrá.

Para la construcción del Mockup se hizo uso de la plataforma de Figma, como primer mockup se tiene la posible representación del inicio de sesión en la aplicación.

Imagen 2

Mokup iniciar sesión



Nota. Mokup planteado para el inicio de sesión de la aplicación. Tomado de: Autoría propia

Al ser una aplicación de ofertas laborales se tendrá una vista en donde se podrán visualizar las ofertas laborales.

Imagen 3

Mokup Ofertas laborales



Nota. Mokup planteado para visualizar las ofertas laborales. Tomado de: Autoría propia

Para los usuarios que decidan realizar una postulación podrán adicionar la información desde una vista similar a la presentada a continuación (Se manejará una interfaz similar para los demás formularios)


Imagen 4

Mokup añadir publicación

We Hire

Menú

Mi perfil



ADMINISTRACIÓN

PUBLICACIONES

Generar Publicación

Consultar Publicaciones

ETAPAS

Consultar Etapa Inicial

Consultar Etapa Media

Consultar Etapa Final

PUBLICACIÓN

ID1

TituloDesarrollador web

Salario2.000.000

Descripción

- En WeHire estamos en la búsqueda de un Desarrollador en WordPress y Shopify, CPanel, PHP y SEO para unirse a nuestro equipo. Buscamos a una persona apasionada por la tecnología y con ganas de innovar.

Requisitos

- Experiencia de 2 años como Desarrollador
- Desarrollo de Páginas Web: Dominio en la creación y diseño de sitios web funcionales y atractivos en WordPress, Shopify u otro CMS.
- Instalación y configuración de temas y plugins en WordPress y Shopify

Publicar

Ubicación

Colombia

Antioquia - Medellín

Avenida 67 # 89-102

Contactanos

Correo : informacion@gmail.com

Whatsapp: 3203285133

Teléfono fijo: 4223480





Nota. Mokup planteado para añadir una publicación. Tomado de: Autoría propia

Como se mencionó anteriormente, la interfaz de los formularios será similar, a continuación, se podrá realizar la edición y/o modificación del perfil del usuario que haya iniciado sesión en la aplicación.


Imagen 5

Mokup Modificar perfil

PublicacionesMis postulacionesMi perfil

MODIFICAR PERFIL

Nombre completo	Correo
Jennifer Andrea Quiceno	jenyaristizabal@gmail.com
Fecha de nacimiento	Ciudad de residencia
06/02/2000	Medellin
Dirección de residencia	Teléfono de contacto
Calle 2 #94-40	Calle 2 #94-40
Contraseña	 Adjuntar hoja de vida, recuerde añadir estudios, experiencia laboral y referencias.
***** 	
 Ver comentarios y retroalimentación	
Modificar y guardar	

**Ubicación**
Colombia
Antioquia - Medellín
Avenida 67 # 89-102

**Contactanos**
Correo : informacion@gmail.com
Whatsapp: 3203285133
Teléfono fijo: 4223480



Nota. Mokup planteado para modificar el perfil. Tomado de: Autoría propia

7.3. IMPLEMENTACIÓN

La implementación se llevó a cabo en el backend utilizando el lenguaje Python junto con el framework Django, y se empleó SQL como gestor de base de datos.

7.3.1. Backend

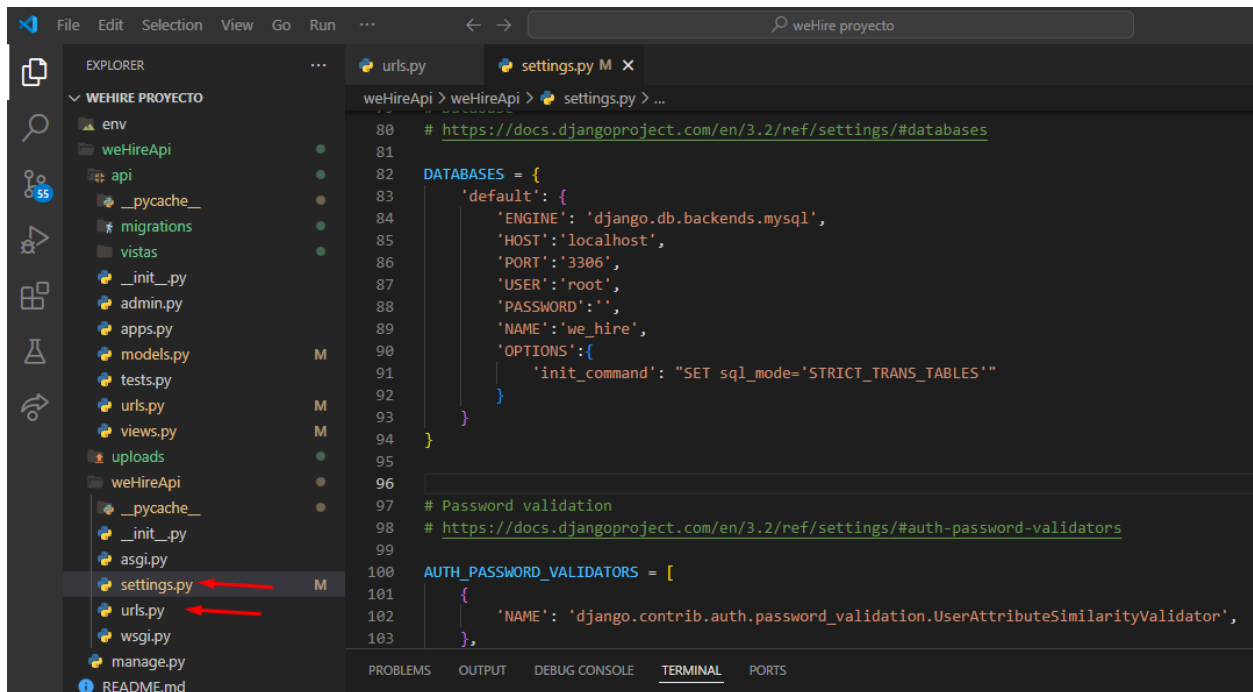
Se creó el proyecto llamado WeHireApi, que contiene los archivos de configuración general necesarios para el funcionamiento del sistema, los cuales son:

settings.py: Gestiona las configuraciones globales del proyecto, como la base de datos, las aplicaciones instaladas y las configuraciones de seguridad.

urls.py: Define las rutas principales del proyecto, permitiendo conectar las solicitudes de los usuarios

Imagen 6

Configuraciones Python



Nota. Archivo de configuración global del proyecto py

El modelo fue desarrollado utilizando el archivo **models.py**, donde se definen las estructuras de datos que serán representadas en la base de datos, como se muestra a continuación

Imagen 7

Archivo de models.py

```

from django.db import models

# Create your models here.
class Rol(models.Model):
    nombre = models.CharField(max_length=50)
    def __str__(self):
        return self.nombre

class Usuarios(models.Model):
    nombre=models.CharField(max_length=50,null=True)
    apellido=models.CharField(max_length=50,null=True)
    correo=models.EmailField(max_length=255)
    clave=models.CharField(max_length=255,null=True)
    fecha_nacimiento = models.DateTimeField(null=True)
    telefono= models.CharField(max_length=50,null=True)
    celular= models.CharField(max_length=50,null=True)
    hoja_vida= models.FileField(upload_to='uploads/', null=True, blank=True)
    rol = models.ForeignKey(
        Rol,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name="usuarios"
    )
    def __str__(self):
        return self.nombre

class Publicaciones(models.Model):
    titulo=models.CharField(max_length=255)
    descripcion=models.TextField(null=True)
    salario=models.DecimalField(max_digits=10, decimal_places=2, null=True)
    requisitos=models.TextField(null=True)
    estado=models.CharField(max_length=255,null=True)
    reclutador = models.ForeignKey(
        Usuarios,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name="publicaciones"
    )
    candidato = models.ForeignKey(
        Usuarios,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name="publicaciones_candidato"
    )
    def __str__(self):
        return self.titulo

class EtapaInicial(models.Model):
    estado=models.CharField(max_length=255,null=True)
    observaciones = models.TextField(null=True)
    usuario = models.ForeignKey(
        Usuarios,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name="etapa_inicial"
    )
    publicacion = models.ForeignKey(
        Publicaciones,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,

