# Informe de Laboratorio 07

## Tema: Laboratorio 07Django forms, views and templates

| 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,<br>mjarama@unsa.edu.pe,<br>dcarbajalg@unsa.edu.pe,<br>rchambillap@unsa.edu.pe | Escuela Profesional de Ingeniería de Sistemas | Programación Web 2<br>Semestre: II<br>Código: 1702122 |

| Laboratorio | Tema | Duración |
|---|---|---|
| 07 | Laboratorio 07Django forms, views and templates | 04 horas |

| Semestre académico | Fecha de inicio | Fecha de entrega |
|---|---|---|
| 2024 - A | Del 12 de Junio 2024 | Al 16 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/rikich3/lab7Forms`

## 4. Estructura de laboratorio 07

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

```
lab07Forms/
  |---/CoderDojoUNSA
      |---__init.py
      |---asgi.py
      |---settings.py
      |---urls.py
      |---wsgi.py
  |---/courses
      |---/migrations
      |---/models
          |---__init__.py
          |---assigment.py
          |---Course.py
          |---NotasAlumnoPorCurso.py
      |---templates/
          |---sistema/
              |---create_curso.html
              |---index.html
              |---list_cursos.html
      |---__init_.py
      |---admin.py
      |---apps.py
      |---forms.py
      |---models.py.deprecated
      |---tests.py
      |---views.py
  |---/users
      |---/migrations
```

```
        |---/models
            |---__init.py
            |---Student.py
            |---User.py
        |---templates/
            |---form.html
            |---success.html
        |---__init_.py
        |---admin.py
        |---apps.py
        |---forms.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
```

# 5. Planificación

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

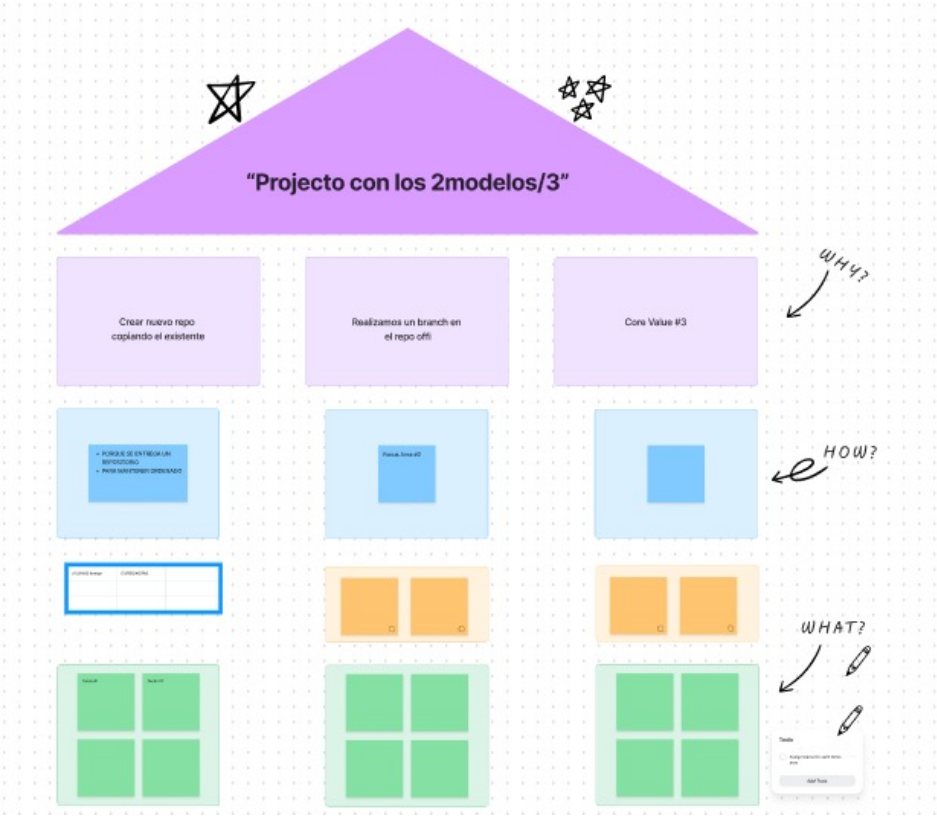



Figura 1: Planificación de actividades

- Para este laboratorio hemos decidido hacer el laboratorio todos al mismo tiempo, esto a traves de la app VS Code que nos otorga la extensión Live Share. A continuación se muestran las capturas de como se interactuan con esta extensión

