

Lab 05

Tema: Django Admin

Nota

Estudiantes	Escuela	Asignatura
Roger Infa Sanchez, Gary Vilca, Patrick Paredes, Kryzthian Kurt	Carrera Profesional de Ingeniería de Software	Ingeniería Web Semestre: VIII Código: 20231001

EXAMEN PARCIAL	Tema	Duración
01	Django Admin	06 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 24 de octubre 2023	Al 31 de octubre 2023

1. Tarea

- Elabore un primer informe grupal, con el modelo de datos de una aplicación que desarrollará durante este semestre.
- Utilicen todas las recomendaciones encontradas en la aplicación library.
- Acuerdos :
 - Los grupos pueden estar conformado por 1 a 4 integrantes.
 - Sólo se presenta un informe grupal.
 - Sólo se revisa un repositorio. (Avisar al profesor, en el cuál esté el informe grupal).
 - Todos los integrantes deben tener una copia del laboratorio e informe en su repositorio privado también.
 - Todos los integrantes deben pertenecer al mismo grupo de laboratorio y en el mismo horario.
 - El docente preguntará en cualquier momento a cualquier integrante sobre el proyecto, modelo de datos, código fuente, avance.

2. Pregunta

Por cada integrante del equipo, resalte un aprendizaje que adquirió al momento de estudiar esta primera parte de Django (Admin). No se reprima de ser detallista. Coloque su nombre entre parentesis para saber que es su aporte.

3. Entregables

- El informe debe tener un enlace al directorio específico del laboratorio en su repositorio GitHub privado donde esté todo el código fuente y otros que sean necesarios. Evitar la presencia de archivos: binarios, objetos, archivos temporales, cache, librerías, entornos virtuales. Si hay configuraciones particulares puede incluir archivos de especificación como: requirements.txt, o leeme.txt.
- No olvide que el profesor debe ser siempre colaborador a su repositorio (Usuario del profesor @rescobedoq).
- Para ser considerado con la calificación de máxima nota, el informe debe estar elaborado en LATEX
- Usted debe describir sólo los commits más importantes que marcaron hitos en su trabajo, adjuntando capturas de pantalla, del commit, del código fuente, de sus ejecuciones y pruebas.
- En el informe siempre se debe explicar las imágenes (código fuente, capturas de pantalla, commits, ejecuciones, pruebas, etc.) con descripciones puntuales pero precisas.
- Partes de entrega:
 - Modelo de datos. (Diagrama Entidad-Relación)
 - Modelos Python. (En archivos independientes)
 - Implementación del Django Administrador. (CRUD para todas las tablas)
 - Considerar: atributos adecuados, relaciones necesarias, visualización de datos adecuadas, restricciones en el modelo importantes.

4. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/rinfasulasalle/django_project.git`
- Ubicación en Git de mi Repositorio del lab04
- `https://github.com/rinfasulasalle/django_project/tree/main`