```

Nota. Archivo de models para el modelo de WeHire. Tomado de: Autoría propia

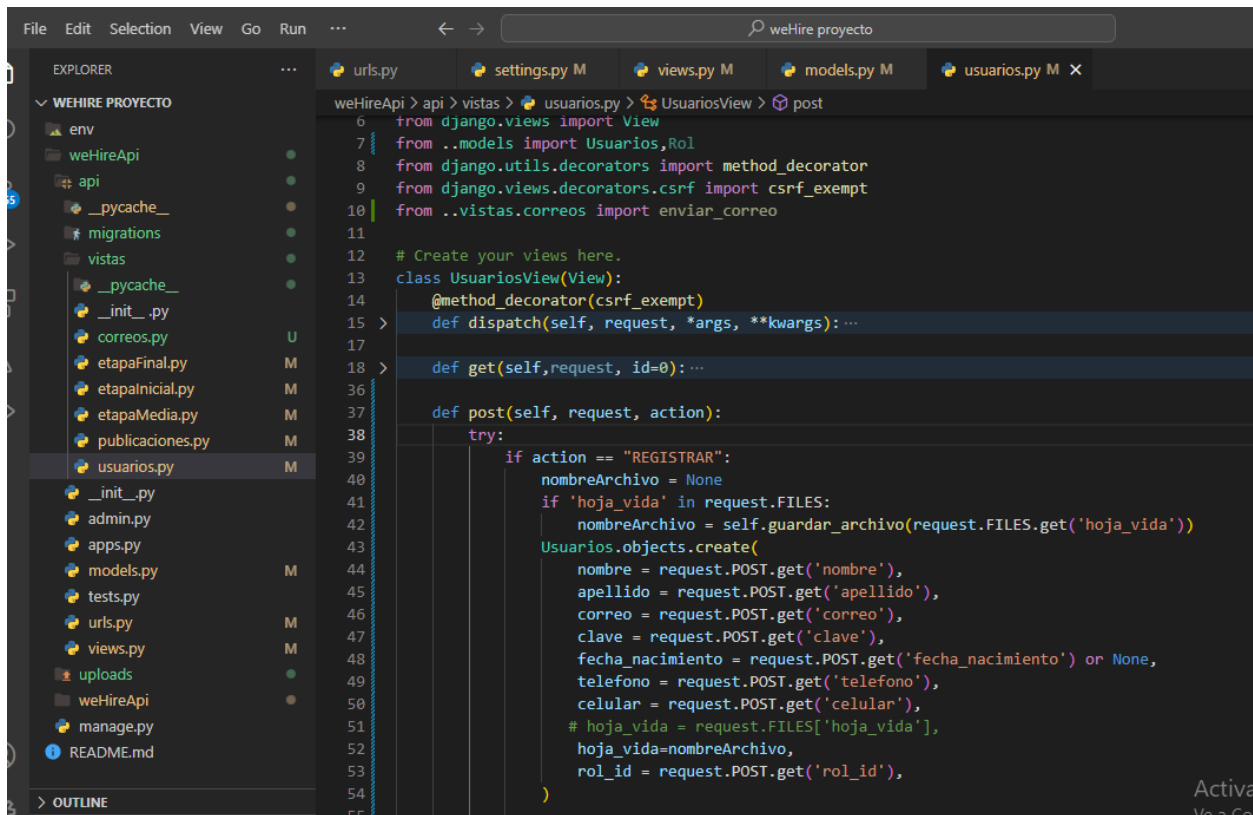
No se hizo uso de templates para las vistas, ya que el proyecto se diseñó como una API. En su lugar se creó una carpeta llamada vistas para gestionar la lógica de negocio, separándola de manera organizada según los modelos planteados.

Por ejemplo, para el modelo de Usuarios, se incluyeron las vistas necesarias para realizar las operaciones principales:

- Creación de usuarios
- Actualización de información
- Listado de usuarios

Imagen 8

Vista de usuarios



```
File Edit Selection View Go Run ... weHire proyecto
EXPLORER
WEHIRE PROYECTO
  env
  weHireApi
    api
    __pycache__
    migrations
    vistas
      __pycache__
      __init__.py
      correos.py
      etapaFinal.py
      etapaInicial.py
      etapaMedia.py
      publicaciones.py
      usuarios.py
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
    uploads
    weHireApi
    manage.py
    README.md
  > OUTLINE

urls.py settings.py M views.py M models.py M usuarios.py M X
weHireApi > api > vistas > usuarios.py > UsuariosView > post
6 from django.views import View
7 from ..models import Usuarios,Rol
8 from django.utils.decorators import method_decorator
9 from django.views.decorators.csrf import csrf_exempt
10 from ..vistas.correos import enviar_correo
11
12 # Create your views here.
13 class UsuariosView(View):
14     @method_decorator(csrf_exempt)
15     def dispatch(self, request, *args, **kwargs): ...
16
17     def get(self,request, id=0): ...
18
19     def post(self, request, action):
20         try:
21             if action == "REGISTRAR":
22                 nombreArchivo = None
23                 if 'hoja_vida' in request.FILES:
24                     nombreArchivo = self.guardar_archivo(request.FILES.get('hoja_vida'))
25                 Usuarios.objects.create(
26                     nombre = request.POST.get('nombre'),
27                     apellido = request.POST.get('apellido'),
28                     correo = request.POST.get('correo'),
29                     clave = request.POST.get('clave'),
30                     fecha_nacimiento = request.POST.get('fecha_nacimiento') or None,
31                     telefono = request.POST.get('telefono'),
32                     celular = request.POST.get('celular'),
33                     # hoja_vida = request.FILES['hoja_vida'],
34                     hoja_vida=nombreArchivo,
35                     rol_id = request.POST.get('rol_id'),
36                 )
37         except:
38             pass
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
```

Nota. Vista para la lógica implementada en usuarios. Tomado de: Autoría propia

De la misma manera, aplica para

- **Publicaciones:** Incluye las vistas necesarias para crear, actualizar y listar las ofertas de empleo publicadas.

Imagen 8

Vista de publicaciones

```

10 class PublicacionesView(View):
11
12     def post(self, request, action):
13         try:
14             if action == "REGISTRAR":
15                 try:
16                     print(f"Datos recibidos: {request.POST}")
17                     Publicaciones.objects.create(
18                         titulo=request.POST.get('titulo'),
19                         descripcion=request.POST.get('descripcion'),
20                         salario=request.POST.get('salario'),
21                         requisitos=request.POST.get('requisitos'),
22                         estado=request.POST.get('estado'),
23                         reclutador_id=request.POST.get('reclutador_id'),
24                         candidato_id=request.POST.get('candidato_id')
25                     )
26                     return JsonResponse({'message': 'UsuPublicaicon ario registrado correctamente'}, status=201)
27                 except Exception as e:
28                     print(f"Error al crear publicación: {e}")
29                     return JsonResponse({'error': 'Error al registrar publicación'}, status=400)
30             elif action == "ACTUALIZAR":
31                 publicacion= list(Publicaciones.objects.filter(id=request.POST.get('id')).values())
32                 if len(publicacion)>0:
33                     publicacion =Publicaciones.objects.get(id=request.POST.get('id'))
34                     publicacion.titulo=request.POST.get('titulo') if request.POST.get('titulo') != None else ;
35                     publicacion.descripcion=request.POST.get('descripcion') if request.POST.get('descripcion') != None else ;
36                     publicacion.salario=request.POST.get('salario') if request.POST.get('salario') != None else ;
37                     publicacion.requisitos=request.POST.get('requisitos') if request.POST.get('requisitos') != None else ;
38                     publicacion.estado=request.POST.get('estado') if request.POST.get('estado') != None else ;
39                     publicacion.save()
40
41         except:
42             return JsonResponse({'error': 'Error al registrar publicación'}, status=400)

```

Nota. Vista para la lógica implementada en publicaciones. Tomado de: Autoría propia

- **Etapas:** Gestiona las diferentes fases del proceso de contratación, permitiendo organizar y seguir el progreso de los candidatos