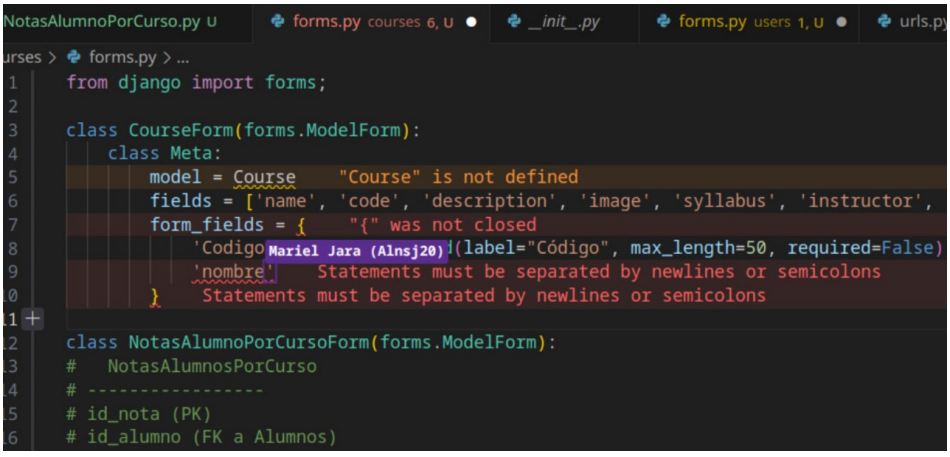



Figura 2: Integrante Mariel en LiveShare

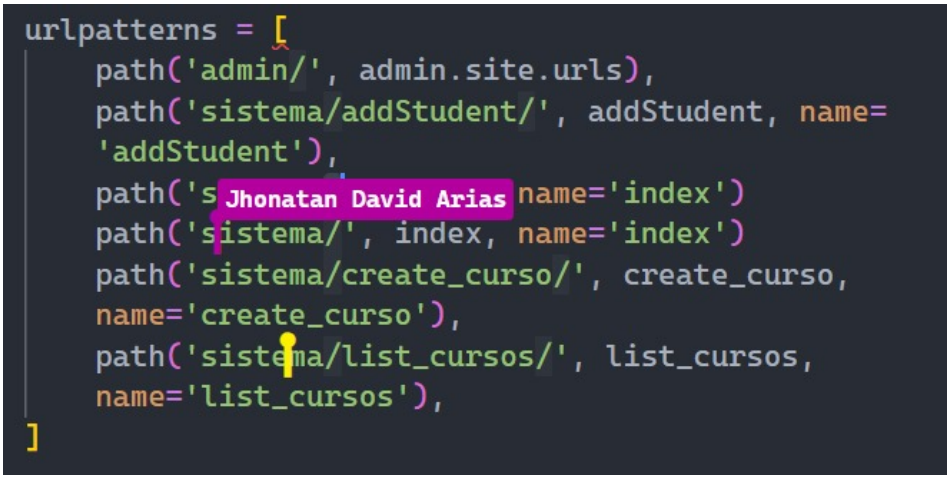



Figura 3: Integrante Jhonatan en LiveShare


Figura 4: Integrante Ricardo en LiveShare

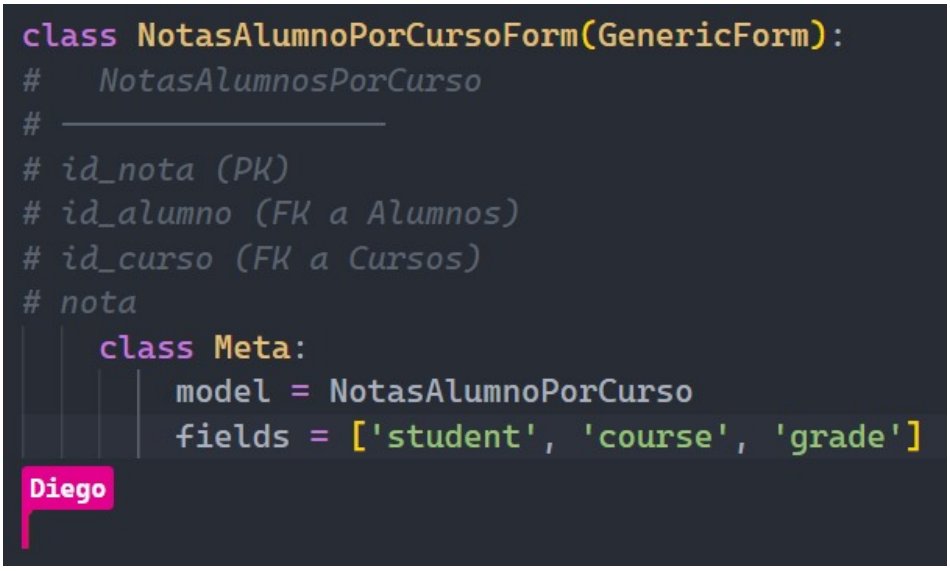



Figura 5: Integrante Diego en LiveShare

# 6. Laboratorio 07

Este informe detalla el desarrollo y la implementación de un sistema académico utilizando Django, que permite la gestión de cursos, estudiantes y sus notas. El sistema se divide en dos aplicaciones principales: `courses` y `users`. En esta seccion, nos centraremos en la aplicación `courses`, abordando los modelos (`Course.py, NotasAlumnoPorCurso.py`), vistas (`views.py`), formularios (`forms.py`), y plantillas del directorio (`templates`). A través de ejemplos de código, se explicarán las funcionalidades y la lógica detrás de cada componente.

# 7. Modelos de Datos (`models.py`)

## 7.1. Modelo de Curso (`Course`)

El modelo `Course` define la estructura de los cursos que se ofrecen en el sistema. Cada curso tiene un código único, un nombre y una descripción.

Listing 1: Modelo de Curso

```python
from django.db import models

class Course(models.Model):
    code = models.CharField(max_length=10, unique=True, primary_key=True, default='')
    name = models.CharField(max_length=255, blank=False, null=False)
    description = models.TextField(blank=True, null=False)

    def __str__(self):
        return f"({self.code}) {self.name}"
```

### 7.1.1. Explicación del Código

- `code`: Campo de tipo `CharField` que actúa como clave primaria y debe ser único.
- `name`: Campo de tipo `CharField` para el nombre del curso.

- `description`: Campo de tipo `TextField` para la descripción del curso.
- `__str__`: Método que define la representación en cadena del objeto `Course`.

## 7.2. Modelo de Notas por Curso (`NotasAlumnoPorCurso`)

El modelo `NotasAlumnoPorCurso` registra las notas de los estudiantes en los cursos. Este modelo incluye relaciones con los modelos `Course` y `Student`.

