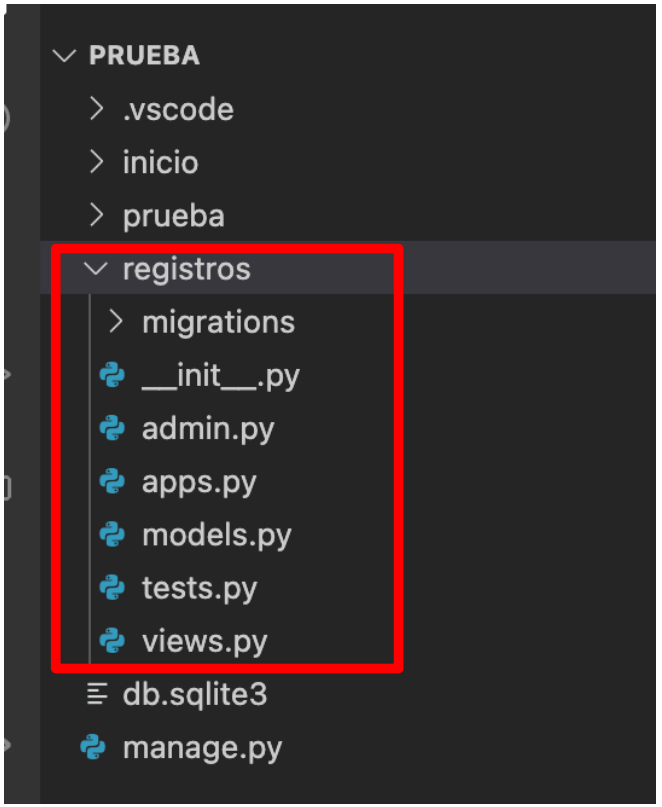
ADMIN DJANGO

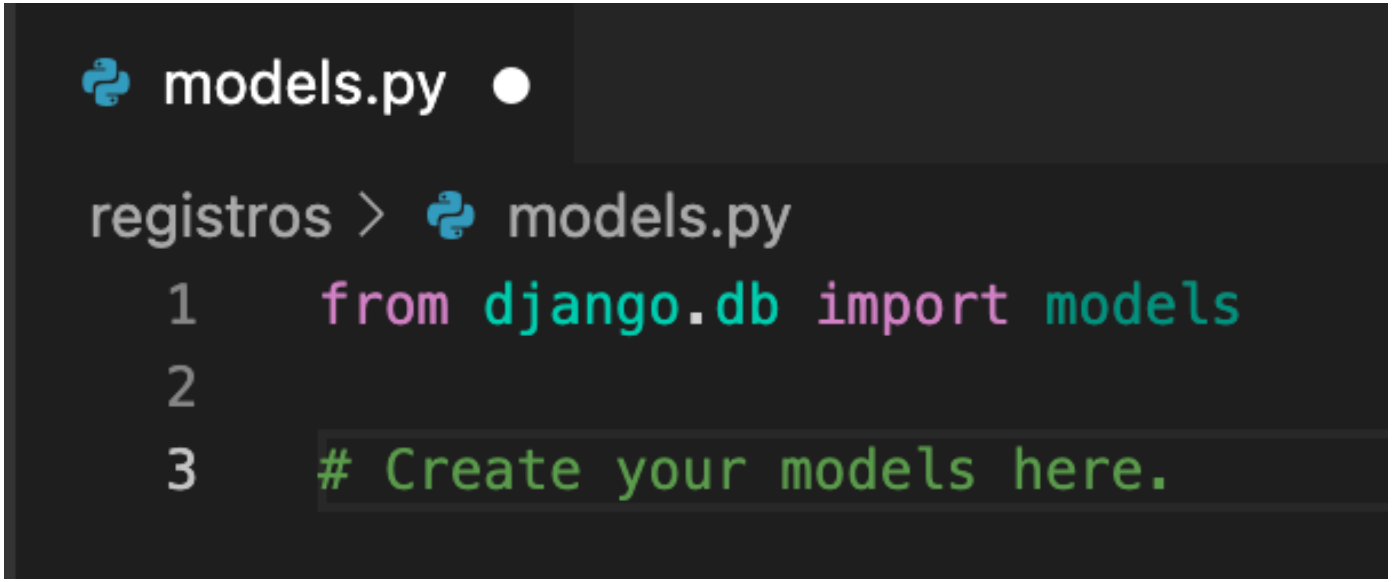
PRIMERA PARTE

INTERACCIÓN CON BASE DE DATOS

- Crearemos una nueva APP. → registros
  - python manage.py startapp **registros**



