# Informe de Laboratorio 06

## Tema: Laboratorio 06Django (Admin)

| Nota |
| --- |
| |

| Estudiante | Escuela | Asignatura |
| --- | --- | --- |
| Diego Alejandro Carbajal Gonzales, Mariel Alisson Jara Mamani, Jhonatan David Arias Quispe, Ricardo Mauricio Chambilla Perca<br>jariasq@unsa.edu.pe | Escuela Profesional de Ingeniería de Sistemas | Programación Web 2<br>Semestre: II<br>Código: 1702122 |

| Laboratorio | Tema | Duración |
| --- | --- | --- |
| 06 | Laboratorio 06Django (Admin) | 04 horas |

| Semestre académico | Fecha de inicio | Fecha de entrega |
| --- | --- | --- |
| 2024 - A | Del 05 de Junio 2024 | Al 08 de Junio 2024 |

# 1. Actividades

- 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.

## 1.1. 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.

# 2. Equipos, materiales y temas utilizados

- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- Termux
- NeoVim
- Git 2.42.0

- Cuenta en GitHub con el correo institucional.
- Latex
- Python 3.10.2
- Django 4.0.2
- Pip
- Virtualenv
- Figma
- Graphviz

## 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JhonatanDczel/CoderDojo-UNSA.git`

## 4. Estructura de laboratorio 06

- El contenido que se entrega en este laboratorio es el siguiente:

```
CoderDojo/
  |---/CoderDojoUNSA
      |---__init.py
      |---asgi.py
      |---settings.py
      |---urls.py
      |---wsgi.py
  |---/courses
      |---/migrations
      |---/models
          |---__init.py
          |---admin.py
          |---apps-py
          |---models.py.deprecated
          |---tests.py
          |---views.py
  |---/groups
      |---/migrations
      |---/models
          |---__init.py
          |---admin.py
          |---apps-py
          |---models.py.deprecated
          |---tests.py
          |---views.py
  |---/users
      |---/migrations
      |---/models
          |---__init.py
          |---admin.py
          |---apps-py
          |---models.py.deprecated
          |---tests.py
          |---views.py
```

```
|---/latex
    |---/acts
    |---/foot
    |---/head
    |---/img
    |---/src
    |--- informe-latex.tex
    |--- informe-latex.pdf
|---manage.py
|--- README.md
|---.gitignore
|---requirements.txt
|---diagram.png
|---diagram.dot
```

# 5. Planificación

Para tener claras las actividades a realizar, se ha realizado una planificación de las mismas, usando la herramienta Figma:

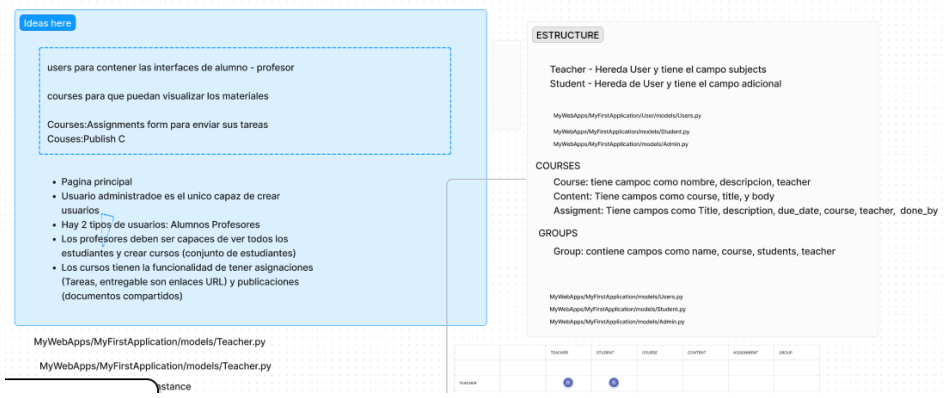


Figura 1: Planificación de actividades

# 6. Modelo de datos

Para el desarrollo de la aplicación se ha planteado el siguiente modelo de datos:

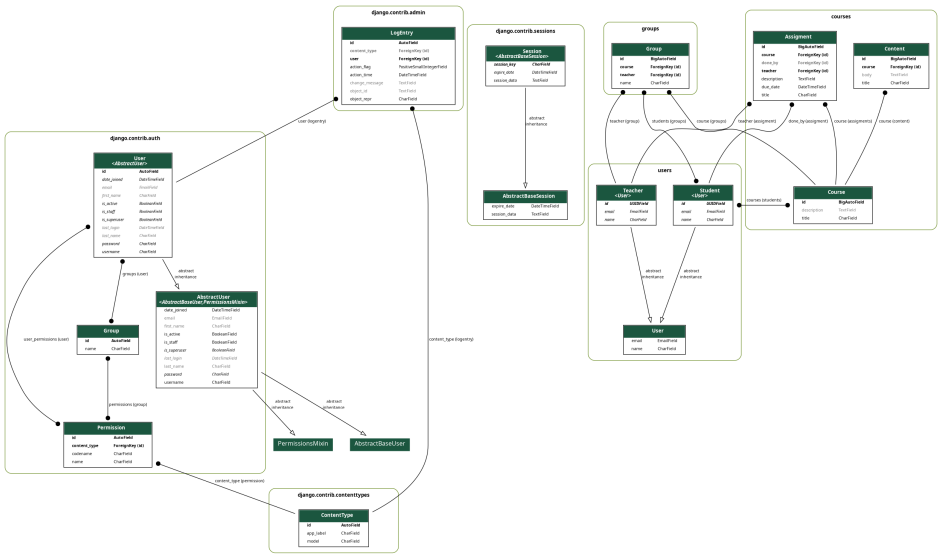


Figura 2: Modelo de datos

A continuacion veremos una explicacion mas detallada sobre los modelos en Django.

# 7. Modelos Python

Los siguientes modelos fueron creados en Django, en archivos separados, en la carpeta models de cada aplicación:

## 7.1. Student

El modelo Student hereda de User, y tiene una relación de muchos a uno con Course, es decir, un estudiante puede estar inscrito en varios cursos, pero un curso solo puede tener un estudiante.

```
class Student(User):
    courses = models.ForeignKey(Course, related_name='students', on_delete=models.CASCADE)
```

## 7.2. Teacher

El modelo Teacher hereda de User, y tiene una relación de muchos a uno con Groups, que maneja la lógica de los grupos de estudiantes y sus cursos.

```
class Teacher(User):
    pass
```

## 7.3. Course

El modelo Course tiene un título y una descripción, y una relación de muchos a uno con Teacher, es decir, un curso solo puede tener un profesor, pero un profesor puede tener varios cursos.

Los campos title y description son obligatorios, y el campo teacher es opcional.

```
class Course(models.Model):
  title = models.CharField(max_length=255, blank=False, null=False)
```

```
  description = models.TextField(blank=True, null=False)

  #teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE)
  def __str__(self):
    return self.title
```

## 7.4. Content

El modelo Content tiene un título y un cuerpo, y una relación de muchos a uno con Course, es decir, un contenido solo puede pertenecer a un curso, pero un curso puede tener varios contenidos.

Representa el contenido de un curso, como una lección, un video, un archivo, etc.

```
class Content(models.Model):
  course = models.ForeignKey(Course, on_delete=models.CASCADE)
  title = models.CharField(blank=False, max_length=255, null=False)
  body = models.TextField(blank=True, null=False)

  class Meta:
    ordering = ['course', 'title', 'body']

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

## 7.5. Assigment

El modelo Assignment tiene un título, una descripción, una fecha de entrega, y una relación de muchos a uno con Course y Teacher, y una relación de uno a uno con Student, es decir, una tarea solo puede ser asignada a un estudiante, pero un estudiante puede tener varias tareas.

Los estudiantes enviarán sus tareas a través de la aplicación, y los profesores podrán calificarlas, mediante un link.

```
class Assignment(models.Model):
    title = models.CharField(max_length=255, blank=False, null=False)
    description = models.TextField(blank=False, null=False)
    due_date = models.DateTimeField()
    course = models.ForeignKey(Course, related_name='assigments', on_delete=models.CASCADE)
    teacher = models.ForeignKey('users.Teacher', on_delete=models.CASCADE)
    done_by = models.ForeignKey('users.Student', null=True, blank=True, on_delete=models.SET_NULL)

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

## 7.6. Groups

El modelo Group tiene un nombre, una relación de muchos a uno con Course y Teacher, y una relación de muchos a muchos con Student, es decir, un grupo solo puede tener un profesor y un curso, pero un profesor puede tener varios grupos, y un grupo puede tener varios estudiantes.

Los grupos representan la división de los estudiantes en clases, y cada grupo tiene un profesor y un curso asignado.

```
class Group(models.Model):

    name = models.CharField(max_length=255, blank=False, null=False)
    course = models.ForeignKey(Course, related_name='groups', on_delete=models.CASCADE)
    students = models.ManyToManyField('users.Student', related_name='groups')
    teacher = models.ForeignKey('users.Teacher', on_delete=models.CASCADE)
```

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

# 8. Admin CRUD

Para poder realizar el CRUD de los modelos en Django, se debe registrar los modelos en el archivo admin.py de cada aplicación. A continuación se muestra un ejemplo de cómo se registra el modelo Course en el archivo admin.py de la aplicación courses:

```
from django.contrib import admin
from courses.models import Course
from courses.models import Content
from courses.models import Assignment
from users.models import Student
from users.models import Teacher
from groups.models import Group


# Register your models here.
admin.site.register(Course)
admin.site.register(Content)
admin.site.register(Assignment)
admin.site.register(Student)
admin.site.register(Teacher)
admin.site.register(Group)
```

Esta es una forma de registrar los modelos en el panel de administración de Django, para poder realizar el CRUD de los modelos.