5. Resolución

5.1. Modelo de Datos

h

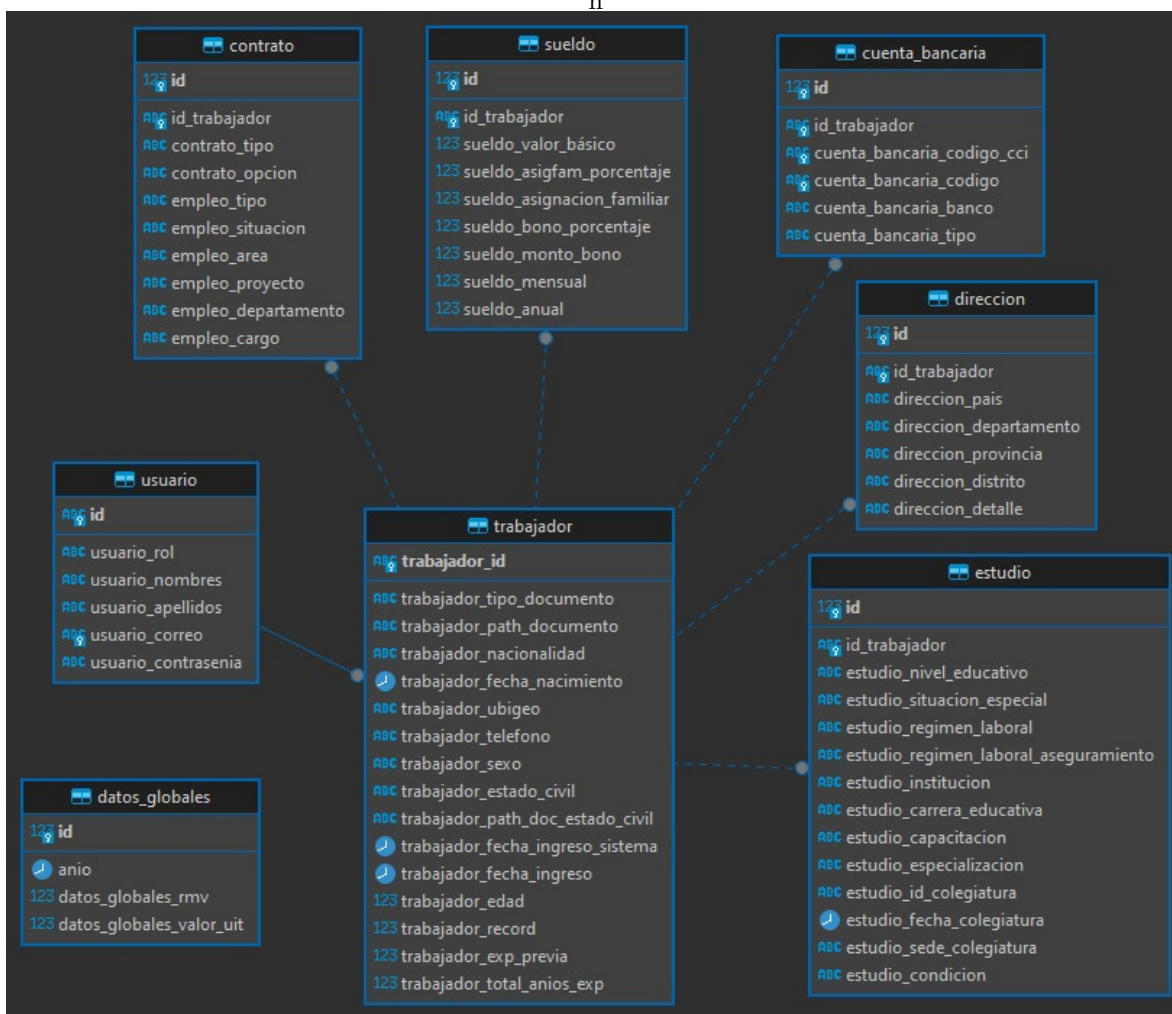


Figura 1: Diagrama Entidad Relación

5.2. Guía para levantar el Proyecto

Primero activamos el entorno virtual con el comando `activate`, luego entramos a nuestro proyecto y ejecutamos un `pip install -r req.txt` para acceder a las librerías que necesitamos en nuestro proyecto.

```
PS C:\Users\Leo\Desktop\iw-23b\lab05\my_env\Scripts> .\activate.bat
PS C:\Users\Leo\Desktop\iw-23b\lab05\my_env\Scripts> .\activate
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab05\my_env\Scripts> cd ..
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab05\my_env> cd ..
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab05> cd ..
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06> cd lab06
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06> cd .\django_project\
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06\django_project> cd .\mysite\
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06\django_project\mysite> pip install -r req.txt
```

Después creamos una migración como sigue, para que se logre observar las relaciones los atributos todo aquello de los modelos:

```
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06\django_project\mysite> python manage.py
migrate
```

Luego usamos el comando `makemigrations` como se observa en seguida:

```
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06\django_project\mysite> python manage.py
makemigrations
```

Finalmente le damos a `runserver` para ver el proyecto, si no hemos recibido una notificación quiere decir que se creo bien el proyecto.

```
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06\django_project\mysite> python manage.py
runserver
```

No olvidar que debemos crear un superusuario como se observa a continuación:

```
(my_env) PS C:\Users\Leo\Desktop\iw-23b\lab06\django_project\mysite> python manage.py
createsuperuser
Username (leave blank to use 'leo'): root
Email address:
Password:
Password (again):
```

Con esto podemos acceder a nuestra ruta admin y ver que ocurre en el proyecto. Como último consejo sería que debemos cambiar el archivo `config` para poder ingresar la configuración de nuestra base de datos.

5.3. Modelos Python

Los modelos se encuentra en nuestra carpeta `gestiontrabajadores` y despues en el archivo `models.py`. Este es el codigo del modelo Usuario:

```
class Usuario(models.Model):
    id = models.CharField(primary_key=True, max_length=20, unique=True)
    usuario_rol = models.CharField(max_length=20, choices=[...], default='Sin acceso')
    usuario_nombres = models.CharField(max_length=100)
    usuario_apellidos = models.CharField(max_length=100)
    usuario_correo = models.EmailField(max_length=100, unique=True)
```

```
usuario_contrasenia = models.CharField(max_length=50)
usuario_sexo = models.CharField(max_length=20, choices=[...], default='No Especificado')
usuario_telefono = models.CharField(max_length=50)

def __str__(self):
    return self.id
```

Listing 1: Modelo Usuario

Aquí colocamos los atributos que posee usuario como el nombre , apellidos , correo, contrasenia, sexo y telefono.

```
class Trabajador(models.Model):
    trabajador_id = models.CharField(primary_key=True, max_length=20, unique=True)
    trabajador_fecha_nacimiento = models.DateField()
    trabajador_tipo_documento = models.CharField(max_length=50)
    trabajador_path_documento = models.CharField(max_length=255, default='PATH/noNe')
    trabajador_nacionalidad = models.CharField(max_length=50, default='No Especificado')
    trabajador_ubigeo = models.CharField(max_length=255, default='No Especificado')
    trabajador_estado_civil = models.CharField(max_length=20, choices=[...], default='No Especificado')
    trabajador_path_doc_estado_civil = models.CharField(max_length=255, default='PATH/noNe')
    trabajador_fecha_ingreso = models.DateField()
    trabajador_fecha_ingreso_sistema = models.DateField()
    trabajador_edad = models.IntegerField()
    trabajador_record = models.DecimalField(max_digits=20, decimal_places=2)
    trabajador_exp_previa = models.DecimalField(max_digits=20, decimal_places=2)
    trabajador_total_anios_exp = models.DecimalField(max_digits=20, decimal_places=2)

    usuario = models.OneToOneField(Usuario, on_delete=models.CASCADE)

    def __str__(self):
        return self.trabajador_id
```

Listing 2: Modelo Trabajador

En este modelo se usa el id, una fecha de nacimiento, tipo de documento, la ruta del documento, la nacionalidad, el ubigeo, el estado civil, la ruta de su documento, su fecha de ingreso, su edad, el record, su experiencia previa y el total de años además establece una relación con la tabla usuarios.

```
class Sueldo(models.Model):
    id_trabajador = models.OneToOneField(Trabajador, on_delete=models.CASCADE,
    primary_key=True)
    sueldo_valor_basico = models.DecimalField(max_digits=20, decimal_places=2)
    sueldo_asigfam_porcentaje = models.DecimalField(max_digits=20, decimal_places=2)
    sueldo_asignacion_familiar = models.DecimalField(max_digits=20, decimal_places=2)
    sueldo_bono_porcentaje = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    sueldo_monto_bono = models.DecimalField(max_digits=20, decimal_places=2)
    sueldo_mensual = models.DecimalField(max_digits=20, decimal_places=2)
    sueldo_anual = models.DecimalField(max_digits=20, decimal_places=2)

    def __str__(self):
        return str(self.id_trabajador)
```

Listing 3: Modelo Sueldo

El modelo de sueldo tiene un sueldo básico, una asignación en porcentaje, un bono, un monto del bono,

un sueldo mensual y sueldo anual. Finalmente una relación con la tabla trabajador.

```
class Contrato(models.Model):
    id_trabajador = models.OneToOneField(Trabajador, on_delete=models.CASCADE,
    primary_key=True)
    contrato_tipo = models.CharField(max_length=255)
    contrato_opcion = models.CharField(max_length=255)
    empleo_tipo = models.CharField(max_length=50)
    empleo_situacion = models.CharField(max_length=50)
    empleo_area = models.CharField(max_length=50)
    empleo_proyecto = models.CharField(max_length=50)
    empleo_departamento = models.CharField(max_length=50)
    empleo_cargo = models.CharField(max_length=50)

    def __str__(self):
        return str(self.id_trabajador)
```

