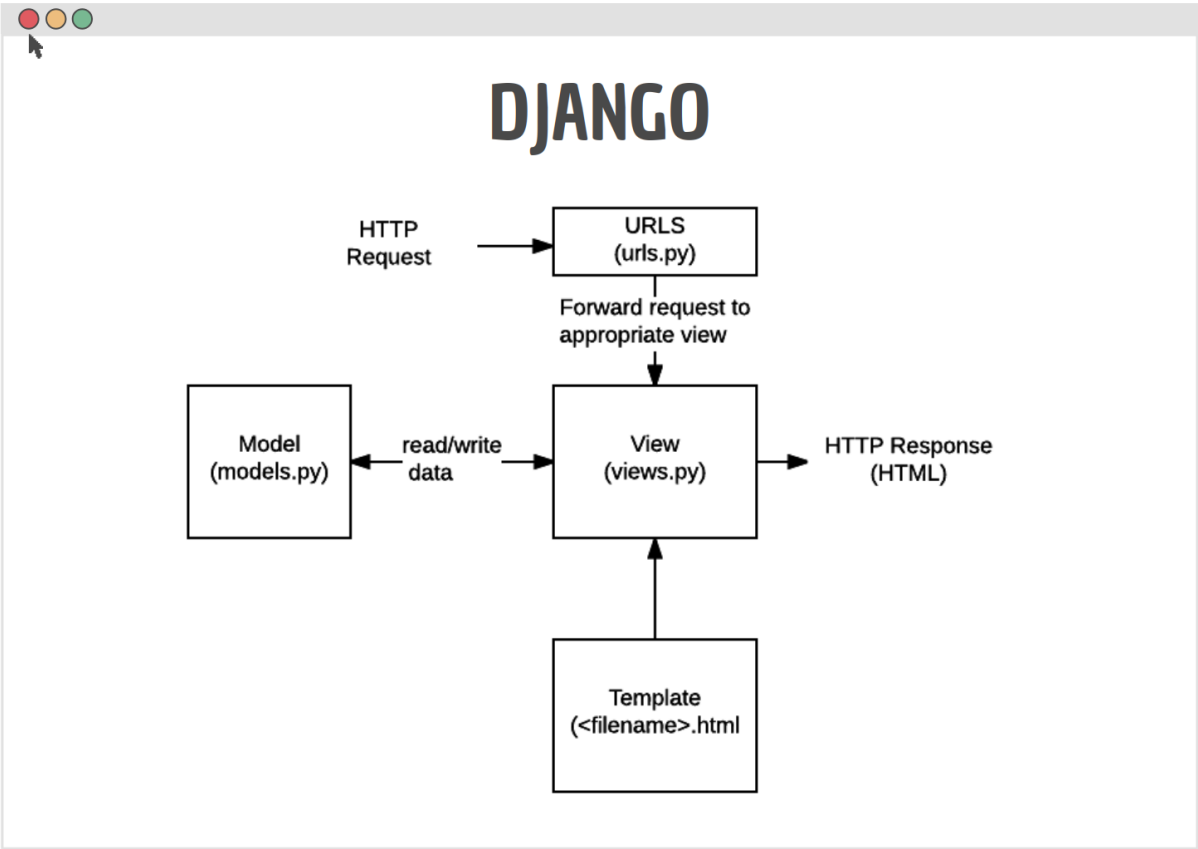
PROGRAMACIÓN WEB 2



Profesor(a):
Corrales Delgado, Carlo Jose Luis

Estudiantes:
Mamani Huarsaya, Jorge Luis
Velarde Saldaña Jhossep Fabritzio

Repositorio GitHub:
https://github.com/jorghee/django-projects

29 de mayo, 2024

Modelos de Datos

Los modelos de datos son la columna vertebral de cualquier aplicación Django, ya que definen la estructura y la relación entre los datos. En nuestro sistema, hemos definido tres modelos principales:

- **Student:** Este modelo representa la entidad de los estudiantes. Cada instancia de este modelo contiene información sobre un estudiante específico, incluyendo su nombre, apellido y correo electrónico. Al definir estos campos como `CharField` y `EmailField`, podemos aplicar restricciones como la longitud máxima y el formato de correo electrónico válido.

```python
class Student(models.Model):
  first_name = models.CharField(max_length=50)
  last_name = models.CharField(max_length=50)
  email = models.EmailField(unique=True)
```

- **Course:** Este modelo representa los cursos ofrecidos por la universidad. Cada curso tiene un nombre descriptivo y un código identificador único. La inclusión de un campo code único garantiza la integridad de los datos y facilita la búsqueda y gestión de cursos en la base de datos.

```python
class Course(models.Model):
  name = models.CharField(max_length=100)
  code = models.CharField(max_length=10, unique=True)

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

- **StudentGrade:** Este modelo establece la relación entre estudiantes y cursos, almacenando las calificaciones de los estudiantes en cada curso. Utiliza claves externas `ForeignKey` para establecer relaciones con los modelos de Estudiante y Curso. Además, utiliza un campo grade como DecimalField para almacenar la calificación con precisión.