MODELS.PY → APP REGISTROS



MODELOS

- Un Django model es una tabla de la base de datos, los atributos de ese modelo se convierten en las columnas de esa tabla esos atributos reciben el nombre de django fields los cuales manejan automáticamente las conversiones de tipos de datos para la base de datos que estemos usando.
- Una de las grandes características de Django es su ORM (Object–relational mapping), gracias a el no tenemos que escribir ninguna consulta de base de datos, e incluso se recomienda NO escribir una al usar Django.

MODELOS

- ORM convierte los Django models y todas las operaciones que realiza con las consultas de base de datos correspondientes. Esto significa que toda la manipulación se hará ahora con los objetos de Python creados a partir de ese modelo, y todo el material de la base de datos subyacente será cuidado por el ORM de Django.
- El ORM de Django es compatible con todas las bases de datos principales como Postgres , MySQL , sqlite3 y otras. Esto significa que si deseamos cambiar de una base de datos a otra, podemos hacerlo sin cambiar una sola línea de la lógica de su aplicación, solo cambiamos la cadena de la base de datos en el archivo de configuración settings.py.

MODELS.PY → APP REGISTROS

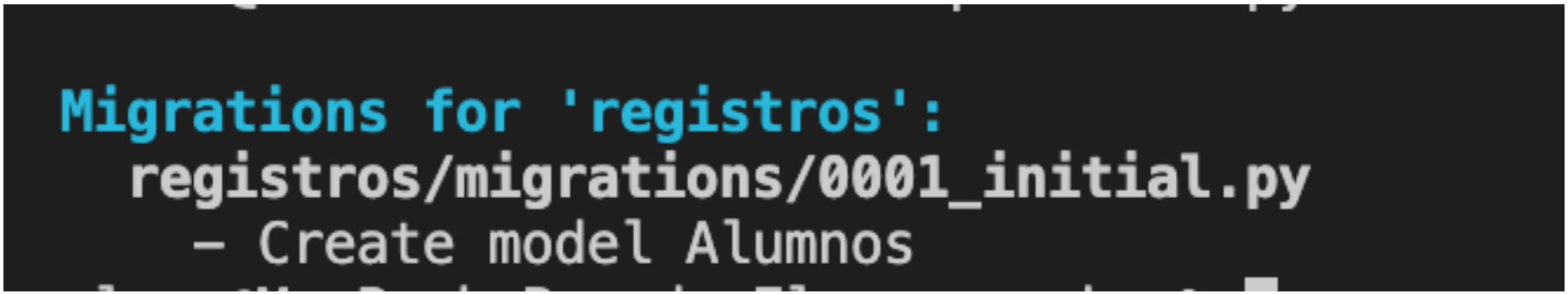
```
from django.db import models

class Alumnos(models.Model): #Define la estructura de nuestra tabla
      matricula = models.CharField(max_length=12) #Texto corto
      nombre = models.TextField() #Texto largo
      carrera = models.TextField()
      turno = models.CharField(max_length=10)
      created = models.DateTimeField(auto_now_add=True) #Fecha y tiempo
      updated = models.DateTimeField(auto_now_add=True)
```