Listing 4: Modelo Contrato

El modelo contrato tiene atributos como: tipo, opción, empleo tipo, empleo situación, empleo área, empleo proyecto, empleo departamento, empleo cargo. Esta vinculado con el modelo trabajador

```
class CuentaBancaria(models.Model):
    id_trabajador = models.OneToOneField(Trabajador, on_delete=models.CASCADE,
    primary_key=True)
    cuenta_bancaria_codigo_cci = models.CharField(max_length=255, unique=True)
    cuenta_bancaria_codigo = models.CharField(max_length=255, unique=True)
    cuenta_bancaria_banco = models.CharField(max_length=255)
    cuenta_bancaria_tipo = models.CharField(max_length=20, choices=[...])

    def __str__(self):
        return str(self.id_trabajador)
```

Listing 5: Modelo Cuenta Bancaria

El modelo cuenta Bancaria tiene atributos como: código bancario código cci, cuenta bancaria código, cuenta bancaria banco, cuenta bancaria tipo. Además cuenta con una relación con el modelo trabajador.

```
class Direccion(models.Model):
    id_trabajador = models.OneToOneField(Trabajador, on_delete=models.CASCADE,
    primary_key=True)
    direccion_pais = models.CharField(max_length=255)
    direccion_departamento = models.CharField(max_length=255)
    direccion_provincia = models.CharField(max_length=255)
    direccion_distrito = models.CharField(max_length=255)
    direccion_detalle = models.CharField(max_length=255)

    def __str__(self):
        return str(self.id_trabajador)
```

Listing 6: Modelo Dirección

El modelo dirección tiene atributos como: dirección país, dirección departamento, dirección provincia, dirección distrito, dirección detalle. Además se encuentra vinculado a la tabla trabajador.

```
class Estudio(models.Model):
    id_trabajador = models.OneToOneField(Trabajador, on_delete=models.CASCADE,
    primary_key=True)
    estudio_nivel_educativo = models.CharField(max_length=255)
    estudio_situacion_especial = models.CharField(max_length=255, default='Situación especial
    no especificada')
    estudio_regimen_laboral = models.CharField(max_length=255, default='Regimen laboral no
    especificado')
    estudio_regimen_laboral_aseguramiento = models.CharField(max_length=255,
    default='Aseguramiento no especificado')
    estudio_institucion = models.CharField(max_length=255, default='Institución no
    especificada')
    estudio_carrera_educativa = models.CharField(max_length=255, default='Carrera no
    especificada')
    estudio_capacitacion = models.CharField(max_length=255, default='Capacitación no
    especificada')
    estudio_especializacion = models.CharField(max_length=255, default='Especialización no
    especificada')
    estudio_id_colegiatura = models.CharField(max_length=255, default='No especificado')
    estudio_fecha_colegiatura = models.DateField(default='1212-12-12')
    estudio_sede_colegiatura = models.CharField(max_length=255, default='Sede colegiatura no
    especificada')
    estudio_condicion = models.CharField(max_length=20, choices=[...], default='No
    especificado')

    def __str__(self):
        return str(self.id_trabajador)
```

Listing 7: Modelo Estudio

Contiene los atributos nivel de estudios educativos, nivel situación especial, estudio regimen laboral, estudio regimen laboral aseguramiento, estudio institución, estudio carrera educativa, estudio capacitación, estudio especialización, id colegiatura, estudio sede colegiatura y la condición del estudio.

5.4. Creación de las operaciones de las tablas

5.4.1. Tabla Trabajador

El código que se ha proporcionado es una migración en Django, un popular framework de desarrollo web en Python. La migración está creando un modelo llamado "Trabajador" en la base de datos con una serie de campos y atributos. Este modelo representa la información sobre los trabajadores, como su identificación, fecha de nacimiento, nacionalidad, estado civil, experiencia laboral previa, etc.

```
operations = [
    migrations.CreateModel(
        name="Trabajador",
        fields=[
            (
                "trabajador_id",
                models.CharField(
                    max_length=20, primary_key=True, serialize=False, unique=True
                ),
            ),
            ("trabajador_fecha_nacimiento", models.DateField()),
            ("trabajador_tipo_documento", models.CharField(max_length=50)),
            (
```

```
        "trabajador_path_documento",
        models.CharField(default="PATH/noNe", max_length=255),
    ),
    (
        "trabajador_nacionalidad",
        models.CharField(default="No Especificado", max_length=50),
    ),
    (
        "trabajador_ubigeo",
        models.CharField(default="No Especificado", max_length=255),
    ),
    (
        "trabajador_estado_civil",
        models.CharField(
            choices=[
                ("Soltero", "Soltero"),
                ("Casado", "Casado"),
                ("Viudo", "Viudo"),
                ("Divorciado", "Divorciado"),
                ("Conviviente", "Conviviente"),
                ("No Especificado", "No Especificado"),
            ],
            default="No Especificado",
            max_length=20,
        ),
    ),
    (
        "trabajador_path_doc_estado_civil",
        models.CharField(default="PATH/noNe", max_length=255),
    ),
    ("trabajador_fecha_ingreso", models.DateField()),
    ("trabajador_fecha_ingreso_sistema", models.DateField()),
    ("trabajador_edad", models.IntegerField()),
    (
        "trabajador_record",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    (
        "trabajador_exp_previa",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    (
        "trabajador_total_anios_exp",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
],
),
```

Listing 8: Modelo Trabajador

A continuación, se detallará cuál es el funcionamiento del código:

- `trabajador.id`: Es un campo de texto que sirve como clave primaria y es único para cada trabajador.
- `trabajador.fecha_nacimiento`: Almacena la fecha de nacimiento del trabajador.

- trabajador_estado_civil: Es un campo de elección que permite seleccionar el estado civil del trabajador.
- trabajador_record, trabajador_exp_previa, y trabajador_total_anios_exp: Campos numéricos que almacenan información sobre el registro laboral y experiencia previa del trabajador.
- Hay otros campos que almacenan información personal y laboral del trabajador.