Listing 2: Modelo de Notas por Curso

```python
from django.db import models
from django.core.validators import MinValueValidator, MaxValueValidator
from courses.models import Course

class NotasAlumnoPorCurso(models.Model):
    id_nota = models.CharField(max_length=100, unique=True, primary_key=True)
    id_curso = models.ForeignKey(Course, related_name='curso', on_delete=models.CASCADE)
    id_alumno = models.ForeignKey('users.Student', related_name='student', on_delete=models.CASCADE)
    nota = models.IntegerField(validators=[MinValueValidator(0), MaxValueValidator(100)])

    def __str__(self):
        return f"({self.id_alumno}) {self.nota}"
```

### 7.2.1. Explicación del Código

- `id`$_{n}ota$ : $Campodetipo$`CharField`$queactúacomoclaveprimariaydebeserúnico.$`id`$_{c}urso$ : $Campodetipo$`ForeignKey`$querelaci$

# 8. Vistas (`views.py`)

Las vistas manejan la lógica de las solicitudes y respuestas del sistema. En esta sección, se describen las vistas para la creación y listado de cursos, así como la adición de notas.

## 8.1. Vista de Inicio (`index`)

Listing 3: Vista de Inicio

```python
from django.shortcuts import render

def index(request):
    return render(request, 'sistema/index.html')
```

## 8.2. Vista para Crear Curso (`create_curso`)

Listing 4: Vista para Crear Curso

```python
from django.shortcuts import render, redirect
from .forms import CourseForm

def create_curso(request):
    if request.method == 'POST':
        form = CourseForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('list_cursos')
    else:
```

```
        form = CourseForm()
    return render(request, 'sistema/create_curso.html', {'form': form})
```

### 8.2.1. Explicación del Código

- Si el método de la solicitud es `POST`, se procesa el formulario.
- Si el formulario es válido, se guarda el nuevo curso y se redirige a la lista de cursos.
- Si el método es `GET`, se muestra un formulario vacío.