Aqui se muestra el acceso por medio de la URL /admin, donde se puede ver los modelos registrados en el panel de administración de Django.

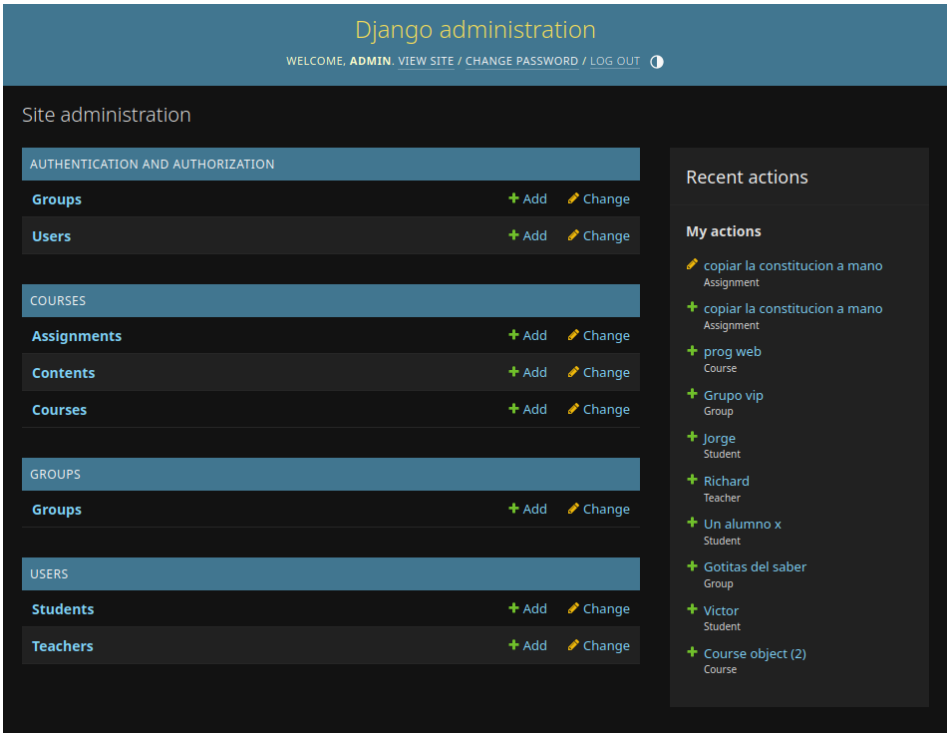


Figura 3: Panel de administración de Django

# 9. Apreciaciones del laboratorio

- **Mariel:** Aprendí que Django ofrece una forma poderosa y eficiente de trabajar con bases de datos utilizando modelos, que son esencialmente clases de Python que representan tablas en la base de datos. Esto me permite definir la estructura de los datos de mi aplicación de una manera limpia y legible, y luego utilizar el ORM de Django para realizar operaciones CRUD en esos datos sin tener que escribir consultas SQL directamente. Además, el concepto de aplicaciones en Django me permite organizar mi código de una manera modular y escalable, lo que facilita el mantenimiento y la reutilización del código.

- **Diego:** Me di cuenta de la utilidad del panel de administración de Django (admin) como una forma conveniente de interactuar con los datos de mi aplicación. Puedo registrar mis modelos en el panel de administración y luego gestionar los datos de mi aplicación directamente desde allí sin tener que escribir código adicional. Esto facilita la administración de los datos durante el desarrollo y también puede ser útil en la fase de producción para realizar tareas de administración sin tener que crear interfaces de usuario personalizadas.

- **Jhonatan:** Descubrí que el proceso de migración en Django es una herramienta poderosa para mantener actualizada la estructura de la base de datos de mi aplicación con respecto a los cambios en los modelos. Cuando realizo cambios en mis modelos, como agregar nuevos campos o tablas, puedo generar migraciones usando makemigrations y luego aplicar esas migraciones a mi base de datos usando migrate. Esto me permite mantener la integridad de los datos y garantizar que mi aplicación funcione correctamente incluso después de realizar cambios en los modelos.