SETTINGS.PY

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'inicio',
    'registros',
]
```

APLICAMOS EL CAMBIO AL MODELO

- python manage.py makemigrations registros



APLICAMOS EL CAMBIO A LA BASE DE DATOS

- python manage.py migrate registros

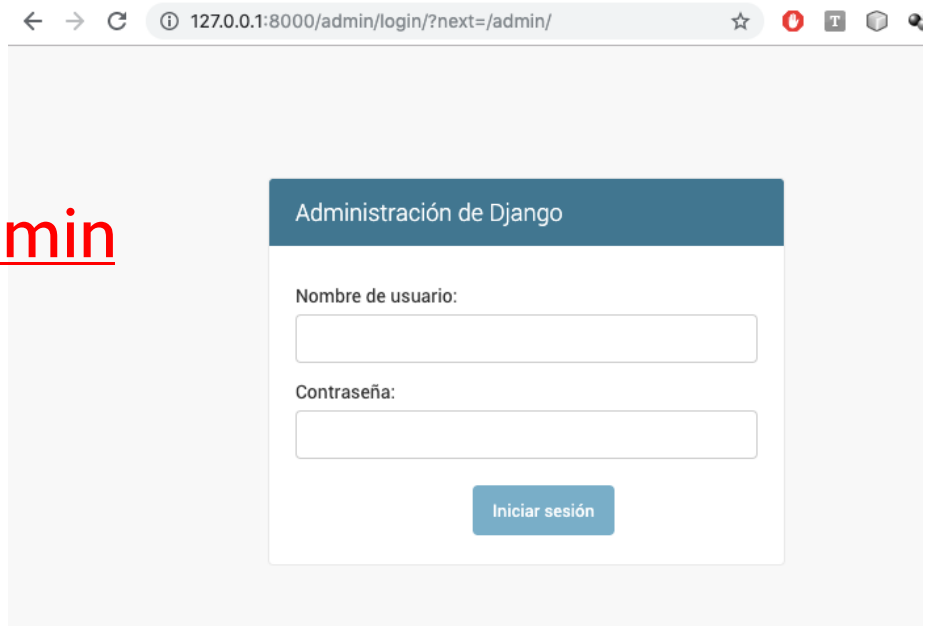
```
Operations to perform:
  Apply all migrations: registros
Running migrations:
  Applying registros.0001_initial... OK
elena@MacBook-Pro-de-Elena prueba %
```

Cualquier cambio al modelo de la tabla, deberá implicar la ejecución de los dos comandos (makemigrations y migrate)

UTILIZAR EL PANEL ADMINISTRADOR

Ejecutamos el servidor y accedemos al administrador colocando en la ruta /admin

http://127.0.0.1:8000/admin



URLS

```
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('',views.principal,name="Principal"),
23      path('contacto/',views.contacto,name="Contacto"),
24      path('formulario/',views.formulario,name="Formulario"),
25      path('ejemplo/',views.ejemplo,name="Ejemplo"),
26  ]
27
```

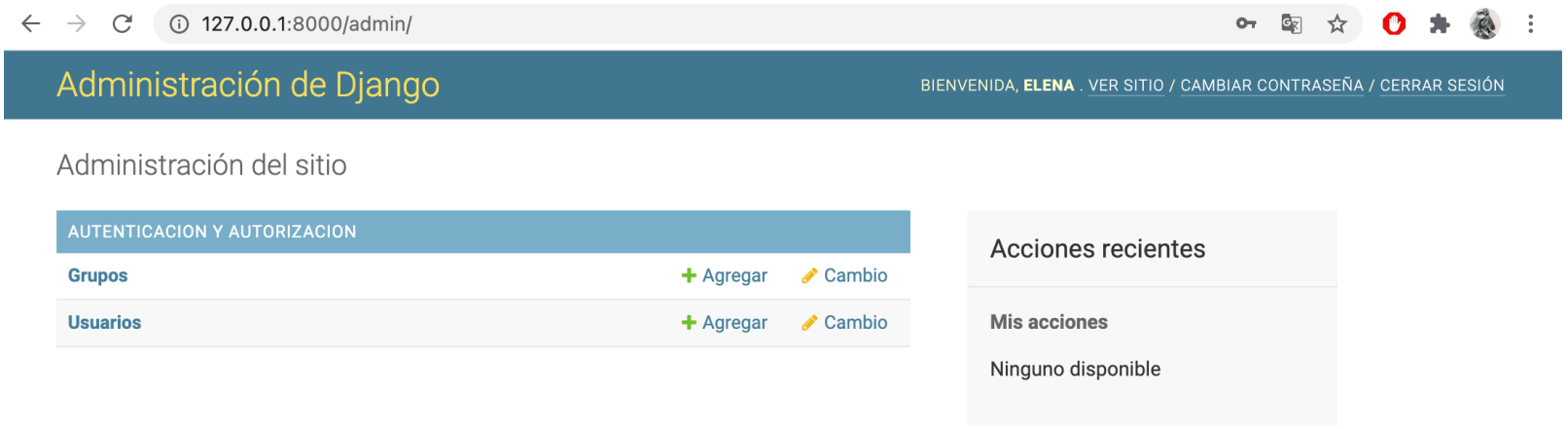
El panel de administración se encuentra definido por default

CREANDO UN USUARIO

- Creamos un usuario administrador
  - **python manage.py createsuperuser**