## 8.3. Vista para Listar Cursos (`list_cursos`)

Listing 5: Vista para Listar Cursos

```
from django.shortcuts import render
from .models import Course

def list_cursos(request):
    cursos = Course.objects.all()
    return render(request, 'sistema/list_cursos.html', {'cursos': cursos})
```

### 8.3.1. Explicación del Código

- Recupera todos los cursos de la base de datos y los pasa al template para ser mostrados.

## 8.4. Vista para Agregar Nota (`addGrade`)

Listing 6: Vista para Agregar Nota

```
from django.shortcuts import render, redirect
from .forms import NotasAlumnoPorCursoForm

def addGrade(request):
    if request.method == 'POST':
        form = NotasAlumnoPorCursoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('success')
    else:
        form = NotasAlumnoPorCursoForm()
    return render(request, 'sistema/add_grade.html', {'form': form})
```

### 8.4.1. Explicación del Código

- Similar a `create_curso`, pero para la adición de notas.

# 9. Formularios (`forms.py`)

Los formularios son esenciales para la entrada y validación de datos en Django.

## 9.1. Formulario de Curso (`CourseForm`)

Listing 7: Formulario de Curso

```python
from django import forms
from .models import Course

class CourseForm(forms.ModelForm):
    class Meta:
        model = Course
        fields = ['name', 'code', 'description']
```

### 9.1.1. Explicación del Código

- Se especifica el modelo y los campos a incluir en el formulario.

## 9.2. Formulario de Notas por Curso (`NotasAlumnoPorCursoForm`)

Listing 8: Formulario de Notas por Curso

```python
from django import forms
from .models import NotasAlumnoPorCurso

class NotasAlumnoPorCursoForm(forms.ModelForm):
    class Meta:
        model = NotasAlumnoPorCurso
        fields = ['id_curso', 'id_alumno', 'nota']
        labels = {
            'id_alumno': 'Alumno',
            'id_curso': 'Curso',
            'nota': 'Nota'
        }

    def clean_grade(self):
        grade = self.cleaned_data.get('grade')
        if grade < 0 or grade > 100:
            raise forms.ValidationError('La nota debe estar entre 0 y 100')
        return grade
```

### 9.2.1. Explicación del Código

- Se especifican el modelo y los campos a incluir.
- Se incluye un método de validación para asegurarse de que la nota esté dentro del rango permitido.

# 10. Plantillas (`templates`)

Las plantillas definen cómo se presenta la información en la interfaz de usuario.

## 10.1. Plantilla para Crear Curso (`create_curso.html`)

Listing 9: Plantilla para Crear Curso

```html
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Crear Curso</title>
</head>
<body>
    <header>Sistema Acadmico</header>
    <main>
        <a href="{% url 'index' %}">Regresar al men</a>
        <h2>Crear Curso</h2>
        <form method="post">
            {% csrf_token %}
            <table>
                {{ form.as_table }}
            </table>
            <button type="submit">Crear</button>
        </form>
        <a href="{% url 'list_cursos' %}">Lista de cursos</a>
    </main>
</body>
</html>
```