- **Ricardo:** Aprendí sobre la importancia de la modularidad en Django, especialmente al trabajar con modelos. La capacidad de organizar mi código en aplicaciones me permite mantener una estructura clara y separada para diferentes componentes de mi proyecto. Esto no solo facilita el

desarrollo y la colaboración en equipo, sino que también mejora la escalabilidad y la reutilización del código. Además, el uso de el archivo init en la carpeta models para establecer enlaces entre los modelos de diferentes archivos me permite mantener mi código ordenado y fácil de entender.

## 10. Commits

Estos son los commits mas importantes realizados en el proyecto:

[language=bash] commit b01493cbf970099021f37da47fb26ceb5a04e1de Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿Date: Sat Jun 8 21:54:41 2024 -0500

Actualiza los requerimientos

commit 78cc7f71b752a448bd5269a8a31d8d8d29d0fcc0 Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿Date: Sat Jun 8 21:53:05 2024 -0500

Diagrama de la base de datos

commit 71800a5e4cdd7ec6bf19623c30e2a3568855f103 Author: Alsnj20 ¡marieljara656@gmail.com¿Date: Sat Jun 8 21:25:14 2024 -0500

Migraciones de la app user y courses

commit 54e2f4780f88a95754a33d6b80ebef48fd0fd7ae Author: L0rD1ego ¡diegoalejandrocarbajal@gmail.com¿Date: Sat Jun 8 21:19:10 2024 -0500

en este cambio se agrego group en groups y assigments en courses

commit 2ba0eac33ff6b5c138563d40c9bc2268cc63177f Author: Alsnj20 ¡marieljara656@gmail.com¿Date: Sat Jun 8 20:58:06 2024 -0500

Añadiendo el esqueleto de course y content en la app courses

commit 70f16b293f2dbf74e1df372e8924eee9f93e87a0 Author: rikich3 ¡rchambillap@unsa.edu.pe¿Date: Sat Jun 8 20:45:15 2024 -0500

Realizado el esqueleto de users y students techers

commit 6f0b78d49cb5bd5d3e54862eb8da30e96bcebf0e Author: Alsnj20 ¡marieljara656@gmail.com¿Date: Sat Jun 8 20:16:55 2024 -0500

Modificación del archivo con los requerimientos

commit fceda264a2745cc2ae6d221933259bc4891bcd4d Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿Date: Sat Jun 8 16:20:33 2024 -0500

Estructura inicial

commit acb10a5d94fbbbbf64e36f36f08f3088e39f5cdc Author: Alsnj20 ¡marieljara656@gmail.com¿Date: Sat Jun 8 16:17:21 2024 -0500

Readme: Indicaciones para usar el projecto

commit 133b54fefadf469534f1467de27930c988feb936 Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿Date: Fri Jun 7 18:44:39 2024 -0500

Arregla error en el nombre del proyecto

commit e6e034fee85432a4261cabe50de07a1863762ffa Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿Date: Fri Jun 7 18:43:24 2024 -0500

Agrega el archivo requeriments.txt

commit 307422f74a2f5d3f33600b8e827484f2d83a2d51 Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿Date: Fri Jun 7 18:31:41 2024 -0500

Carga las configuraciones iniciales

Agrega las aplicaciones necesarias para el proyecto: users, assignments y courses

# 11. Rúbricas

## 11.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

| | Nivel | | | |
|---|---|---|---|---|
| **Puntos** | Insatisfactorio 25 % | En Proceso 50 % | Satisfactorio 75 % | Sobresaliente 100 % |
| **2.0** | 0.5 | 1.0 | 1.5 | 2.0 |
| **4.0** | 1.0 | 2.0 | 3.0 | 4.0 |

Tabla 2: Rúbrica para contenido del Informe y demostración

| Contenido y demostración | | Puntos | Checklist | Estudiante | Profesor |
|---|---|---|---|---|---|
| **1. GitHub** | Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entrega bles. (Se descontará puntos por error o onser vación) | 2 | X | 2 | |
| **2. Commits** | Hay porciones de código fuente asociado a los commits planificados con explicaciones deta lladas. (El profesor puede preguntar para re frendar calificación). | 4 | X | 4 | |
| **3. Codigo fuente** | Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones. | 2 | X | 2 | |
| **4. Ejecución** | Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión) | 2 | X | 1.5 | |
| **5. Pregunta** | Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación) | 2 | X | 2 | |
| **6. Fechas** | Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos. | 2 | X | 2 | |
| **7. Ortografía** | El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado) | 2 | X | 1.5 | |
| **8. Madurez** | El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agre gando diagramas generados a partir del códi go fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar cali ficación). | 4 | X | 3 | |
| **Total** | | 20 | | 16 | |