```
^C(django2) Air-de-Elena:novenoAHerencia elena$ python manage.py createsuperuser
Username (leave blank to use 'elena'): elena
Email address: elena@utm.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is too common.
This password is entirely numeric.
Superuser created successfully.
```

ACCEDEMOS AL PANEL DE ADMINISTRACIÓN



En el último ejemplo pudiste ingresar con tu usuario al panel de administración ahora procederemos a cambiar algunos elementos.

REGISTROS/ADMIN.PY

registros > admin.py

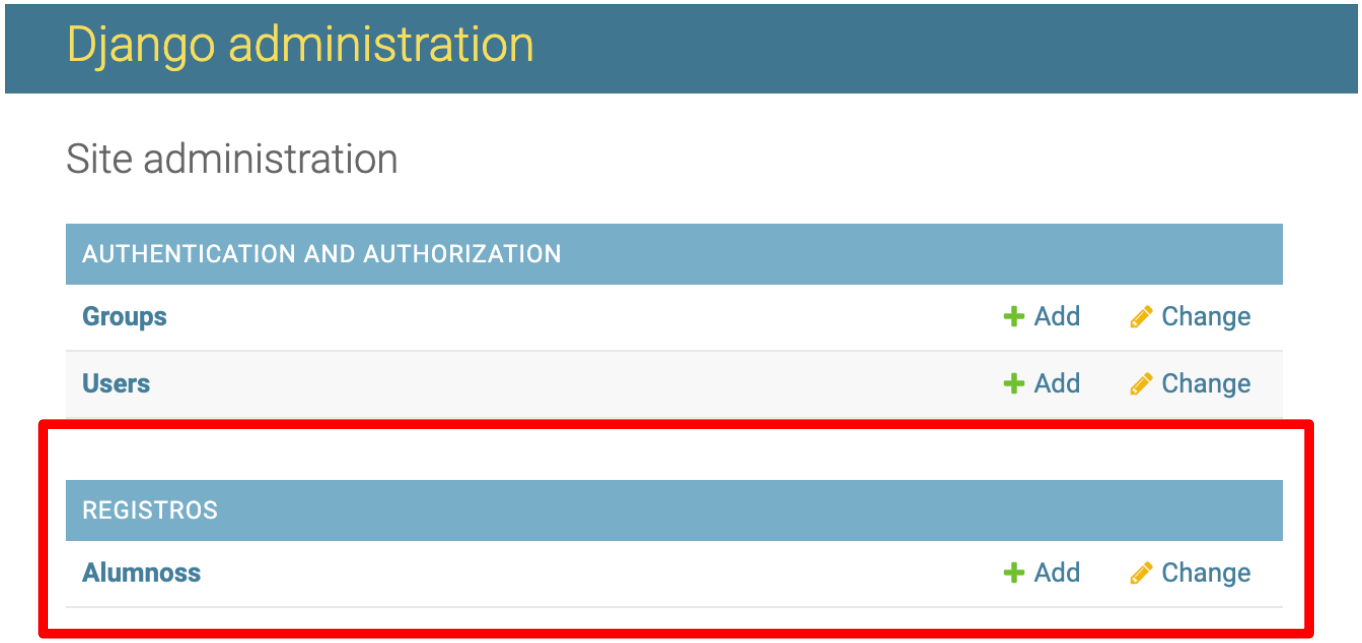
```
1    from django.contrib import admin
2
3    # Register your models here.
4
```

REGISTROS/ADMIN.PY

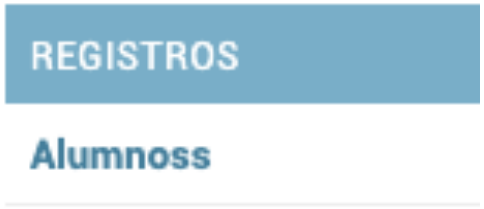
```
from django.contrib import admin
from .models import Alumnos