Imagen 9

Vista etapa Inicial

```

6 from django.views.decorators.csrf import csrf_exempt
7 from django.core.exceptions import ObjectDoesNotExist
8 from ..vistas.correos import enviar_correo
9
10 class EtapasInicialesView(View):
11     @method_decorator(csrf_exempt)
12     def dispatch(self, request, *args, **kwargs):
13
14     def get(self, request, id=0, action=""):
15         if(id>0):
16             etapasIniciales= list(EtapasIniciales.objects.filter(id=id).values())
17             if len(etapasIniciales)>0:
18                 etapasIniciales= etapasIniciales[0]
19                 datos= {'message': "Success",'etapa_inicial': etapasIniciales}
20             else:
21                 datos={'message': "No se encuentra la etapa"}
22             return JsonResponse(datos)
23         elif action=="por_id":
24             etapasIniciales = EtapasIniciales.objects.select_related('usuario', 'publicacion').filter()
25             if len(etapasIniciales)>0:
26                 datos= {'message': "Success","etapa_inicial":etapasIniciales}
27             else:
28                 datos= {'message': "Etapa no encontrada"}
29             return JsonResponse(datos)
30         else:
31             etapasIniciales = list(EtapasIniciales.objects.values())
32             if len(etapasIniciales)>0:
33                 datos= {'message': "Success","etapa_inicial":etapasIniciales}
34             else:
35                 datos= {'message': "No se encuentra la etapa"}
36             return JsonResponse(datos)

```

Nota. Vista para la lógica implementada en etapa inicial. Tomado de: Autoría propia

Imagen 10

Vista de etapa media

```
class EstapaMediaView(View):
    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

> def get(self, request, id=0, action=""): ...

    def post(self, request, action):
        try:
            if action == 'REGISTRAR':
                nombreArchivo = None
                if 'archivo' in request.FILES:
                    nombreArchivo = self.guardar_archivo(request.FILES.get('archivo'))
                data = json.loads(request.body)
                usuario_id = data.get('usuario_id')
                publicacion_id = data.get('publicacion_id')
                estado = data.get('estado')
                observaciones = data.get('observaciones')
                #archivo = data.get('archivo')

                if not usuario_id or not publicacion_id:
                    return JsonResponse({'message': 'Datos incompletos'}, status=400)

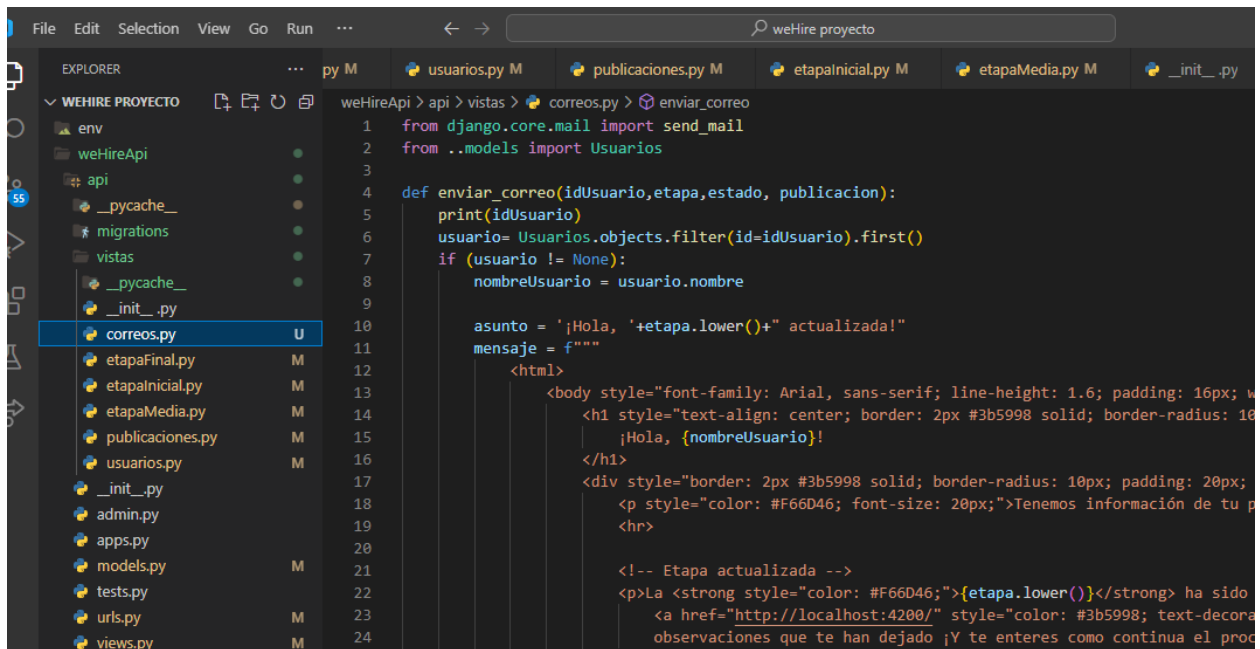
                usuario = Usuarios.objects.filter(id=usuario_id).first()
                if not usuario:
                    return JsonResponse({'message': 'Usuario no encontrado'}, status=404)
```

Nota. Vista para la lógica implementada en etapa media. Tomado de: Autoría propia

- **Envío de correos:** Contiene la lógica para automatizar las notificaciones, como correos de actualizaciones de estado y modificaciones

Imagen 11

Vista correo

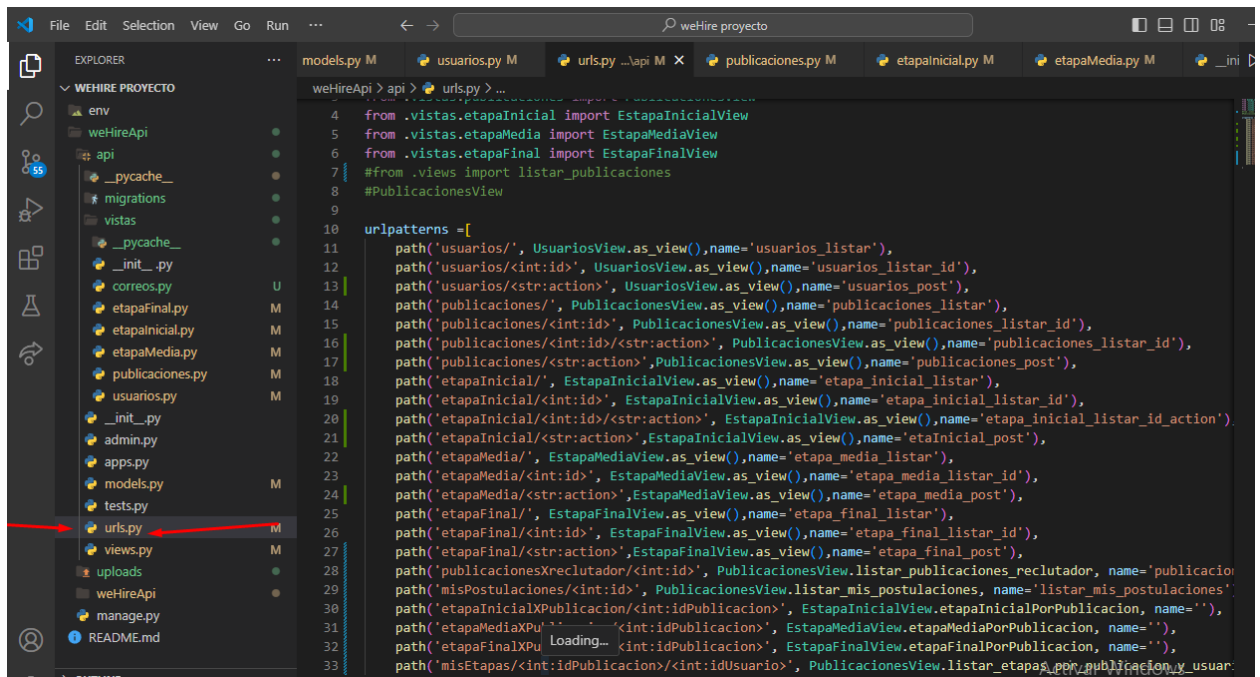


Nota. Vista para la lógica implementada en correos. Tomado de: Autoría propia

Por ultimo y no menos importante se tiene el archivo de URLs, el cual nos ayuda a exponer las vistas como endpoints de la API, las cuales pueden ser consumidas por el frontend o cualquier cliente que interactúe con el sistema.

Imagen

Url PY



Nota. Urls para exponer. Tomado de: Autoría propia

7.3.2. Base de datos

Para el manejo de la base de datos, se está utilizando el gestor phpMyAdmin, a continuación, se muestra la interfaz de phpMyAdmin que refleja cómo se administra la base de datos del proyecto, permitiendo interactuar fácilmente con las tablas y los datos almacenados.