```python
class StudentGrade(models.Model):
  student = models.ForeignKey(Student, on_delete=models.CASCADE)
  course = models.ForeignKey(Course, on_delete=models.CASCADE)
  grade = models.DecimalField(max_digits=5, decimal_places=2)

  def __str__(self):
    return f"{self.student} - {self.course}: {self.grade}"
```

Formularios

Facilitan la entrada de datos al sistema. En nuestro caso, hemos definido formularios correspondientes a cada modelo para permitir la creación de nuevos registros:

- **StudentForm:** Este formulario permite la creación de nuevos estudiantes. Está vinculado al modelo `Student` y proporciona campos para ingresar el nombre, apellido y correo electrónico del estudiante.

```python
class StudentForm(forms.ModelForm):
  class Meta:
    model = Student
    fields = ['first_name', 'last_name', 'email']
```

- **CourseForm** Similar al formulario de estudiantes, este formulario facilita la creación de nuevos cursos utilizando el modelo `Course`. Los campos incluyen el nombre del curso y su código identificador único.

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

- **StudentGradeForm** Este formulario permite ingresar las calificaciones de los estudiantes por curso, utilizando el modelo `StudentGrade`. Proporciona una lista desplegable de estudiantes y cursos disponibles, junto con un campo para ingresar la calificación.

```python
class StudentGradeForm(forms.ModelForm):
  class Meta:
    model = StudentGrade
    fields = ['student', 'course', 'grade']
```

Vistas

Las vistas en Django controlan la lógica de negocio de la aplicación y determinan qué datos se presentan al usuario y cómo interactúa el usuario con ellos. En nuestro sistema, hemos definido las siguientes vistas:

- **home:** Esta vista renderiza la página de inicio del sistema, que proporciona enlaces para agregar estudiantes, cursos y notas.

```python
from django.shortcuts import render
from .models import Student, Course, StudentGrade
from .forms import StudentForm, CourseForm, StudentGradeForm

def home(request):
  return render(request, 'grades/home.html')
```

- **create_student:** Procesa la creación de nuevos estudiantes. Si se recibe un formulario válido, se guarda el nuevo estudiante en la base de datos.

```python
def create_student(request):
  if request.method == 'POST':
    form = StudentForm(request.POST)
    if form.is_valid():
      form.save()
      return redirect('home')
  else:
```

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

- **create_course:** Maneja la creación de nuevos cursos. Similar a **create_student**, guarda el nuevo curso en la base de datos si el formulario es válido.

```
def create_course(request):
  if request.method == 'POST':
    form = CourseForm(request.POST)
    if form.is_valid():
      form.save()
      return redirect('home')
  else:
    form = CourseForm()
  return render(request, 'grades/create_course.html', {'form': form})
```

- **create_student_grade:** Esta vista gestiona la creación de nuevas notas de estudiantes por curso. Al igual que las vistas anteriores, guarda la nueva nota en la base de datos si el formulario es válido.

```
def create_student_grade(request):
  if request.method == 'POST':
    form = StudentGradeForm(request.POST)
    if form.is_valid():
      form.save()
      return redirect('home')
  else:
    form = StudentGradeForm()
  return render(request, 'grades/create_student_grade.html', {'form': form})
```

- **list_students_grades:** Se encarga de la generación de la lista de todos los estudiantes con sus respectivos cursos y su nota en dicho curso.

```
def list_student_grades(request):
  grades = StudentGrade.objects.select_related('student', 'course').all()
  return render(request, 'grades/list_student_grades.html', {'grades': grades})
```

Configuración de URLs

Para que nuestro sistema funcione correctamente, es necesario configurar las rutas (URLs) que manejan las solicitudes de los usuarios. Hemos realizado las siguientes modificaciones en los archivos urls.py tanto del proyecto principal como de la aplicación `grades`:

URLs del Proyecto Principal

```
from django.contrib import admin
from django.urls import path, include

```

```
urlpatterns = [
  path('admin/', admin.site.urls),
  path('', include('grades.urls')),
]
```

En este archivo, usamos `include('grades.urls')` para redirigir cualquier solicitud que no sea para la administración al archivo `urls.py` de la aplicación `grades`.

URLs de la Aplicación

```
from django.urls import path
from .views import home, create_student, create_course, create_student_grade,
↪  list_student_grades

urlpatterns = [
  path('', home, name='home'),
  path('create-student/', create_student, name='create_student'),
  path('create-course/', create_course, name='create_course'),
  path('create-student-grade/', create_student_grade, name='create_student_grade'),
  path('list_student_grades/', list_student_grades, name='list_student_grades'),
]
```

En este archivo, definimos las siguientes rutas.

- `path('', home, name='home')`: Ruta para la vista de inicio.
- `path('create-student/', create_student, name='create_student')`: Ruta para la vista de creación de estudiantes.
- `path('create-course/', create_course, name='create_course')`: Ruta para la vista de creación de cursos.
- `path('create-student-grade/', create_student_grade, name='create_student_grade')`: Ruta para la vista de creación de notas de estudiantes.
- `path('list_student_grades/', list_student_grades, name='list_student_grades')`: Ruta para la creación de la lista con las notas de los estudiantes de cada curso.

Plantillas HTML

Las plantillas HTML determinan cómo se presenta la información al usuario en un navegador web. Hemos creado plantillas para representar las diferentes páginas del sistema:

- **home_html:** La página de inicio muestra enlaces para agregar estudiantes, cursos y notas. Proporciona una interfaz fácil de usar para navegar por el sistema.
- **create_student.html, create_course.html, create_student_grade.html:** Estas plantillas representan los formularios de creación de estudiantes, cursos y notas respectivamente. Cada una contiene campos de formulario apropiados y un botón de envío para enviar los datos al servidor.