# Register your models here.
admin.site.register(Alumnos)
```

Alumno es el nombre de nuestro clase
en el archivo models.py de la app registro

- Actualiza el navegador, debererás observar que se muestra el módulo de registros con el acceso al modelo de alumnos.
- Observa como se agrega una "s" al final del nombre del modelo, en tu proyecto, el administrador agrega la s por default.



CAMBIANDO EL TITULO DE LA APP EN EL PANEL DE ADMINISTARDOR



registros/apps.py

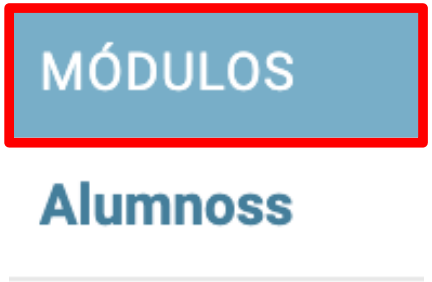
```
from django.apps import AppConfig


class RegistrosConfig(AppConfig):
name = 'registros'
verbose_name='Módulos'
```

copiamos

Agregamos

SETTINGS.PY

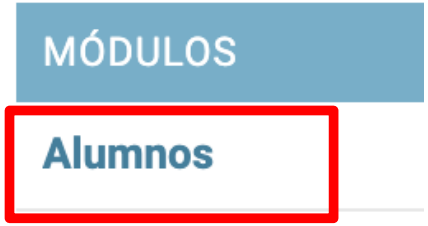
```
INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'inicio',
'registros.apps.RegistrosConfig',
]
```



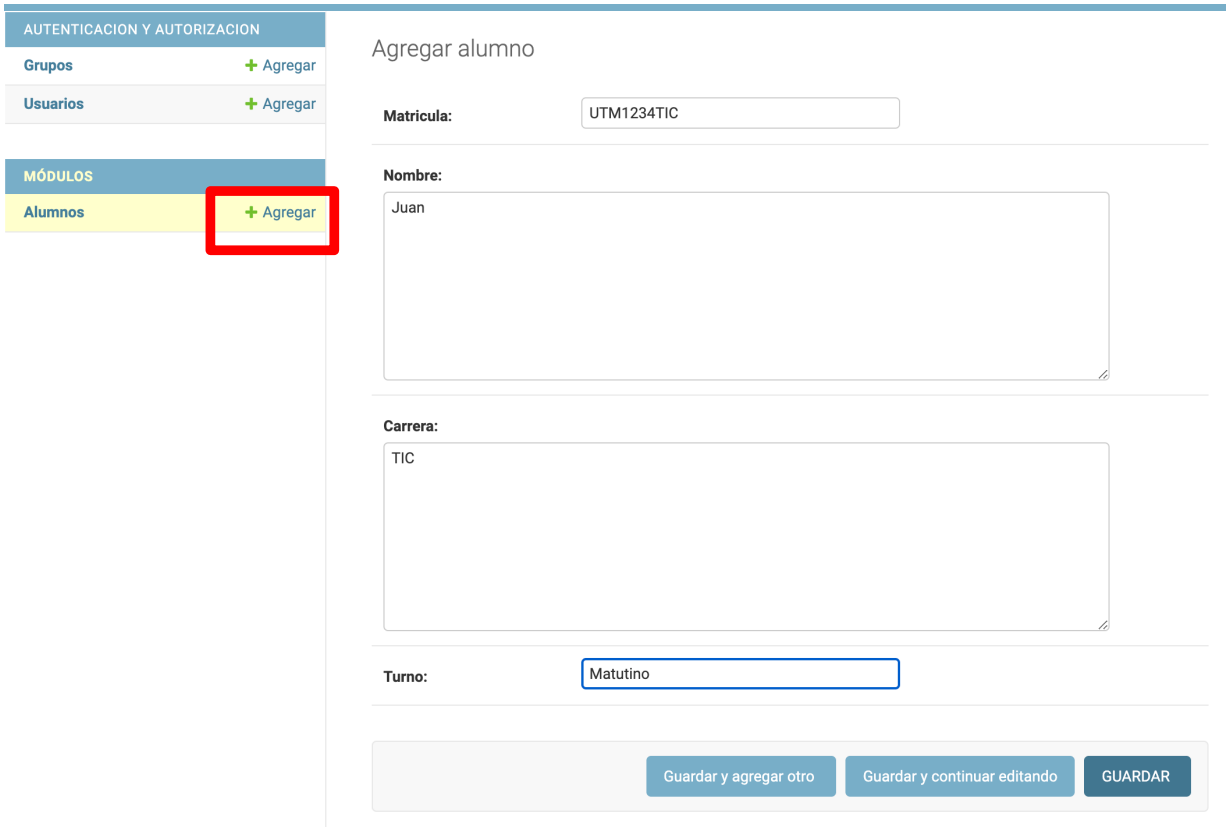
REGISTROS/MODELS.PY

- Cambiaremos el titulo del modelo y daremos un ordenamiento a los registros

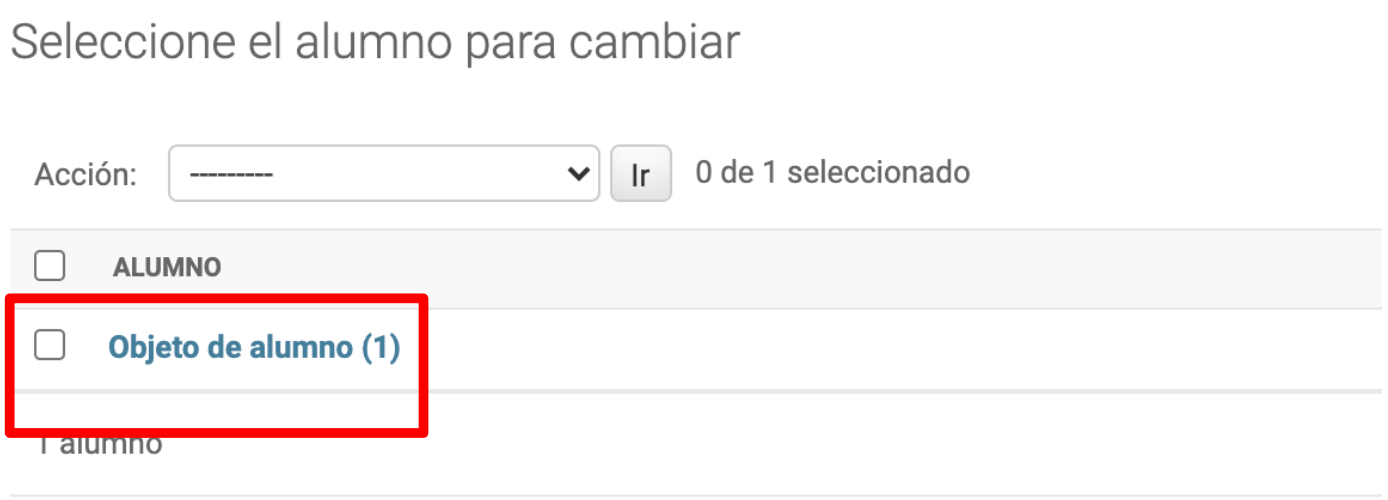
```
from django.db import models