Imagen 12

Base de datos



Nota. Base de datos de weHire. Tomado de: Autoría propia

7.3.3. Frontend

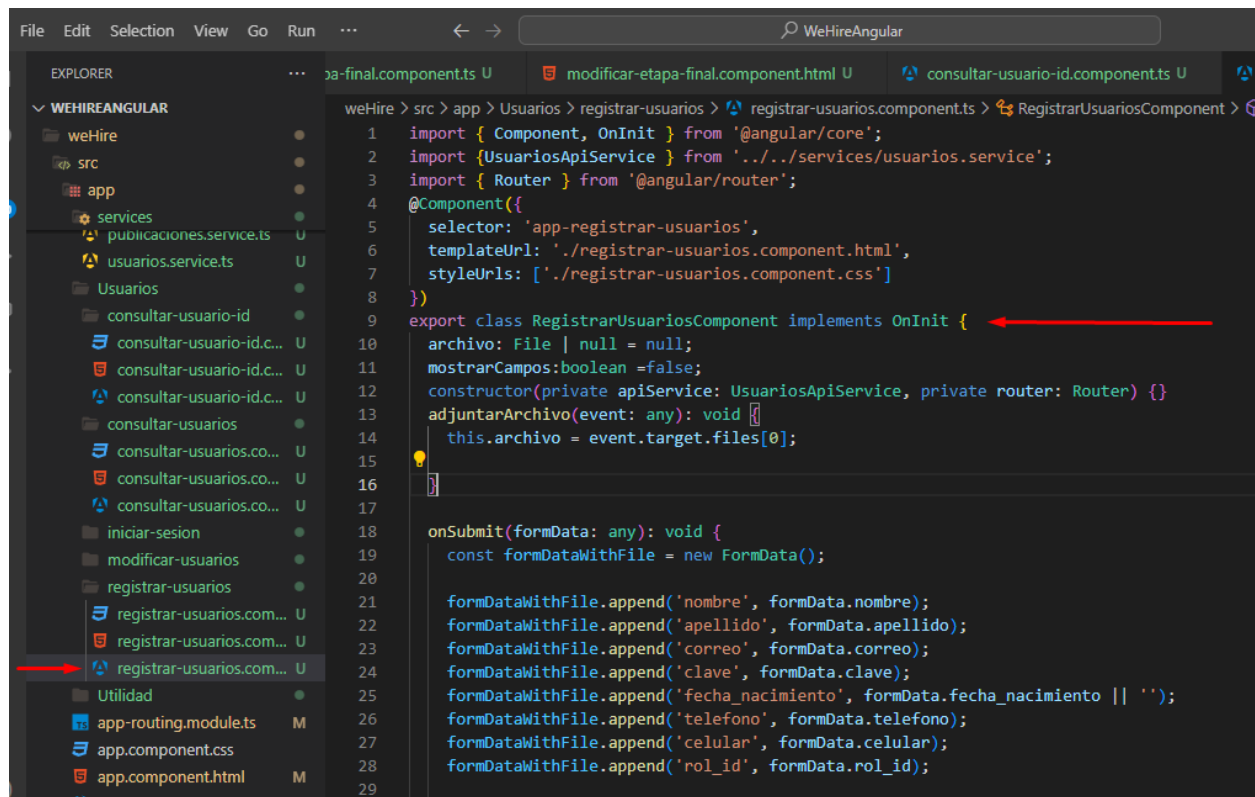
Para el frontend, se utilizó Angular 13 siguiendo una estructura similar a la del backend en Python/Django. La organización del proyecto en Angular se centró en generar un componente por cada modelo

Usuarios: Se crearon componentes para las siguientes operaciones:

- **Registrar Usuario**

Imagen 13

Registrar usuario



The screenshot shows an IDE with the Explorer panel on the left, the Search bar at the top, and the Code editor on the right. The Explorer panel shows the project structure for 'WEHIREANGULAR', with the 'registrar-usuarios' folder selected. The Code editor displays the 'registrar-usuarios.component.ts' file. The code defines the 'RegistrarUsuariosComponent' class, which implements 'OnInit'. It includes imports for '@angular/core', 'UsuariosApiService', and '@angular/router'. The component's metadata includes a selector, templateUrl, and styleUrls. The 'constructor' method takes 'apiService' and 'router' as arguments. The 'adjuntarArchivo' method handles file uploads. The 'onSubmit' method appends form data to a 'FormData' object and sends it to the 'apiService'.

```
1 import { Component, OnInit } from '@angular/core';
2 import { UsuariosApiService } from '../services/usuarios.service';
3 import { Router } from '@angular/router';
4 @Component({
5   selector: 'app-registrar-usuarios',
6   templateUrl: './registrar-usuarios.component.html',
7   styleUrls: ['./registrar-usuarios.component.css']
8 })
9 export class RegistrarUsuariosComponent implements OnInit {
10   archivo: File | null = null;
11   mostrarCampos:boolean =false;
12   constructor(private apiService: UsuariosApiService, private router: Router) {}
13   adjuntarArchivo(event: any): void {
14     this.archivo = event.target.files[0];
15   }
16
17   onSubmit(formData: any): void {
18     const formDataWithFile = new FormData();
19
20     formDataWithFile.append('nombre', formData.nombre);
21     formDataWithFile.append('apellido', formData.apellido);
22     formDataWithFile.append('correo', formData.correo);
23     formDataWithFile.append('clave', formData.clave);
24     formDataWithFile.append('fecha_nacimiento', formData.fecha_nacimiento || '');
25     formDataWithFile.append('telefono', formData.telefono);
26     formDataWithFile.append('celular', formData.celular);
27     formDataWithFile.append('rol_id', formData.rol_id);
28   }
29 }
```