5.4.2. Tabla Usuario

El código que se ha proporcionado también es una definición de migración en Django y crea un modelo llamado "Usuario". El código está creando un modelo de base de datos en Django llamado "Usuario". Este modelo representa información sobre los usuarios de la aplicación de gestión de trabajadores para una empresa. A continuación, se muestra el código respectivo:

```
migrations.CreateModel(  
    name="Usuario",  
    fields=[  
        (  
            "id",  
            models.CharField(  
                max_length=20, primary_key=True, serialize=False, unique=True  
            ),  
        ),  
        (  
            "usuario_rol",  
            models.CharField(  
                choices=[  
                    ("Administrador", "Administrador"),  
                    ("Recursos Humanos", "Recursos Humanos"),  
                    ("Trabajador", "Trabajador"),  
                    ("Sin acceso", "Sin acceso"),  
                ],  
                default="Sin acceso",  
                max_length=20,  
            ),  
        ),  
        ("usuario_nombres", models.CharField(max_length=100)),  
        ("usuario_apellidos", models.CharField(max_length=100)),  
        ("usuario_correo", models.EmailField(max_length=100, unique=True)),  
        ("usuario_contrasenia", models.CharField(max_length=50)),  
        (  
            "usuario_sexo",  
            models.CharField(  
                choices=[  
                    ("Masculino", "Masculino"),  
                    ("Femenino", "Femenino"),  
                    ("No Especificado", "No Especificado"),  
                ],  
                default="No Especificado",  
                max_length=20,  
            ),  
        ),  
        ("usuario_telefono", models.CharField(max_length=50)),  
    ],  
)
```

Listing 9: Modelo Usuario

A continuación, se detallará cuál es el funcionamiento del código:

- **id:** Es un campo de texto que sirve como clave primaria y es único para cada usuario. Sin embargo, suele ser común utilizar un campo automático de ID en lugar de un campo de texto para la clave primaria.
- **usuario_rol:** Es un campo de elección que permite seleccionar el rol del usuario (Administrador, Recursos Humanos, Trabajador, Sin acceso).
- **usuario_nombres y usuario_apeellidos :** *Almacena el nombre y apellido del usuario.*
- **usuario_correo:** Almacena la dirección de correo electrónico del usuario y tiene la restricción de ser único.
- **usuario_contrasenia:** Almacena la contraseña del usuario.
- **usuario_sexo:** Es un campo de elección que permite seleccionar el sexo del usuario (Masculino, Femenino, No Especificado).
- **usuario_telefono:** Almacena el número de teléfono del usuario.

5.4.3. Tabla Contrato

El código que se ha proporcionado es otra definición de migración en Django que crea un modelo llamado "Contrato". El fragmento del código está creando un modelo de base de datos en Django llamado "Contrato". Este modelo parece representar información sobre los contratos de trabajo y los empleos de los trabajadores. El código es el siguiente:

```
migrations.CreateModel(  
    name="Contrato",  
    fields=[  
        (  
            "id_trabajador",  
            models.OneToOneField(  
                on_delete=django.db.models.deletion.CASCADE,  
                primary_key=True,  
                serialize=False,  
                to="gestion_trabajadores.trabajador",  
            ),  
        ),  
        ("contrato_tipo", models.CharField(max_length=255)),  
        ("contrato_opcion", models.CharField(max_length=255)),  
        ("empleo_tipo", models.CharField(max_length=50)),  
        ("empleo_situacion", models.CharField(max_length=50)),  
        ("empleo_area", models.CharField(max_length=50)),  
        ("empleo_proyecto", models.CharField(max_length=50)),  
        ("empleo_departamento", models.CharField(max_length=50)),  
        ("empleo_cargo", models.CharField(max_length=50)),  
    ],  
)
```

Listing 10: Modelo Contrato

A continuación, se detallará cuál es el funcionamiento del código:

- `id.trabajador`: Es un campo que se define como una clave primaria (`primary_key=True`) y como una clave foránea (`models.OneToOneField`) que se relaciona con el modelo "Trabajador" de la aplicación "gestion_trabajadores". Esto establece una relación uno a uno entre el contrato y el trabajador al que se aplica el contrato.
- `contrato_tipo`, `contrato_opcion`, `empleo_tipo`, `empleo_situacion`, `empleo_area`, `empleo_proyecto`, `empleo_departamento` y `empleo_cargo`: Estos campos almacenan información relacionada con el tipo de contrato, opciones de contrato y detalles del empleo, como el tipo de empleo, la situación laboral, el área, el proyecto, el departamento y el cargo del trabajador.

5.4.4. Tabla Cuenta Bancaria

El código que se ha proporcionado es otra definición de migración en Django que crea un modelo llamado "CuentaBancaria". El fragmento del código está creando un modelo de base de datos en Django llamado "CuentaBancaria". Este modelo representa información sobre las cuentas bancarias de los trabajadores. Algunos aspectos clave del modelo son:

```
migrations.CreateModel(
    name="CuentaBancaria",
    fields=[
        (
            "id_trabajador",
            models.OneToOneField(
                on_delete=django.db.models.deletion.CASCADE,
                primary_key=True,
                serialize=False,
                to="gestion_trabajadores.trabajador",
            ),
        ),
        (
            "cuenta_bancaria_codigo_cci",
            models.CharField(max_length=255, unique=True),
        ),
        (
            "cuenta_bancaria_codigo",
            models.CharField(max_length=255, unique=True),
        ),
        ("cuenta_bancaria_banco", models.CharField(max_length=255)),
        (
            "cuenta_bancaria_tipo",
            models.CharField(
                choices=[("Sueldo", "Sueldo"), ("CTS", "CTS")], max_length=20
            ),
        ),
    ],
)
```

Listing 11: Modelo Cuenta Bancaria

A continuación, se detallará cuál es el funcionamiento del código:

- `id.trabajador`: Al igual que en el modelo anterior, este campo se define como una clave primaria (`primary_key=True`) y como una clave foránea (`models.OneToOneField`) que se relaciona con el modelo "Trabajador" de la aplicación "gestion_trabajadores". Esto establece una relación uno a uno entre la cuenta bancaria y el trabajador al que pertenece.

- `cuenta_bancaria_codigo_cci` y `cuenta_bancaria_codigo`: Estos campos almacenan códigos únicos relacionados con la cuenta bancaria.
- `cuenta_bancaria_banco`: Almacena el nombre del banco asociado a la cuenta bancaria.
- `cuenta_bancaria_tipo`: Es un campo de elección que permite seleccionar el tipo de cuenta bancaria (Sueldo o CTS).

5.4.5. Tabla Direccion

El código que se ha proporcionado es otra definición de migración en Django que crea un modelo llamado "Direccion". El código está creando un modelo de base de datos en Django llamado "Direccion". Este modelo representa información sobre las direcciones de los trabajadores. Se muestra el código:

```
migrations.CreateModel(  
    name="Direccion",  
    fields=[  
        (  
            "id_trabajador",  
            models.OneToOneField(  
                on_delete=django.db.models.deletion.CASCADE,  
                primary_key=True,  
                serialize=False,  
                to="gestion_trabajadores.trabajador",  
            ),  
        ),  
        ("direccion_pais", models.CharField(max_length=255)),  
        ("direccion_departamento", models.CharField(max_length=255)),  
        ("direccion_provincia", models.CharField(max_length=255)),  
        ("direccion_distrito", models.CharField(max_length=255)),  
        ("direccion_detalle", models.CharField(max_length=255)),  
    ],  
)
```

Listing 12: Modelo Direccion

A continuación, se detallará cuál es el funcionamiento del código:

- `id_trabajador`: Al igual que en los modelos anteriores, este campo se define como una clave primaria (`primary_key=True`) y como una clave foránea (`models.OneToOneField`) que se relaciona con el modelo "Trabajador" de la aplicación "gestion_trabajadores". Esto establece una relación uno a uno entre la dirección y el trabajador al que pertenece.
- `_pais`, `direccion_departamento`, `direccion_provincia`, `direccion_distrito` y `direccion_detalle`: Estos campos almacenan información detallada sobre la dirección del trabajador, como el país, el departamento, la provincia, el distrito y detalles adicionales.