class Alumnos(models.Model): #Define la estructura de nuestra tabla
        matricula = models.CharField(max_length=12) #Texto corto
        nombre = models.TextField() #Texto largo
        carrera = models.TextField()
        turno = models.CharField(max_length=10)
        created = models.DateTimeField(auto_now_add=True) #Fecha y tiempo
        updated = models.DateTimeField(auto_now_add=True) #Fecha y tiempo

        class Meta:
            verbose_name = "Alumno"
            verbose_name_plural = "Alumnos"
            ordering = ["-created"]
            #el menos indica que se ordenara del más reciente al más viejo
```



AGREGAMOS UN ALUMNO



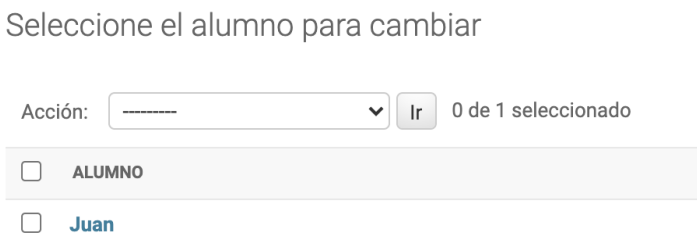
OBSERVARÁS QUE SE MUESTRA COMO OBJETO



REGISTROS/MODELS.PY

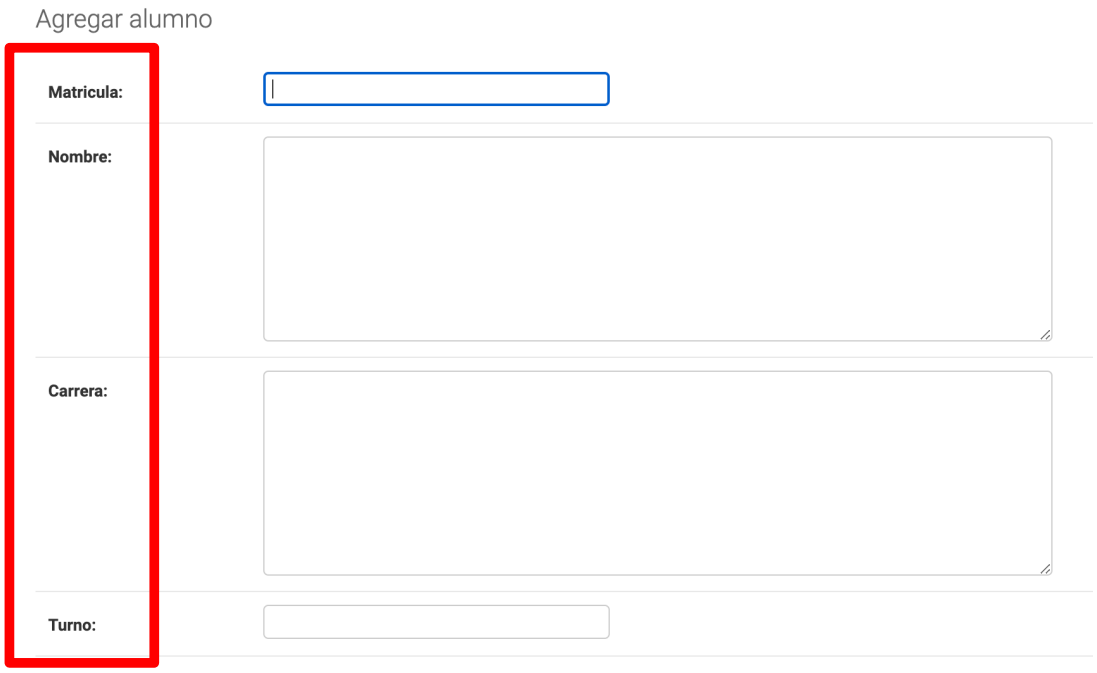
- Cambiaremos la apariencia de la lista de registros para mostrar el nombre del alumno

```
class Meta:
    verbose_name = "Alumno"
    verbose_name_plural = "Alumnos"
    ordering = ["-creado"]
   #el menos indica que se ordenara del más reciente al mas viejo

def __str__(self):
        return self.nombre
        #Indica que se mostrára el nombre como valor en la tabla
```



REGISTROS/MODELS.PY

- Es posible cambiar los nombre de los campos del formulario del administrador



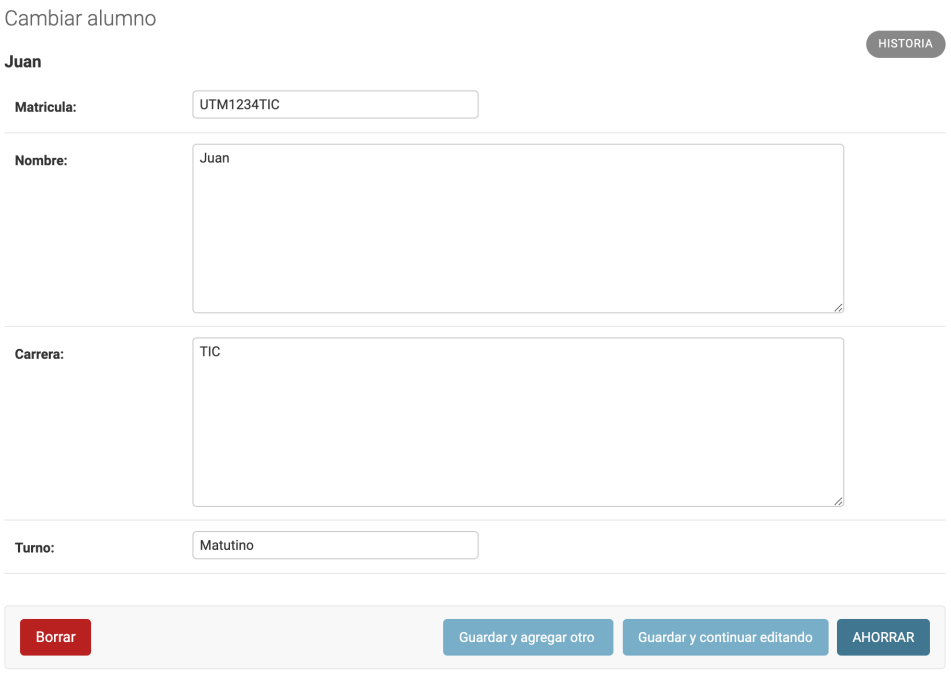
REGISTROS/MODELS.PY

- Cambiaremos el título de los campos