Nota. Componente para registrar usuarios. Tomado de: Autoría propia

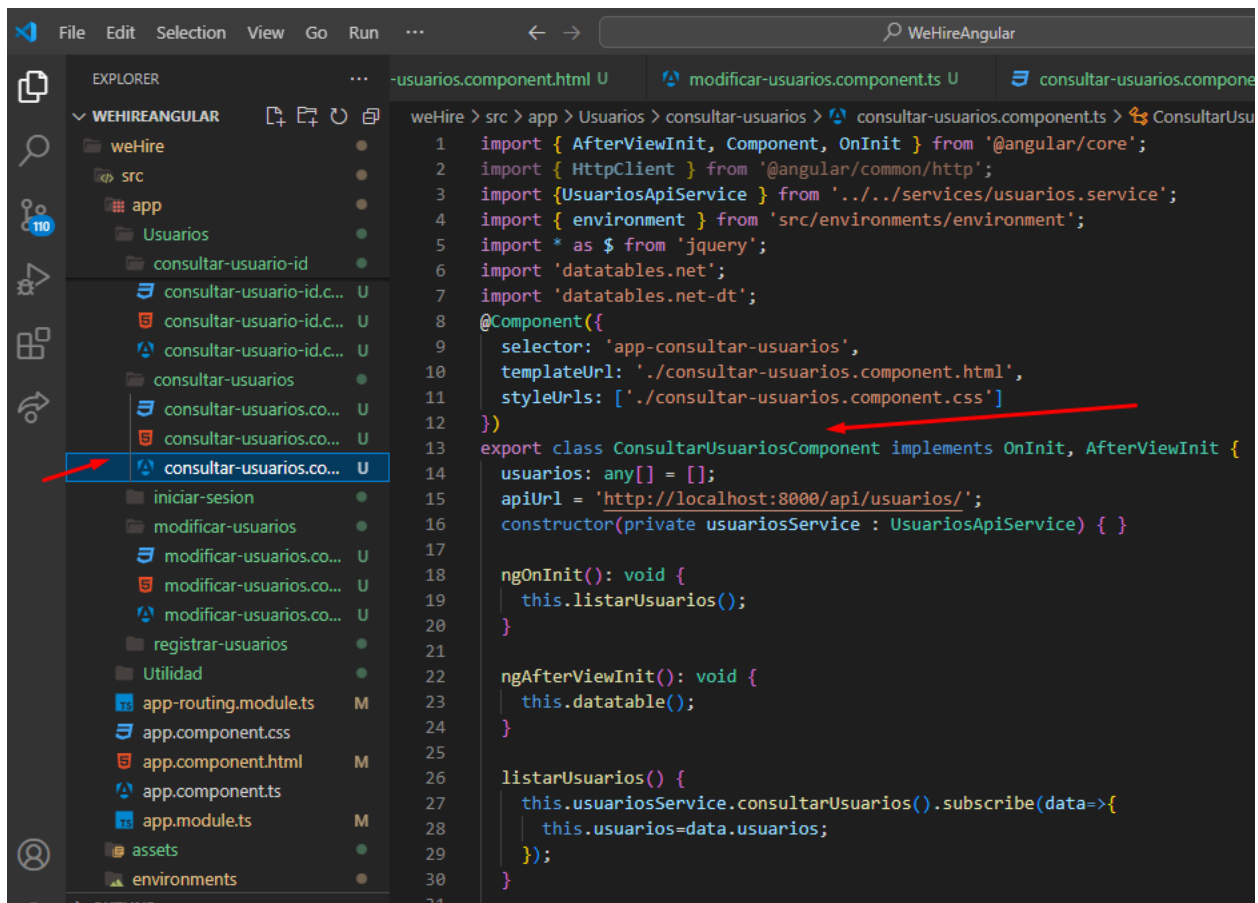
- **Modificar Usuario**

Imagen 14

Modificar usuario



- Imagen 15**
Consultar usuarios

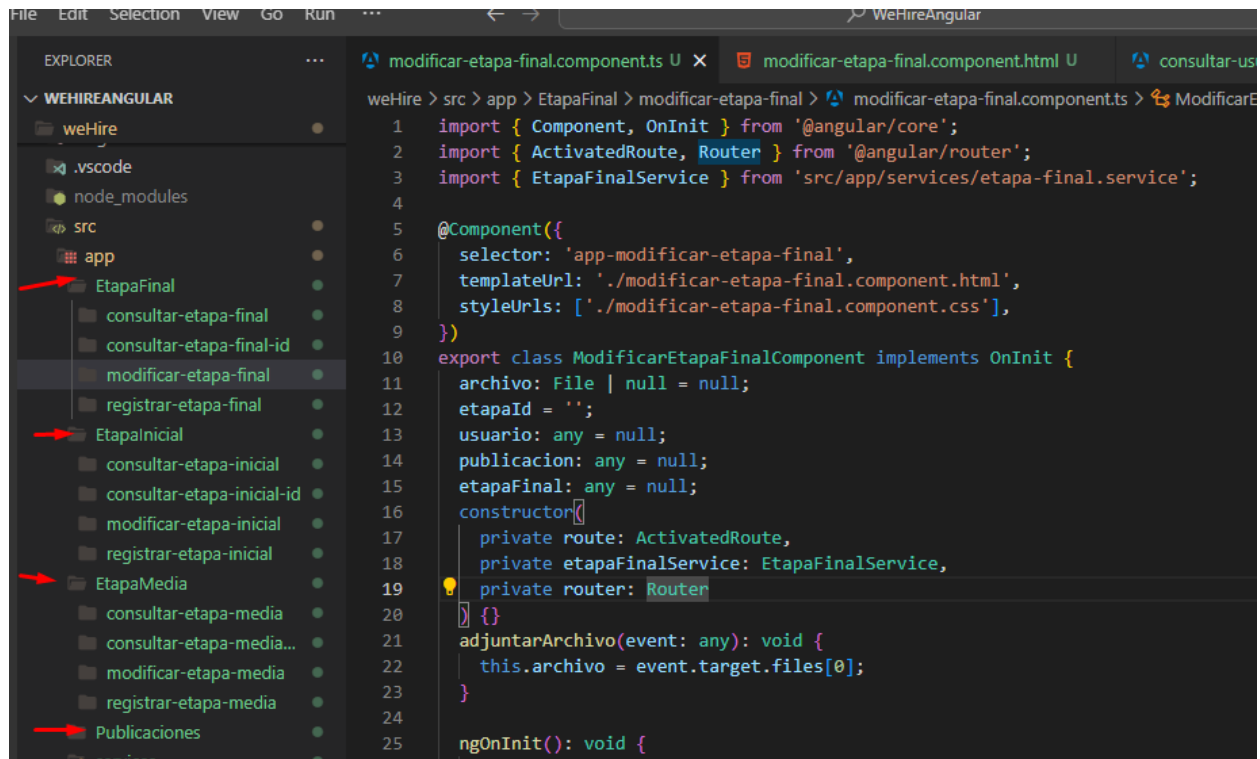


Nota. Componente para consultar usuarios. Tomado de: Autoría propia

De la misma forma, se aplicó la misma estructura modular para los demás componentes relacionados con Publicaciones y Etapas.

Imagen 16

Encarpetado componente



Nota. Encarpetado de los demás componentes . Tomado de: Autoría propia

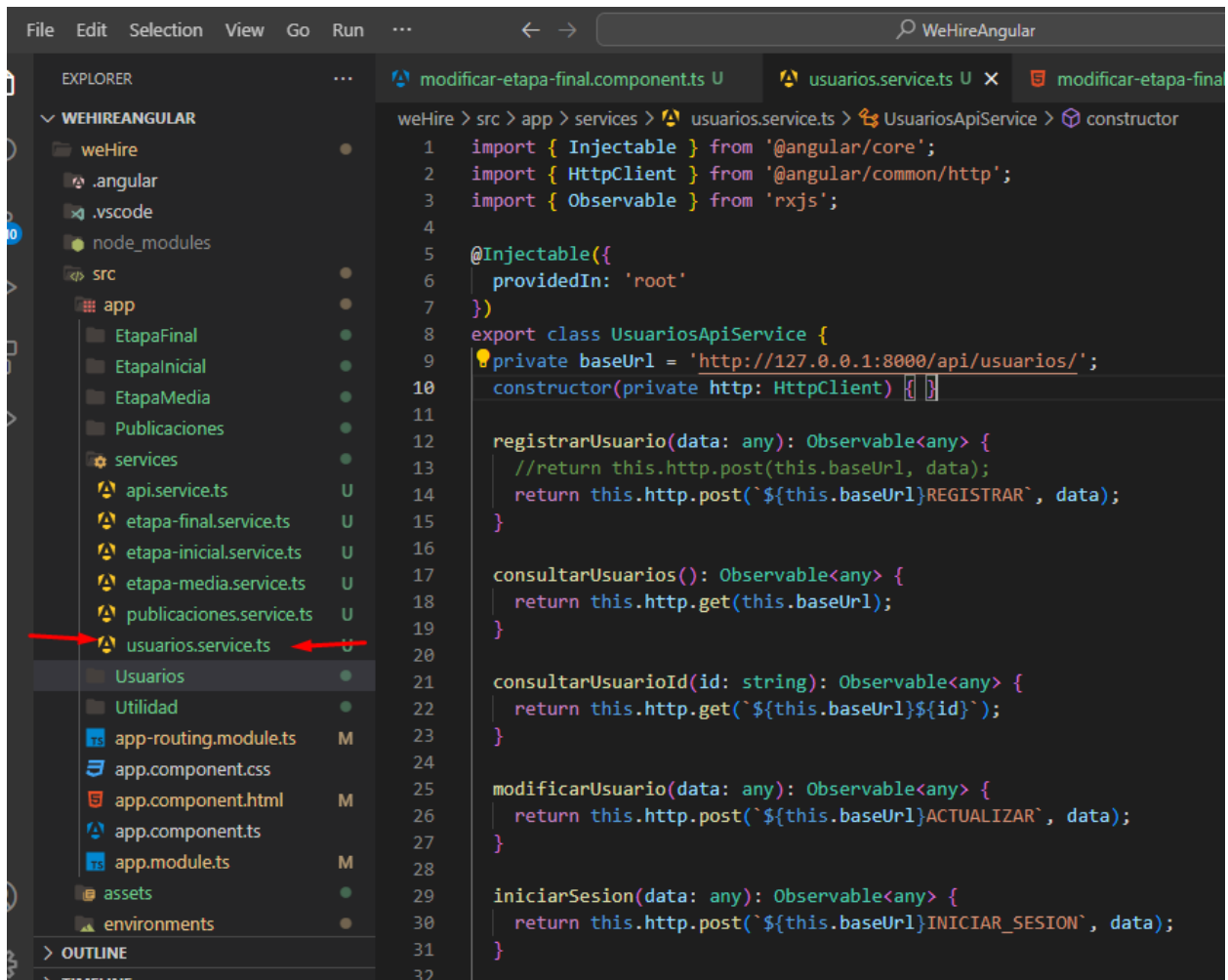
Al crear los modelos en **Angular**, también se generaron los archivos de **servicio** correspondientes para permitir la comunicación con el **backend Django** a través de las **URLs** de la API. Cada servicio fue diseñado para interactuar con su modelo respectivo y gestionar las operaciones consultar, modificar y registrar

Los servicios creados fueron los siguientes:

- **Servicio de Usuarios:** Para gestionar las operaciones relacionadas con los usuarios, como el registro, modificación y consulta.