5.4.6. Tabla Estudio

El código que se ha proporcionado es otra definición de migración en Django que crea un modelo llamado "Estudio". El código está creando un modelo de base de datos en Django llamado "Estudio". Este modelo representa información sobre la educación y los estudios realizados por los trabajadores. La implementación es la siguiente:

```
migrations.CreateModel(  
    name="Estudio",
```

```
fields=[
    (
        "id_trabajador",
        models.OneToOneField(
            on_delete=django.db.models.deletion.CASCADE,
            primary_key=True,
            serialize=False,
            to="gestion_trabajadores.trabajador",
        ),
    ),
    ("estudio_nivel_educativo", models.CharField(max_length=255)),
    (
        "estudio_situacion_especial",
        models.CharField(
            default="Situacin especial no especificada", max_length=255
        ),
    ),
    (
        "estudio_regimen_laboral",
        models.CharField(
            default="Rgimen laboral no especificado", max_length=255
        ),
    ),
    (
        "estudio_regimen_laboral_aseguramiento",
        models.CharField(
            default="Aseguramiento no especificado", max_length=255
        ),
    ),
    (
        "estudio_institucion",
        models.CharField(
            default="Institucin no especificada", max_length=255
        ),
    ),
    (
        "estudio_carrera_educativa",
        models.CharField(default="Carrera no especificada", max_length=255),
    ),
    (
        "estudio_capacitacion",
        models.CharField(
            default="Capacitacin no especificada", max_length=255
        ),
    ),
    (
        "estudio_especializacion",
        models.CharField(
            default="Especializacin no especificada", max_length=255
        ),
    ),
    (
        "estudio_id_colegiatura",
        models.CharField(default="No especificado", max_length=255),
    ),
    ("estudio_fecha_colegiatura", models.DateField(default="1212-12-12")),
]
```

```
(
    "estudio_sede_colegiatura",
    models.CharField(
        default="Sede colegiatura no especificada", max_length=255
    ),
),
(
    "estudio_condicion",
    models.CharField(
        choices=[
            ("Habilitado", "Habilitado"),
            ("No habilitado", "No habilitado"),
            ("No especificado", "No especificado"),
        ],
        default="No especificado",
        max_length=20,
    ),
),
],
),
```

Listing 13: Modelo Estudio

A continuación, se detallará cuál es el funcionamiento del código:

- **id_trabajador:** Al igual que en los modelos anteriores, este campo se define como una clave primaria (`primary_key=True`) y como una clave foránea (`models.OneToOneField`) que se relaciona con el modelo "Trabajador" de la aplicación "gestion_trabajadores". Esto establece una relación uno a uno entre los estudios y el trabajador al que pertenecen.
- **estudio_nivel_educativo:** Almacena el nivel educativo del trabajador.
- Varios campos como `estudio_situacion_especial`, `estudio_regimen_laboral`, `estudio_institucion`, `estudio_carrera_educativa`, `estudio_capacitacion`, `estudio_especializacion`, `estudio_id_colegiatura`, `estudio_fecha_colegiatura`, `estudio_sede_colegiatura`, y `estudio_condicion` almacenan información relacionada con la situación educativa y profesional del trabajador. Estos campos tienen valores predeterminados para casos en los que la información no esté disponible.
- **estudio_condicion:** Es un campo de elección que permite seleccionar la condición del trabajador en relación con su educación (Habilitado, No habilitado, No especificado).

5.4.7. Tabla Sueldo

El código que se ha proporcionado es otra definición de migración en Django que crea un modelo llamado "Sueldo". El código está creando un modelo de base de datos en Django llamado "Sueldo". Este modelo representa información sobre los sueldos y remuneraciones de los trabajadores.

```
migrations.CreateModel(
    name="Sueldo",
    fields=[
        (
            "id_trabajador",
            models.OneToOneField(
                on_delete=django.db.models.deletion.CASCADE,
                primary_key=True,
                serialize=False,
                to="gestion_trabajadores.trabajador",
            ),
        ),
    ],
)
```

```

    ),
    (
        "sueldo_valor_basico",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    (
        "sueldo_asigfam_porcentaje",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    (
        "sueldo_asignacion_familiar",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    (
        "sueldo_bono_porcentaje",
        models.DecimalField(decimal_places=2, default=0, max_digits=20),
    ),
    (
        "sueldo_monto_bono",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    (
        "sueldo_mensual",
        models.DecimalField(decimal_places=2, max_digits=20),
    ),
    ("sueldo_anual", models.DecimalField(decimal_places=2, max_digits=20)),
1,
),