## 10.2. Plantilla para Listar Cursos (`list_cursos.html`)

Listing 10: Plantilla para Listar Cursos

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Lista de Cursos</title>
</head>
<body>
    <header>Sistema Acadmico</header>
    <main>
        <a href="{% url 'index' %}">Regresar al men</a>
        <h2>Lista de cursos</h2>
        <table>
            <tr>
                <th> Cdigo </th>
                <th>Nombre</th>
                <th>Descripcin</th>
            </tr>
            {% for curso in cursos %}
            <tr>
                <td>{{ curso.code }}</td>
                <td>{{ curso.name }}</td>
                <td>{{ curso.description }}</td>
            </tr>
            {% endfor %}
        </table>
        <a href="{% url 'create_curso' %}">Crear un curso</a>
    </main>
</body>
</html>
```

## 10.3. Plantilla de Índice (`index.html`)

Listing 11: Plantilla de Índice

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    {% load static %}
    <link rel="stylesheet" href="{% static 'sistema/style.css' %}" />
    <title>ndice</title>
</head>
<body>
    <header>
        <h1>Sistema Acadmico</h1>
    </header>
    <main>
        <section class="opciones">
            <h2>Opciones Disponibles</h2>
            <div class="opcion">
                <a href="{% url 'list_alumnos' %}">Lista de Alumnos</a>
            </div>
            <div class="opcion">
                <a href="{% url 'create_alumno' %}">Crear Alumno</a>
            </div>
            <div class="opcion">
                <a href="{% url 'list_cursos' %}">Lista de Cursos</a>
            </div>
            <div class="opcion">
                <a href="{% url 'create_curso' %}">Crear Curso</a>
            </div>
        </section>
    </main>
</body>
</html>
```

# 11. Users app

Ahora nos centraremos en la aplicación `users`, abordando los modelos (`models.py`), vistas (`views.py`), formularios (`forms.py`), y plantillas (`templates`). A través de ejemplos de código, se explicarán las funcionalidades y la lógica detrás de cada componente.

# 12. Modelos de Datos (`models.py`)

## 12.1. Modelo de Usuario (`User`)

El modelo `User` es una clase abstracta que define los campos comunes a todos los usuarios del sistema, como el ID, nombre y correo electrónico. Esta clase es la base para otros modelos específicos de usuario, como `Student`.

Listing 12: Modelo `User`

```python
# users/models/User.py
from django.db import models
import uuid

class User(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    name = models.CharField(max_length=255, null=False, blank=False)
    email = models.EmailField(unique=True, null=False, blank=False)

    class Meta:
```

```
        abstract = True # Esto hace que User sea una clase base abstracta

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

#### 12.1.1. Explicacion del Codigo:

- `id`: Campo de tipo `UUIDField` que actúa como clave primaria y se genera automáticamente.
- `name`: Campo de tipo `CharField` para el nombre del usuario.
- `email`: Campo de tipo `EmailField` que debe ser único.
- `abstract = True`: Indica que `User` es una clase abstracta y no se creará una tabla para esta clase en la base de datos.

### 12.2. Modelo de Estudiante (`Student`)

El modelo `Student` hereda de `User` y añade una relación con el modelo `Course`, permitiendo asignar cursos a los estudiantes.

Listing 13: Modelo `Student`

```
# users/models/student.py
from django.db import models
from .User import User
from courses.models import Course

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

#### 12.2.1. Explicacion del Codigo:

- `courses`: Campo de tipo `ForeignKey` que relaciona al estudiante con un curso. Permite la relación de muchos a uno (muchos estudiantes pueden estar inscritos en un curso).

## 13. Vistas (`views.py`)

Las vistas manejan la lógica de las solicitudes y respuestas del sistema. En esta sección, se describe la vista para la creación de estudiantes.

### 13.1. Vista para Agregar Estudiante (`addStudent`)

Listing 14: Vista para Agregar Estudiante

```
# users/views.py
from django.shortcuts import render, redirect
from .forms import StudentForm

def addStudent(request):
    if request.method == 'POST':
        form = StudentForm(request.POST)
        if form.is_valid():
            form.submit()
            return render(request, 'success.html', {})
```

```
    else:
        form = StudentForm()
    return render(request, 'form.html', {'form': form})
```

#### 13.1.1. Explicacion del Codigo:

- Si el método de la solicitud es `POST`, se procesa el formulario.
- Si el formulario es válido, se guarda el nuevo estudiante y se muestra una página de éxito.
- Si el método es `GET`, se muestra un formulario vacío.

## 14. Formularios (`forms.py`)

Los formularios son esenciales para la entrada y validación de datos en Django.

### 14.1. Formulario de Estudiante (`StudentForm`)

Listing 15: Formulario de Estudiante

```
# users/forms.py
from django import forms
from .models import Student

class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ['name', 'email']

    def clean(self):
        cleaned_data = super().clean()
        student_id = cleaned_data.get('id')
        if Student.objects.filter(id=student_id).exists():
            raise forms.ValidationError('Ya existe un estudiante con ese id')
        return cleaned_data

    def submit(self, commit=True):
        instance = super().save(commit=False)
        if commit:
            instance.save()
        return instance
```

#### 14.1.1. Explicacion del Codigo:

- Se especifica el modelo y los campos a incluir en el formulario.
- `clean`: Método de validación que verifica si ya existe un estudiante con el mismo ID.
- `submit`: Método para guardar la instancia del formulario.