Imagen 17

Servicio usuario

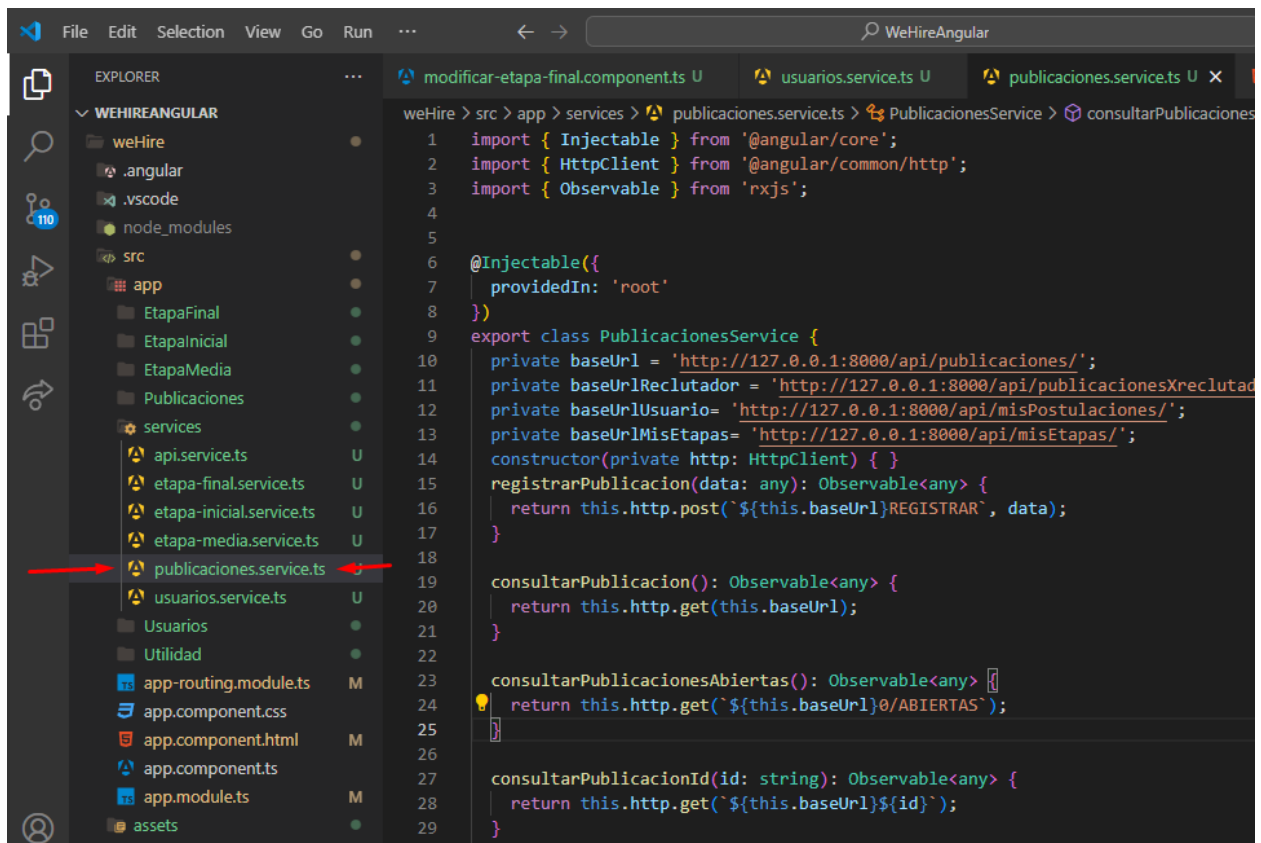


Nota. Servicio de usuarios con los métodos correspondientes. Tomado de: Autoría propia

- **Servicio de Publicaciones:** Para manejar la creación, modificación y consulta de las publicaciones de empleo.

Imagen 18

Servicio publicaciones

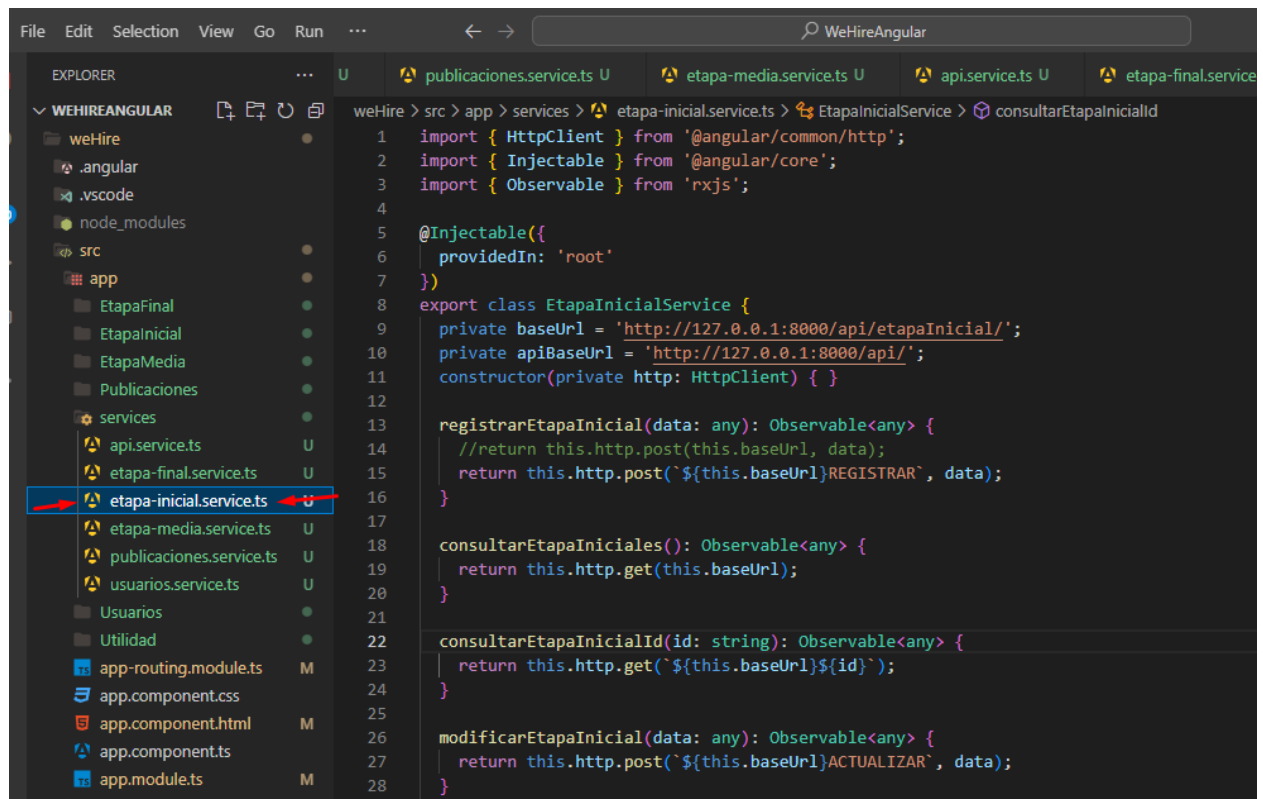


Nota. Servicio de publicaciones con los métodos correspondientes. Tomado de: Autoría propia

- **Servicio de Etapa Inicial:** Para gestionar las operaciones asociadas a la **etapa inicial** del proceso, como el registro, modificación y consulta.