```

Listing 14: Modelo Sueldo

5.5. Ejecución

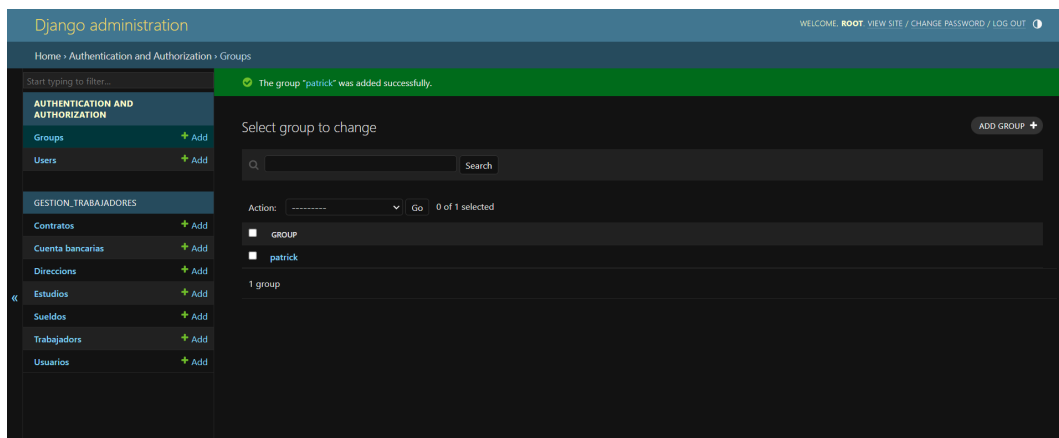


Figura 2: Vista principal del Proyecto

Aquí se observa como se guardan los grupos del proyecto

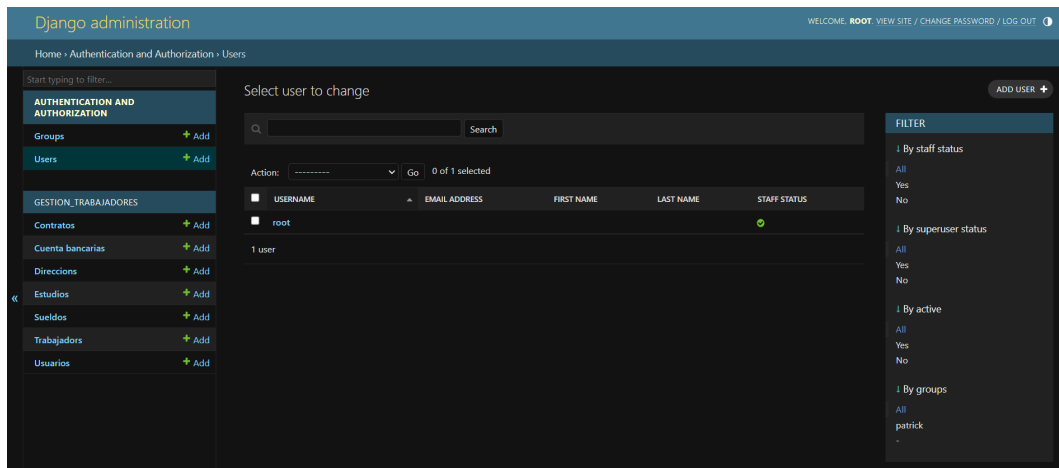


Figura 3: Vista del usuario

Aquí muestra los usuarios guardados y además el que se encuentra por defecto

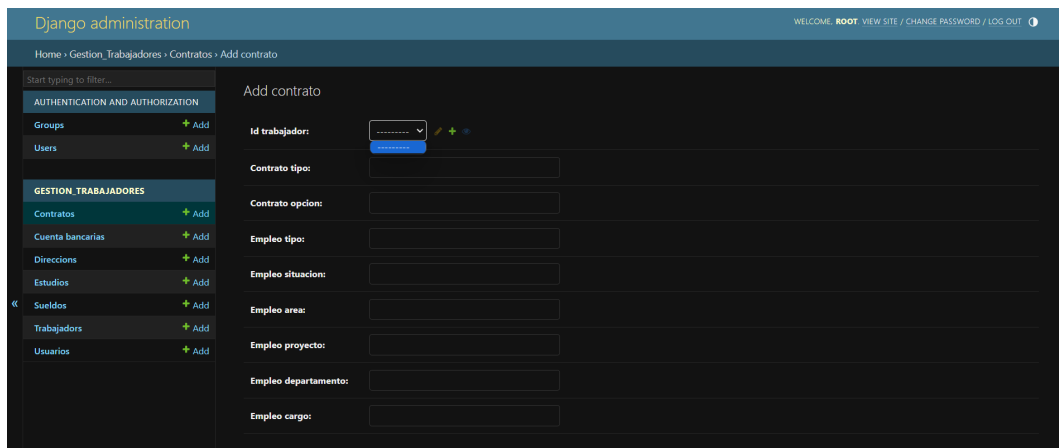
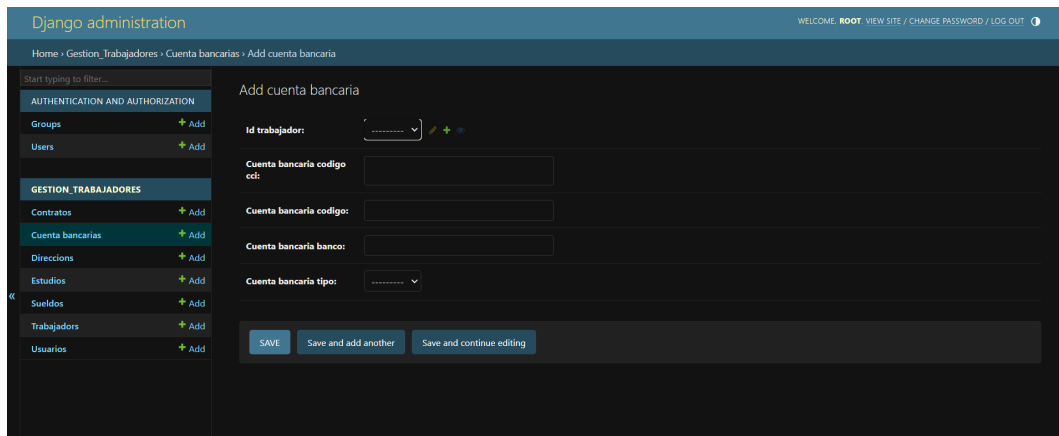


Figura 4: Vista contratos

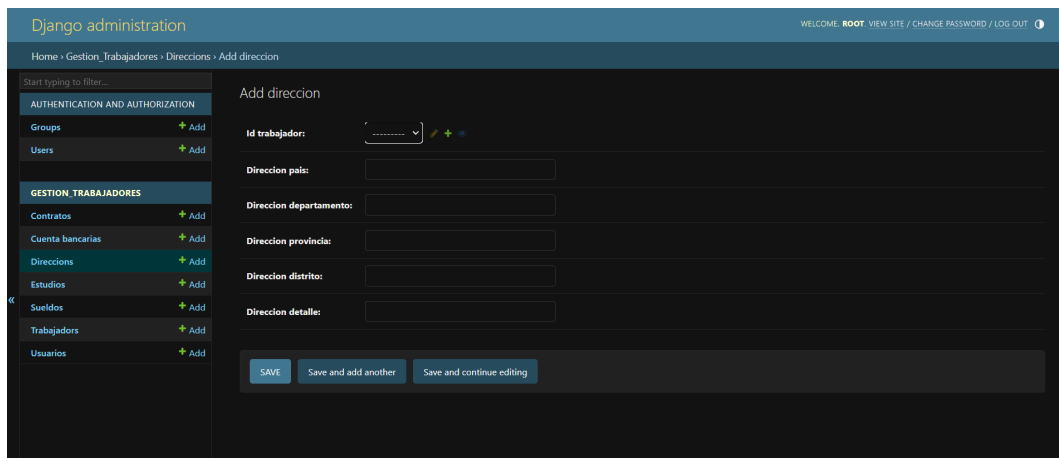
Esta es la vista de contratos en el boton de agregar contrato te muestra todos los atributos de la tabla contratos