## 15. Plantillas (`templates`)

Las plantillas definen cómo se presenta la información en la interfaz de usuario.

### 15.1. Plantilla para el Formulario (`form.html`)

Listing 16: Plantilla para el Formulario

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registrando Usuario</title>
</head>
<body>
    <h1>{{ Label|upper }}</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Registrar</button>
    </form>
</body>
</html>
```

### 15.2. Plantilla de Éxito (`success.html`)

Listing 17: Plantilla de Éxito

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> xito </title>
</head>
<body>
    <h1>REALIZADO CON XITO</h1>
</body>
</html>
```

## 16. Configuración de URLs (`urls.py`)

La configuración de URLs es esencial para enrutar las solicitudes a las vistas correspondientes.

Listing 18: Configuración de URLs

```
# urls.py
from django.contrib import admin
from django.urls import path
from users.views import addStudent
from courses.views import create_curso, list_cursos, addGrade, index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('sistema/addStudent/', addStudent, name='addStudent'),
    path('sistema/', index, name='index'),
    path('/', index, name='index'),
    path('sistema/create_curso/', create_curso, name='create_curso'),
    path('sistema/list_cursos/', list_cursos, name='list_cursos'),
]
```

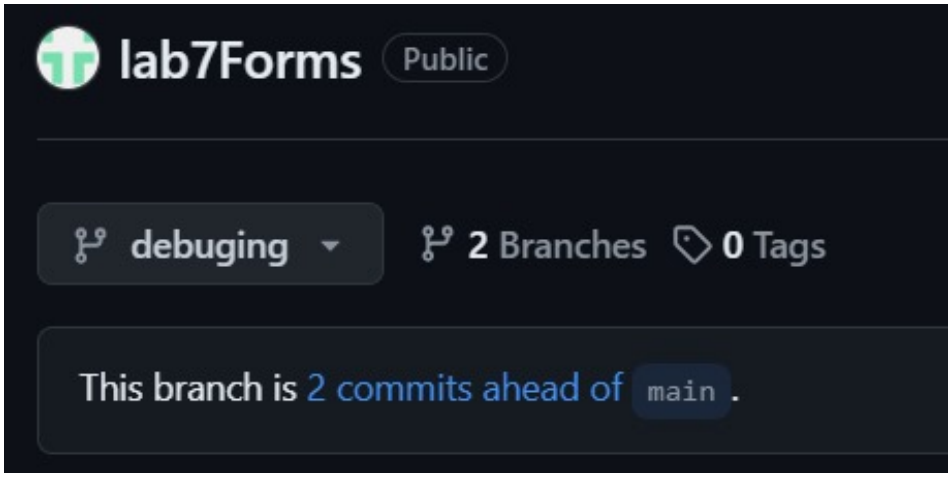
**Explicacion del Codigo:**

- `path('admin/', admin.site.urls)`: URL para la administración de Django.
- `path('sistema/addStudent/', addStudent, name='addStudent')`: URL para agregar un nuevo estudiante.
- `path('sistema/', index, name='index')`: URL para la página de inicio.
- `path('sistema/create_curso/', create_curso, name='create_curso')`: URL para crear un curso.
- `path('sistema/list_cursos/', list_cursos, name='list_cursos')`: URL para listar los cursos.

# 17. Branch

## 17.1. URL de Repositorio de la Rama en Github

- URL de la rama de GitHub para clonar o recuperar.
- `https://github.com/rikich3/lab7Forms/tree/debuging`




## 17.2. Funcionalidad

- Esta rama fue creada para modificar los modelos y de esta forma realizar las migraciones correctamente y una vez que esto ya este realizado vamos a seguir utilizandolo para realizar pruebas sobre los demás fetures que ofrece django.
- El primer commit realizado en esta rama es sobre la elaboracion del reporte en Latex.

# 18. 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

# 19. Rúbricas

## 19.1. 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 | 2.5 | |
| **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. 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 | |