Imagen 19

Servicio etapa inicial

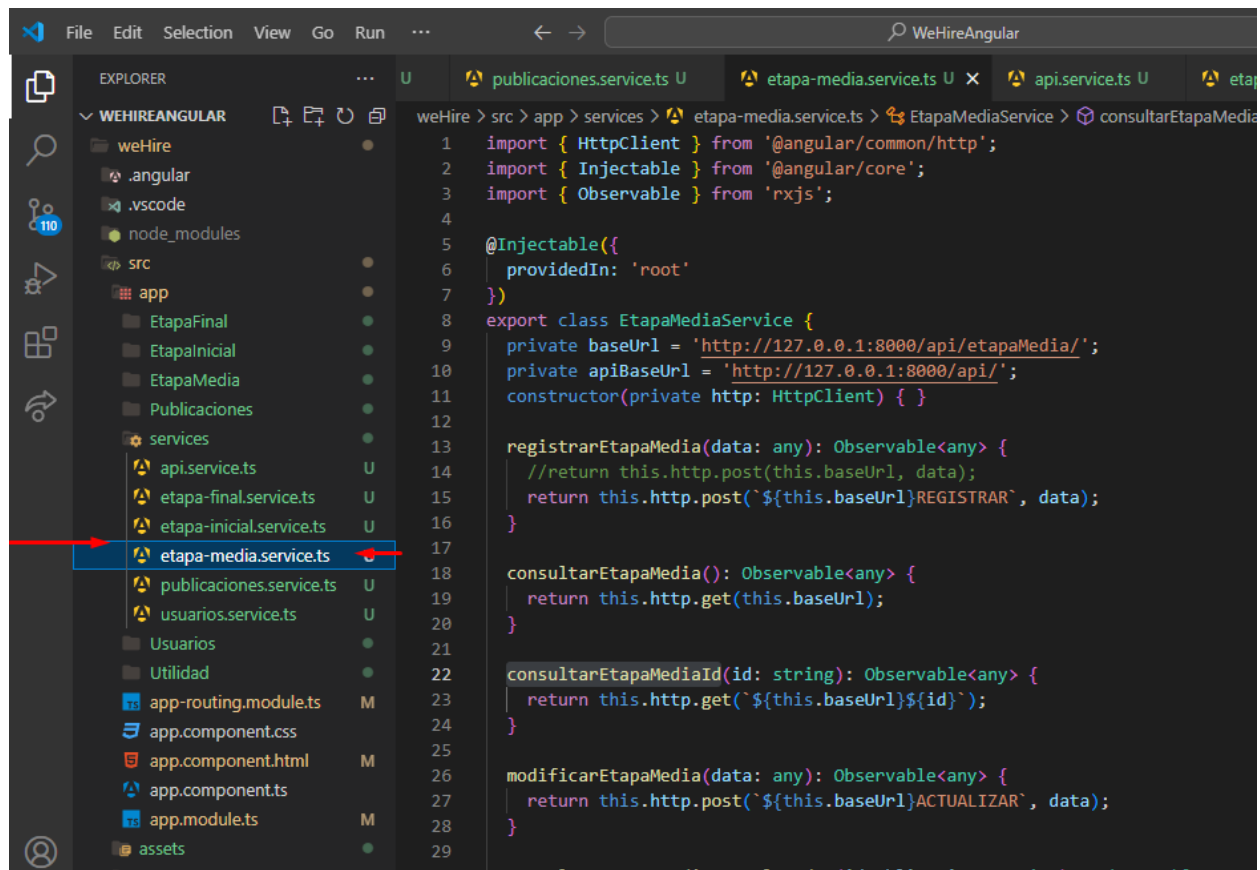


Nota. Servicio de etapa inicial con los métodos correspondientes. Tomado de: Autoría propia

- **Servicio de Etapa Media:** Para gestionar las operaciones asociadas a la **etapa media** del proceso.

Imagen 20

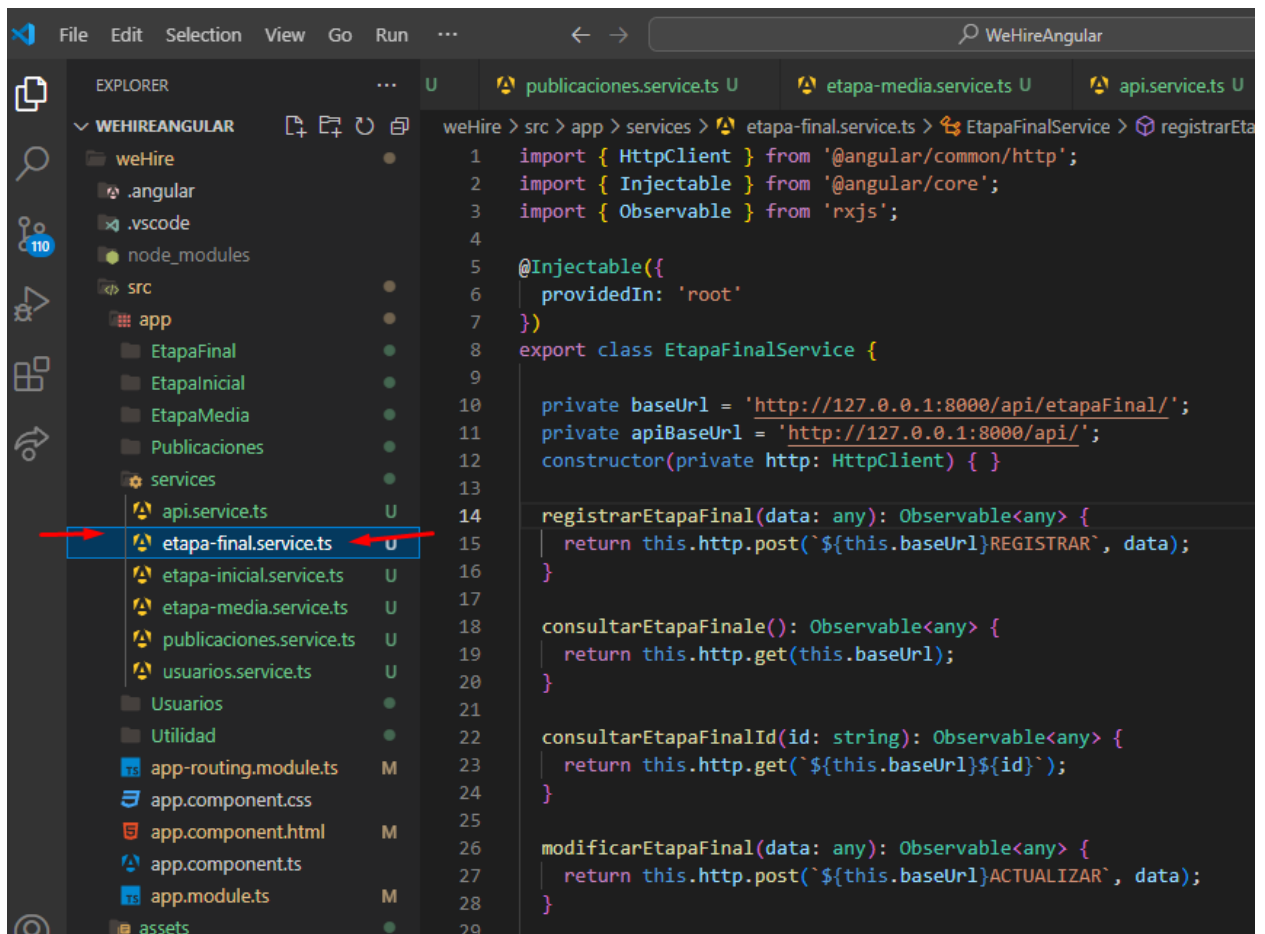
Servicio etapa media



Nota. Servicio de etapa media con los métodos correspondientes. Tomado de: Autoría propia

- **Servicio de Etapa Final:** Para manejar las operaciones relacionadas con la **etapa final** del proceso