The screenshot shows the Django administration interface for the 'Add cuenta bancaria' form. The left sidebar contains a navigation menu with sections: 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'GESTION TRABAJADORES' (Contratos, Cuenta bancarias, Direcciones, Estudios, Sueldos, Trabajadores, Usuarios). The main content area is titled 'Add cuenta bancaria' and includes a breadcrumb trail: 'Home > Gestion_Trabajadores > Cuenta bancarias > Add cuenta bancaria'. The form fields are: 'Id trabajador:' (a dropdown menu), 'Cuenta bancaria codigo ccl:' (a text input), 'Cuenta bancaria codigo:' (a text input), 'Cuenta bancaria banco:' (a text input), and 'Cuenta bancaria tipo:' (a dropdown menu). At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Figura 5: Vista Cuenta bancaria

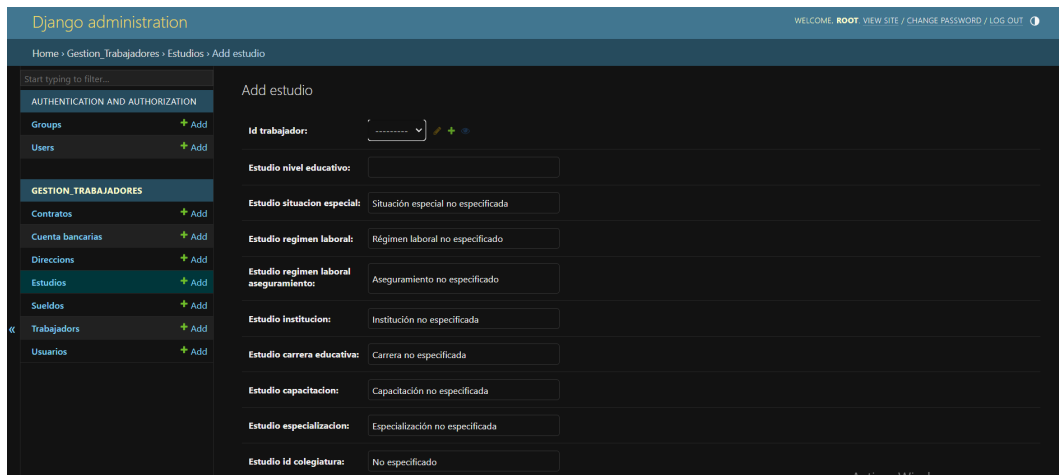
Nos pide el id del trabajador y después completar con lo que corresponde a los atributos de esta tabla



The screenshot shows the Django administration interface for the 'Add direccion' form. The left sidebar is identical to the previous figure. The main content area is titled 'Add direccion' and includes a breadcrumb trail: 'Home > Gestion_Trabajadores > Direcciones > Add direccion'. The form fields are: 'Id trabajador:' (a dropdown menu), 'Direccion pais:' (a text input), 'Direccion departamento:' (a text input), 'Direccion provincia:' (a text input), 'Direccion distrito:' (a text input), and 'Direccion detalle:' (a text input). At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Figura 6: Vista Direcciones

También nos pide el id del trabajador y demás datos para acceder a sus direcciones.

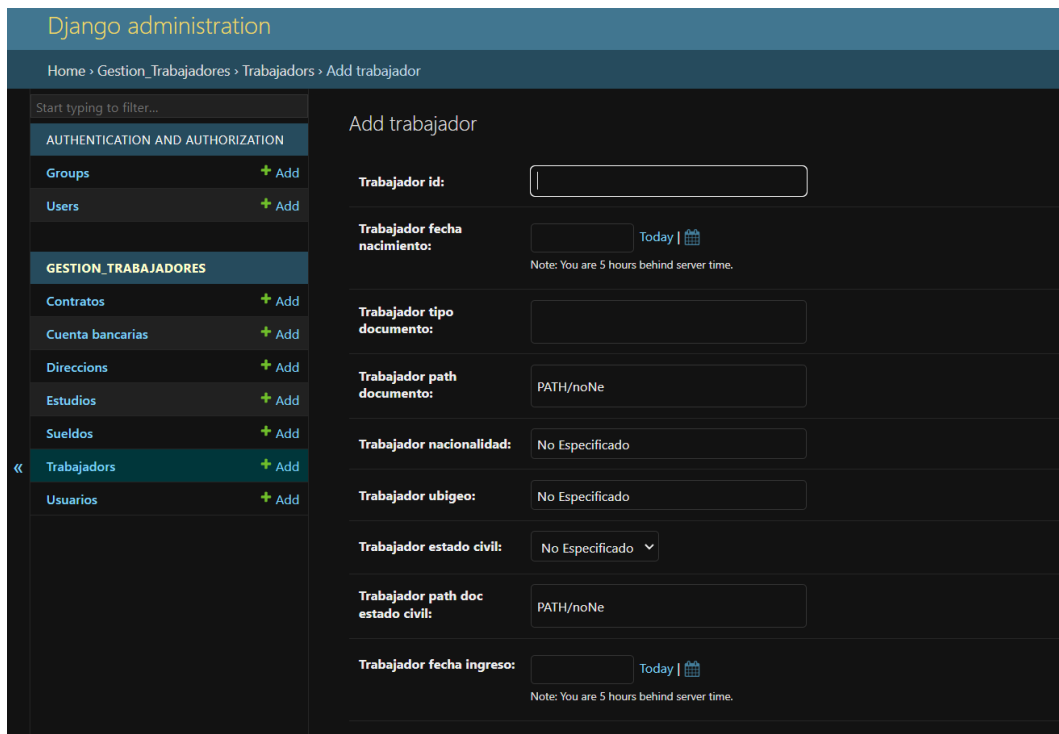


The screenshot shows the 'Add estudio' form in the Django administration interface. The left sidebar contains a menu with 'Estudios' highlighted. The main form area has the following fields:

- Id trabajador:** A dropdown menu.
- Estudio nivel educativo:** A text input field.
- Estudio situacion especial:** A text input field with the value 'Situación especial no especificada'.
- Estudio regimen laboral:** A text input field with the value 'Régimen laboral no especificado'.
- Estudio regimen laboral aseguramiento:** A text input field with the value 'Aseguramiento no especificado'.
- Estudio institucion:** A text input field with the value 'Institución no especificada'.
- Estudio carrera educativa:** A text input field with the value 'Carrera no especificada'.
- Estudio capacitacion:** A text input field with the value 'Capacitación no especificada'.
- Estudio especializacion:** A text input field with the value 'Especialización no especificada'.
- Estudio id colegiatura:** A text input field with the value 'No especificado'.