```
class Alumnos(models.Model):
      matricula = models.CharField(max_length=12,verbose_name="Mat")
```

VISUALIZACIÓN DE CAMPOS AUTOMÁTICOS

- Si acceden al usuario registrado, observarán que los campos automáticos (fechas de creación y modificación) no son visibles.



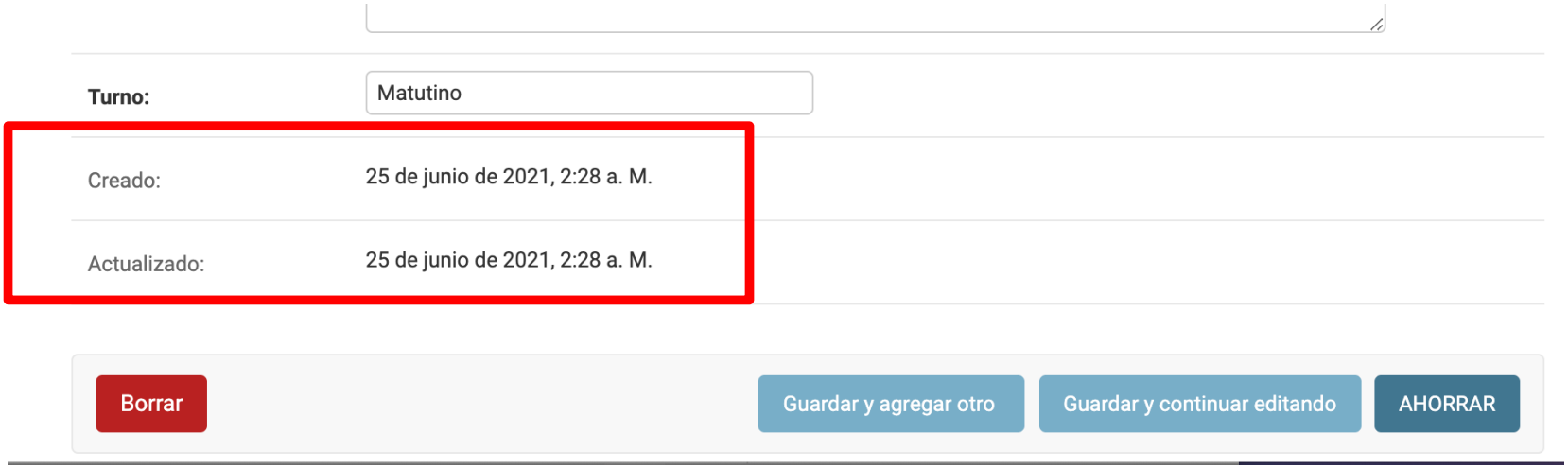
VISUALIZACIÓN DE CAMPOS AUTOMÁTICOS

- **registros/admin.py**

```
from django.contrib import admin
from .models import Alumnos

# Register your models here.

class AdministrarModelo(admin.ModelAdmin):
    readonly_fields = ('created', 'updated')

admin.site.register(Alumnos, AdministrarModelo)
```

VISUALIZACIÓN DE CAMPOS AUTOMÁTICOS

- Actualiza el navegador y deberás observar al final del formulario las fechas de creación y modificación.



RECUERDA QUE TU PROYECTO DE PRUEBA DEBE DE ESTÁR AL CORRIENTE HASTA ESTÉ PUNTO YA QUE SE REQUIERE PARA CONTINUAR CON LOS PROCESOS DE EJEMPLO SIGUIENTES.

TAREA I – SEGURIDAD – TIPOS DE DATOS ORM

- **ACTIVIDAD:**
  Lee sobre los tipos de datos aplicables en los modelos Django.
  - Documentación
    Oficial: https://docs.djangoproject.com/en/5.0/ref/models/fields/
- Realiza en tu cuaderno un resumen sobre los tipos de datos principales (texto, números, booleanos, caracteres, fechar, etc.) y sus restricciones (nulos, en blanco, default, auto incrementales, llaves, etc.). Define de manera clara su sintaxis, significado y uso.