Imagen 20
Servicio etapa final

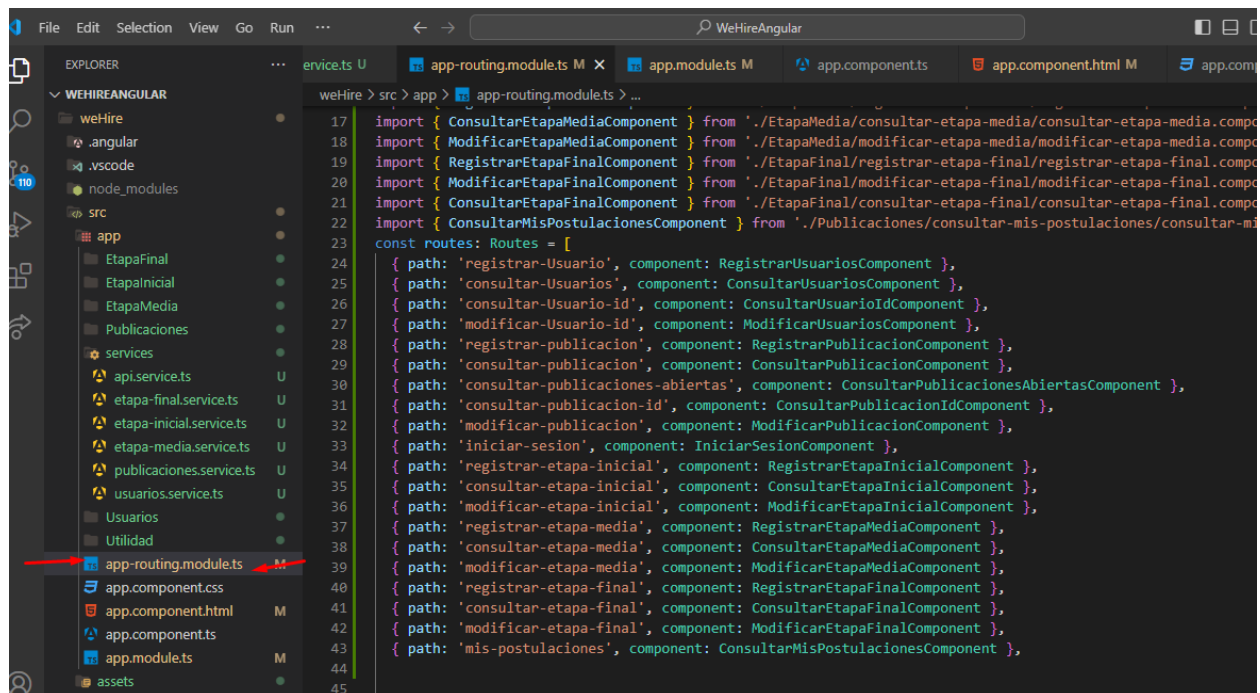


Nota. Servicio de etapa final con los métodos correspondientes. Tomado de: Autoría propia

Para establecer la conexión entre el frontend desarrollado en **Angular** y la API del backend, se configuran las rutas correspondientes en el archivo **app-routing.module.ts**.

Imagen 21

Routing modules



```
17 import { ConsultarEtapaMediaComponent } from './EtapaMedia/consultar-etapa-media/consultar-etapa-media.comp
18 import { ModificarEtapaMediaComponent } from './EtapaMedia/modificar-etapa-media/modificar-etapa-media.comp
19 import { RegistrarEtapaFinalComponent } from './EtapaFinal/regarstrar-etapa-final/regarstrar-etapa-final.comp
20 import { ModificarEtapaFinalComponent } from './EtapaFinal/modificar-etapa-final/modificar-etapa-final.comp
21 import { ConsultarEtapaFinalComponent } from './EtapaFinal/consultar-etapa-final/consultar-etapa-final.comp
22 import { ConsultarMisPostulacionesComponent } from './Publicaciones/consultar-mis-postulaciones/consultar-mi
23 const routes: Routes = [
24   { path: 'registrar-Usuario', component: RegistrarUsuariosComponent },
25   { path: 'consultar-Usuarios', component: ConsultarUsuariosComponent },
26   { path: 'consultar-Usuario-id', component: ConsultarUsuarioIdComponent },
27   { path: 'modificar-Usuario-id', component: ModificarUsuariosComponent },
28   { path: 'registrar-publicacion', component: RegistrarPublicacionComponent },
29   { path: 'consultar-publicacion', component: ConsultarPublicacionComponent },
30   { path: 'consultar-publicaciones-abiertas', component: ConsultarPublicacionesAbiertasComponent },
31   { path: 'consultar-publicacion-id', component: ConsultarPublicacionIdComponent },
32   { path: 'modificar-publicacion', component: ModificarPublicacionComponent },
33   { path: 'iniciar-sesion', component: IniciarSesionComponent },
34   { path: 'registrar-etapa-inicial', component: RegistrarEtapaInicialComponent },
35   { path: 'consultar-etapa-inicial', component: ConsultarEtapaInicialComponent },
36   { path: 'modificar-etapa-inicial', component: ModificarEtapaInicialComponent },
37   { path: 'registrar-etapa-media', component: RegistrarEtapaMediaComponent },
38   { path: 'consultar-etapa-media', component: ConsultarEtapaMediaComponent },
39   { path: 'modificar-etapa-media', component: ModificarEtapaMediaComponent },
40   { path: 'registrar-etapa-final', component: RegistrarEtapaFinalComponent },
41   { path: 'consultar-etapa-final', component: ConsultarEtapaFinalComponent },
42   { path: 'modificar-etapa-final', component: ModificarEtapaFinalComponent },
43   { path: 'mis-postulaciones', component: ConsultarMisPostulacionesComponent },
44
45
```

Nota. Rutas de conexión . Tomado de: Autoría propia

En conclusión, el backend desarrollado en Django y el frontend en Angular 13 trabajan de manera conjunta para ofrecer un sistema modular y escalable. El backend expone las funcionalidades mediante una API, mientras que el frontend organiza las vistas y servicios para consumir estos endpoints.

El código queda almacenado en el siguiente repositorio:
<https://github.com/jenniferQuiceno/WeHire-Proyecto.git>

8. CRONOGRAMA

Con el fin de tener un orden y prioridades durante el desarrollo del proyecto de grado se propone un cronograma de actividades que consiste en un diagrama de Gantt el cual expone de forma general las actividades que se llevarán a cabo para la realización y finalización del proyecto.

Imagen 5
Cronograma de actividades

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Actividades		Agosto				Septiembre				Octubre				Noviembre				Diciembre		
		Sem 1	Sem 2	Sem 3	Sem 4	Sem 1	Sem 2	Sem 3	Sem 4	Sem 1	Sem 2	Sem 3	Sem 4	Sem 1	Sem 2	Sem 3	Sem 4	Sem 1	Sem 2	Sem 3
1	Levantamiento requisitos funcionales																			
2	Levantamiento requisitos no funcionales																			
3	Realización cronograma de actiftidades con inicio y fin																			
4	Creación diseño base de datos																			
5	Escoger lenguaje y herramientas a utilizar en el desarrollo del software																			
6	Creación diseño arquitectura del software con sus integraciones																			
7	Prototipo de pantallas principales																			
8	Preparar entorno de desarrollo																			
9	Implementar los diferentes modulos (pendiente de definir de acuerdo a los requerimientos)																			
10	Realizar pruebas funcionales del desarrollo																			
11	Realizar despliegue del software																			
12	Realizar documentación tecnica																			
13	Realizar manual de usuario																			

Nota. Cronograma de actividades mes a mes. Tomado de: Autoría propia

9. CONCLUSIONES

En conclusión, después de pasar por todas las etapas del desarrollo de software, logramos una página que hace más fácil el proceso de contratación y mejora la comunicación entre reclutadores y candidatos. Desde el levantamiento de requisitos hasta el despliegue, cada paso fue clave para que la herramienta funcione bien y sea fácil de usar. Este proyecto no solo organiza mejor el proceso, sino que también evita confusiones, mantiene a todos informados y hace que contratar sea más rápido y efectivo. En resumen, es una solución que realmente ayuda a las empresas y a los postulantes.

10. BIBLIOGRAFIA

- De La Cueva - Ceo Billin, M. (2024, May 28). Digitalización de procesos administrativos: Beneficios. Blog De Billin. <https://www.billin.net/blog/digitalizacion-procesos-administrativos/>
- Buk. (2022, 8 septiembre). Una comunicación efectiva para los procesos de reclutamiento y selección | Buk. Buk. <https://www.buk.co/blog/comunicacion-efectiva-para-tu-proceso-de-reclutamiento-y-seleccion>
- Eric. (2021, November 18). 9 estrategias para conquistar los mejores talentos a través del reclutamiento digital. Userlike Live Chat. <https://www.userlike.com/es/blog/reclutamiento-digital>
- Cómo hacer más eficiente una contratación. (2017, October 23). Equipos&Talento. <https://www.equiposytalento.com/noticias/2017/10/23/como-hacer-mas-eficiente-una-contratacion>
- Calidad de la contratación: Cómo medirla y consejos para mejorarla. (2023, August 15). Emi Blog. <https://www.emilabs.ai/es-blog/calidad-de-la-contratacion>
- Del Val, D. (2024, August 26). 8 problemas en el reclutamiento y selección de personal. Endalia. <https://www.endalia.com/news/problemas-en-el-reclutamiento-y-seleccion-de-personal/>
- psico-smart.com. (s. f.). ¿Cómo pueden las plataformas de reclutamiento digital simplificar el proceso de contratación en Recursos Humanos? <https://psico-smart.com/articulos/articulo-como-pueden-las-plataformas-de-reclutamiento-digital-simplificar-el-proceso-de-contratacion-en-recursos-humanos-36227>
- Martín, C. (2024, 18 junio). *Plataforma de reclutamiento: tipos, funcionamiento y consejos de uso*. Sesame HR. <https://www.sesamehr.co/blog/reclutamiento-y-seleccion/plataforma-reclutamiento/>