Figura 7: Vista Estudios

Aquí nos pide el Id del trabajador y demás datos del modelo estudios

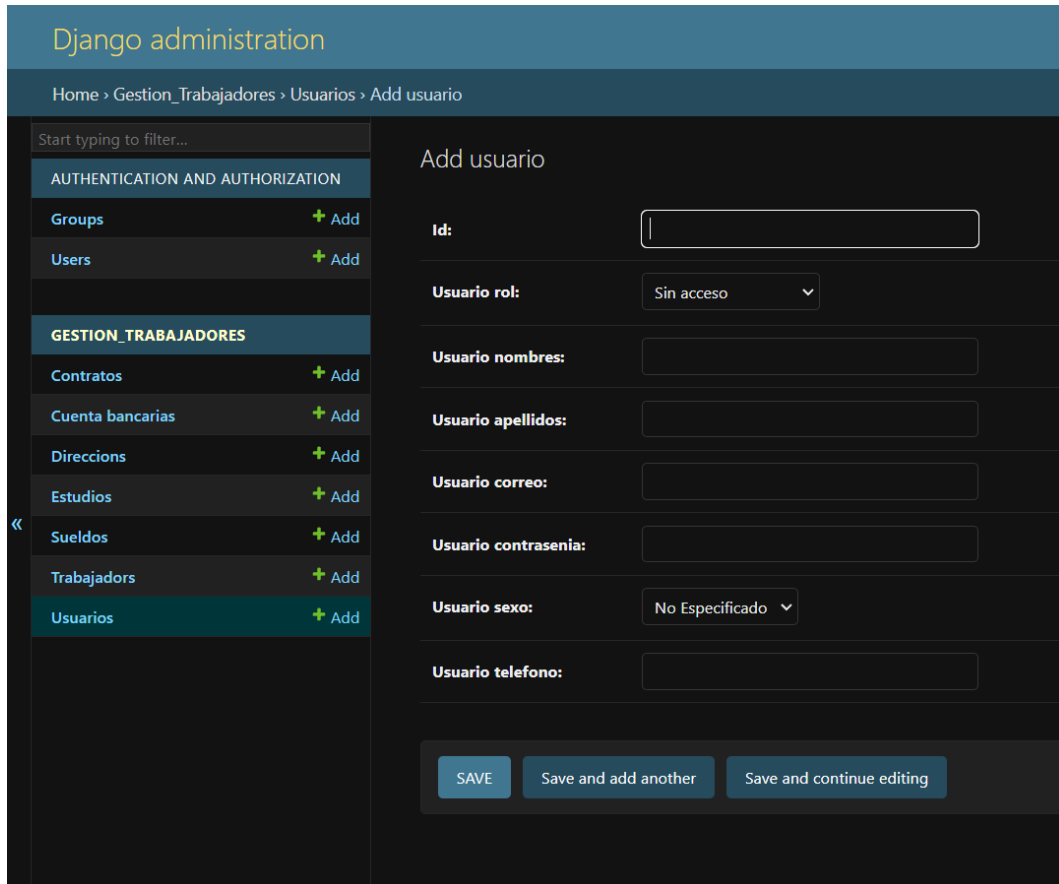


The screenshot shows the 'Add trabajador' form in the Django administration interface. The left sidebar contains a menu with 'Trabajadores' highlighted. The main form area has the following fields:

- Trabajador id:** A text input field.
- Trabajador fecha nacimiento:** A date input field with a calendar icon and the text 'Today'. Below it, a note says 'Note: You are 5 hours behind server time.'
- Trabajador tipo documento:** A text input field.
- Trabajador path documento:** A text input field with the value 'PATH/noNe'.
- Trabajador nacionalidad:** A text input field with the value 'No Especificado'.
- Trabajador ubigeo:** A text input field with the value 'No Especificado'.
- Trabajador estado civil:** A dropdown menu with the value 'No Especificado'.
- Trabajador path doc estado civil:** A text input field with the value 'PATH/noNe'.
- Trabajador fecha ingreso:** A date input field with a calendar icon and the text 'Today'. Below it, a note says 'Note: You are 5 hours behind server time.'

Figura 8: Vista Sueldos

Aquí nos pide también el id del trabajador y demás datos de la tabla sueldos.



The image shows a screenshot of the Django administration interface for adding a new user. The page title is "Django administration". The breadcrumb trail is "Home > Gestion_Trabajadores > Usuarios > Add usuario". The left sidebar contains a search bar and two main sections: "AUTHENTICATION AND AUTHORIZATION" with links for "Groups" and "Users", and "GESTION_TRABAJADORES" with links for "Contratos", "Cuenta bancarias", "Direccions", "Estudios", "Sueldos", "Trabajadors", and "Usuarios" (which is highlighted). The main content area is titled "Add usuario" and contains a form with the following fields: "Id:" (text input), "Usuario rol:" (dropdown menu with "Sin acceso" selected), "Usuario nombres:" (text input), "Usuario apellidos:" (text input), "Usuario correo:" (text input), "Usuario contrasenia:" (text input), "Usuario sexo:" (dropdown menu with "No Especificado" selected), and "Usuario telefono:" (text input). At the bottom of the form are three buttons: "SAVE", "Save and add another", and "Save and continue editing".

Figura 9: Vista Usuarios

Aquí nos pide un id que será su dni y demás datos del usuario.

6. Calificación

Tabla 1: Rúbrica para contenido del Informe y Exposición

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1	Uso del disfraz temático: 31 de Octubre (canción criolla o Halloween)	6	X	6	
2	Expositor empieza oportunamente. Hora pactada o por corresponder correlativo.	3	X	3	
3	El expositor termina toda la exposicion, incluido las preguntas al tiempo asignado (20 min),) puede ser un poco antes, pero no muy básico.	3	X	3	
4	El expositor tiene un dominio del tema usando de forma efectiva los recursos utilizados.	5	X	5	
7	El expositor resuelve la pregunta formulada durante la exposición.	3	X	3	
Total		20		20	

7. Referencias

- <https://www.w3schools.com/java/default.asp>
- <https://stackoverflow.com/questions/2450954/how-to-randomize-shuffle-a-javascript-array>